Volume 14 Number 2 April 1990
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Informatica

A Journal of Computing and Informatics

Subscription Information

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

The subscription price for 1990 (Volume 14) is US$ 30 for companies and US$ 15 for individuals.

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

Printed by Tiskarna Kresija, Ljubljana.

Informacija za naročnike

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

Letna naročnina v letu 1990 (letnik 14) se oblikuje z upoštevanjem tečaja domače valute in znaša okvirno za podjetja DEM 16, za zasebnike DEM 8, za študente DEM 4, za posamezno številko pa DM 5.

Številka žiro računa: 50101–678–51841.

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

Tisk: Tiskarna Kresija, Ljubljana.

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

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

Informatica

A Journal of Computing and Informatics

Subscription Information

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

The subscription price for 1990 (Volume 14) is US$ 30 for companies and US$ 15 for individuals.

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

Printed by Tiskarna Kresija, Ljubljana.

Informacija za naročnike

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

Letna naročnina v letu 1990 (letnik 14) se oblikuje z upoštevanjem tečaja domače valute in znaša okvirno za podjetja DEM 16, za zasebnike DEM 8, za študente DEM 4, za posamezno številko pa DM 5.

Številka žiro računa: 50101 – 678 – 51841.

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

Tisk: Tiskarna Kresija, Ljubljana.

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

Pri financiranju časopisa **Informatica** *sodeluje Republiški komite za raziskovalno dejavnost in tehnologijo, Tržaška 42, 61000 Ljubljana*

Volume 14 Number 4 October 1990
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Letnik 14 Številka 4 Oktober 1990 YU ISSN 0350 – 5596

Informatica

Časopis za računalništvo in informatiko

V S E B I N A

A SYSTEM FOR MORPHOLOGICAL ANALYSIS OF THE SLOVENE LANGUAGE

INFORMATICA 4/90

Tomaž Erjavec
NLU Lab., Department of Computer Science and Informatics
Jožef Stefan Institute
Jamova 39, 61111 Ljubljana

Keywords: morphological analysis, Slovene language.

ABSTRACT: Most computer programs dealing with natural language processing require access to a lexicon in computer readable format. Finding the entry in the lexicon that corresponds to the word-form in the text being processed is the domain of the morphological component. The paper presents an integrated environment for decomposing input word-forms into their constituent morphemes (morphological analysis). The system comprises a morpheme oriented lexicon, a module for updating the lexicon, and a two-level morphological analysis/synthesis unit. The system is written to provide a coverage of Slovene inflectional morphology. The basic paradigms and lexical alternations of word forms are handled by the lexicon system, while the two-level component takes care of the phonologically induced alternations.

1. Introduction

Natural language processing is a rapidly expanding field of computer applications. Programs dealing in NLP range from spelling-checkers to automatic translation systems and natural language interfaces. A basic component of most such systems is the lexicon. The lexicon is needed to store the legal words of the language together with their morpho-syntactic and other (depending on the application) properties.

In this paper we deal with the problem of designing the lexicon and the program for accessing it in such a manner as to allow mapping from word-forms which occur in the text being processed to lexical entries (Erj90a). The problems that occur here are of a morphological nature. To take an example from Slovene: for the system to know that the word-form of the Slovene noun *'telesa'* is the singular genitive or plural/dual nominative or accusative of the noun *'telo'* (*body*), it must know (implicitly or explicitly) that *'teles'* is a alternation of the morpheme *'telo'* which occurs when it is followed by another (non-null) morpheme, and furthermore that the morpheme *'telo'* denotes a noun of neuter gender, which has the ending *'-a'* in the above mentioned number/case combinations.

A possible way around all this complexity could be to simply store all the word-forms of all inflecting words in the lexicon along with their (morpho-syntactic and other) properties. While this approach is readily applicable to morphologically impoverished languages, such as English, it is much more questionable for highly inflected Slovene. Lengthy paradigms of Slovene verbs, nouns and adjectives (e.g. about 50 word-forms for a typical verb) would make the lexicon much larger and highly redundant. With the dropping price of computer storage this alone may not sound like a sufficiently convincing argument. However, even if we were willing to pay the price of such an inflated lexicon, we would still only temporarily evade the problem of (Slovene) morphology. Alas, the morphological knowledge would only be removed form the 'output' module, but would still have to be in the system, namely in the 'input' module, that is, in the part of the system that deals with the input of new words into the lexicon. Imagine otherwise the task of adding a new verb and all its fifty word forms with their specific properties into the lexicon.

The problem of lexicon update is especially relevant for the Slovene language, as there still doesn't exist a lexicon of the language in computer readable (much less usable - i.e. properly structured) form. This is also the reason why our system also includes an input module, which is described in the last part of the paper.

2. The two-level model

The two-level model of Kimmo Koskeniemmi (Kos84, Kos85) was selected as the fundamental scheme for our morphological analyzer. The morphological analyzer (MA) is the component of the system that takes as its input the word-form as it appears in the text - the textual or surface representation - and matches it to an entry in the lexicon - the lexical or deep representation - which is composed of the constituent morphemes of the word-form. The change of form from surface to deep representation is described by two-level rules, which of course differ from language to language. Succinctly put, the two-level model (or program) checks whether a lexical string and a textual string are in a legal correspondence; the name of the model derives from the fact that the rules mediate directly between the two representations.

Our choice of this model was influenced - among other things - by its prevalence in current (computer) morphological studies, which makes it well documented and thus easy to implement, as well as simplifying the task of writing the rules for (phonologically) induced alternations of Slovenian word-forms (Erj89, Erj90a).

We will describe the model only briefly, as it has been extensively dealt with in other publications (e.g. Erj90b). The two-level model has its root in phonological rewriting rules and deals with phono-morphological alternations which occur in 'joining' the morphemes of a language. Thus, if we were to have in our lexicon the root of the noun *'ladj'* (*ship*) along with the information that this morpheme can be followed by any morpheme for the endings of the first female declension of nouns ('*-a*' for nominative singular, '*-e*' for genitive singular and nominative plural, etc.), then a valid lexical word form would be for instance *'ladj-0'*, that is the root, followed by a morpheme boundary and the ending for genitive plural ('zero' morpheme/symbol). The surface word form for this case/number is, however, *'ladij'*, this alternation being caused by the phonology of Slovene; it isn't possible to pronounce a obstruent + *'j'* in word final position.

Our rule (somewhat simplified here) for such cases is:

```
"i epenthesis"
   0:i <=> Obst _ j -:0 #:0 ;
```

To spell it out: the surface symbol 'i' should correspond to the lexical '0' if and only if it is preceded by any symbol from the set 'Obst' which is elsewhere defined as the set of obstruents, and followed by a morpheme boundary and word boundary (in other words a zero ending). We should point out, that the the lexical alphabet is in principle different from the surface alphabet.

Let us conclude the description of the two level model by the description of the algorithm. The two-level rules are first compiled into finite automata (Erj90a, Kar87, Kos86), which take as their input pairs of lexical/surface symbols. On analyzing a surface word form, the automata are fed pairs of such symbols, the surface part of the pairs coming from the textual string and the lexical part from the lexicon. All the automata operate in parallel. If at the end of the surface string no automata have blocked and all are in their final states, then the correct lexical word was found in the lexicon. In such a way the automata actually 'guide' the search procedure in the lexicon into choosing the correct lexical entry, if one exists. The structure of the lexicon which supports such search is dealt with in the next section.

3. The lexicon

A basic part of our system is the lexicon (Dal87, Erj89a), which is composed of sub-lexicons. We can, for instance, have a sub-lexicon for roots, another for endings of a male noun declension of Slovene, another for the conjugation endings of certain verbs, etc. A certain set of sub-lexicons is set as initial, meaning that a (recognizable) word can only start with a member of these sub-lexicons. The other sub-lexicons are connected to initial sub-lexicons by means of pointers, typically making them inflectional paradigms of various word classes.

We shall call the elements of the sub-lexicons lemmas. Each lemma has three parts: the (tentatively speaking) morpheme, information about the possible continuations of the morpheme and its inherent morpho-syntactic (or other) properties, e.g.:

```
bolezen decl_subst_f / eng=illness cat=subst gen=f;
```

This lemma contains the information that the morpheme 'bolezen', in order to make a legal lexical word-form, must be continued with a morpheme from the sub-lexicon named 'decl_subst_f2, and that its properties are that it has the English translation 'illness', and is a noun of female gender. For another example, let us look at a lemma from the sub-lexicon 'decl_subst_f2':

```
-0        #          / nu=sg cas=nom;
```

The lemma states that the nominative singular ending for the second female declension of nouns is a null ending '-0'; a morpheme boundary, followed by a 'zero' symbol. Furthermore, this morpheme has no continuation.

A lexical word-form is thus constructed in the following manner:

* start with the 'empty' lexical string and one of the initial lexicons as the 'active' lexicon;
* choose a lemma in the active lexicon;
* append the morpheme of the lemma to the lexical string;
* mark the continuation lexicon of the lemma as active;
* if the active lexicon is different from '#', then go to step two.

Except in certain special cases (explained in Erj90a) the properties of a lexical word-form are simply the concatenation of the properties of constituent morphemes.

Of course, the purpose of the system is to find a quite specific word-form in the lexicon; namely the one (or ones in cases of ambiguity) that correspond(s) to the surface word-form being analyzed. For this purpose, all the sub-lexicons are first compiled into an internal format. All the morphemes of each sub-lexicon are converted into a letter-tree, i.e. a tree with the name of the lexicon as the root, symbols (letters) of the lexical alphabet as internal nodes and (pointers to) lemmas as leaves. The nodes on a path from the root to a leaf constitute a morpheme of the leaf lemma.

With such a data structure it is then a simple matter for the two-level automata to licence a path down the current tree, backtracking in cases of an ill chosen path.

4. Lexicon interconnection

As we saw, the lexicon system takes care of regular paradigms of the inflecting words of the language, while the two-level rules handle phono-morphological alternations. The Slovene language, however, abounds in alternations that are lexically conditioned. That is not to say that no two-level rules can be constructed to cover these alternations, but rather that they are not (purely) phonologically conditioned. There is for instance an alternation that affects only nouns of male gender which have the "animate" property, and another one which pertains only to the plural and dual of certain Slovene nouns (lengthening of the root with *'ov'*). As two-level rules are sensitive only to the form of the word (string) they are processing, they are inappropriate for expressing such alternations.

To handle lexically conditioned types of alternations, we have concentrated on the linking mechanism between the sub-lexicons. The "continuation" information belonging to a lemma can also include, along with a pointer to another sub-lexicon, a pointer to a description of how to modify the following sub-lexicon to express the desired alternation.

To make the point clearer, we give a simple case of an alternation that affects some nouns of the male gender. In addition to the standard ending for genitive singular *'-a'*, these nouns can also have the ending *'-u'* (e.g. *'sin-a'* as well as *'sin-u'* - *(of) son*):

```
sin decl_subst_m1(gen_u) / eng=son cat=subst gen=m;
```

The continuation information for such nouns includes (together with the pointer linking them with the sub-lexicon of male gender declension endings) the name of a 'transformational' rule. On accessing the continuation sub-lexicon, it is first (locally) modified according to this rule. In our 'gen_u' case, a lemma '-u' with the morpho-syntactic attributes of genitive case and singular number is added to the continuation sub-lexicon.

In addition to adding a lemma, these 'lexical alternations' are able to perform the following operations on sub-lexicons:

* delete a lemma;
* substitute a lemma;
* prefix a sub-lexicon;
* split a sub-lexicon according to given criteria;
* merge two sub-lexicons.

5. Input of new words

Computer lexicons for various applications should be of an "open" type, i.e. they should allow easy entry of new words into the lexicon by an inexperienced user. In our case this means that we cannot expect the user to enter new lemmas as they will appear in the lexicon, since this would require knowledge of the implementation details of the system; e.g. two-level rules, the lexical alphabet, lexical alternations, etc.

The user is therefore only expected to enter (Erj90):

* the "canonical" form of the word (e.g. nominative singular for nouns) the "comparative" word-form of the word (e.g. genitive plural of the noun);
* the basic paradigm;
* certain vital properties of the word (e.g. noun, male, animate).

Both word forms are input in the symbols of the surface alphabet.

The lexicon input module then automatically determines:

* the lexical root of the word (removing the suffix of the canonical word-form and mapping it into lexical symbols);
* the continuation sub-lexicon(s);
* lexical alternations (if any);
* inherent morpho-syntactic properties of the word.

The algorithm works roughly as follows:

* compare the canonical word-form with the comparative word-form to determine (possible) lexical alternations;
* nondeterministically map the canonical word-form into the symbols of the lexical alphabet;
* extract the root from the (lexical) canonical word-form;
* from the paradigm and the morpho-syntactic properties of the comparative word-form determine which lexical ending the comparative word-form should have;
* add this ending to the lexical root, getting the comparative word-form in its lexical form;
* check the lemmatization by two-level matching of derived (lexical) and entered (surface) comparative word-forms;
* if the match succeeds, finish, else redo any of the first three steps in a different way.

6. System structure

We have so far discussed all the separate components of our system; now for a run down of the whole system. The system was implemented on VAX/VMS in Quintus Prolog and consists of the following parts:

* the compiler, which takes as its input two-level rules and produces final state automata (transducers);
* the lexicon module that provides a user interface for the creation and updating of the lexicon;

* the lexicon output module, which is responsible for passing lexical word forms to the MA module.
* the MA module itself, which, having access to the transducers and (indirectly) to the lexicon, is able to analyze Slovene word forms into their lexical counterparts, and also to synthesize word forms from lexical data.

7. Conclusions

Our system is one of the first attempts toward the goal of producing a computer usable lexicon of Slovene words that could be used as a front-end to various natural language processing programs.

So far we have concentrated mainly on the morphology of noun and adjective declensions (Top84), while the (very complicated) morphology of the verb still needs to be worked out. A morphological analysis system for "true life" applications should also be written with perfomance characteristics in mind; i.e minimizing its time complexity and storage reguirements for a lexicon with cca. ten thousand lemmas. The task of keying in all these words and their properties also seems daunting.

In my opinion, the effort would well be worth the trouble, as languages which will not become "computerized" in a very short time face a bleak future, since they could very well be reduced to the status of a local inferior dialect.

References:

(Dal87) Dalrymple M., Kaplan R., Karttunen L., Koskenniemi K., Shaio S., Wescoat M.: **Tools for Morphological Analysis** / Xerox Palo Alto Reasearch Center, Center for the Study of Language and Information, Stanford University, Report No. CLSI-87-108, 1987

(Erj89) Erjavec T., Tancig P.: **Dvo-nivojska pravila za alternacije slovenskih samostalniških sklanjatev (Two-Level Rules for Alternations of Slovene Noun Declensions)** / Proceedings of "V. kongres zveze društev za uporabno jezikoslovje Jugoslavije"; Ljubljana 1989

(Erj89a) Erjavec T., Tancig P.: **Struktura računalniškega leksikona z morfemskimi entitetami; primer slovenščine (The structure of a computer lexicon with morpheme entities for the Solovene Language)** / "IV. Jugoslovanska konferenca o leksikografiji in leksikologiji 'Rječnik i društvo'" Conference Proceedings; Zagreb, 1989

(Erj90) Erjavec T., Tancig P.: **Vnos samostalnikov v odprti leksikon slovenščine (Input of Nouns into an Open Lexicon for the Slovene Language)** / "Informatička tehnologija u primenjenoj lingvistici" Conference Proceedings; Zagreb, 1990

(Erj90a) Erjavec T.: **Računalniški sistem za morfološko analizo in sintezo slovenskega jezika (A Computer System for Morphological Analysis and Synthesis of the Slovene Language)** / Masters Thesis; Faculty for Electrical Engineering and Computer Sciences, University of Ljubljana; Ljubljana, 1990

(Erj90b) Erjavec T.: **The Development and Description of the Two-Level model for Morpholological Analysis and Synthesis** / Journal "Informatica" Vol. 14, No. 3.; Ljubljana 1990

(Kar87) Karttunen L., Koskenniemi K., R.M. Kaplan: **A Compiler for Two-level Phonological Rules** / in "Tools for Morphological Analysis"; Xerox Palo Alto Reasearch Center, Center for the Study of Language and Information, Stanford University, Report No. CLSI-87-108; 1987

(Kos84) Koskenniemi K.: **A General Computational Model for Word-Form Recognition and Production** / Computational Linguistics, Conference Proceedings; COLING '84

(Kos85) Koskenniemi K.: **A General Computational Model for Word Form Recognition and Production** / Computational Morphosyntax, Report on Research 1981-84; University of Helsinky, Dept. of General Lingusistics Publications No. 13

(Kos86) Koskenniemi K.: **Compilation of Automata from Morphological Two-Level Rules** / Papers form the fifth Scandinavian Conference of Computational Linguistics 1985; University of Helsinky, Dept. of General Lingusistics Publications No. 15

(Top84) Toporišič J.: **Slovenska slovnica (Grammar of the Slovene Language)** / Založba Obzorja Maribor; 1984

Keywords: algebra, blindness, breakdown, cultural understanding, disorder, formalization, hermeneutics, idea of god as formalized information, information, informational expert system, informational algebra, informational logic, intelligence, intellectualism, loseableness, losing, misunderstanding, semiotics, understanding.

Anton P. Železnikar
Volaričeva ul. 8
61111 Ljubljana, Yugoslavia

Understanding as information and its formalization can bring to the surface various formal informational concepts that could substantially change our knowledge of understanding as a basic as well as the most complex realm of informational entities. Understanding becomes a formal informational entity that informs spontaneously and circularly, in a hermeneutic (historically cyclic) or parallel-cyclic mode. Understanding can be formally decomposed into its constituting components, for instance, sensing, observing, perceiving, conceiving, and comprehending, all informing in their own and perplexed hermeneutic way. Formal axioms of understanding can be developed in a straightforward manner, constituting the basis of the algebraic theory of understanding. Intelligence and intellectualism can be treated as distinct aspects which pervade other components of understanding. Meaning is the resulting information arising out of the process of understanding. Further, hermeneutics as a modus of understanding can be formalized by the introduction of the specific structure of informational cycles within understanding. Formalization of semiotics introduces the so-called semiotic type of understanding, considering syntax, semantics, and pragmatics as distinct, however perplexed informational entities. Several types of loseableness and losing of understanding can be formalized and perplexed, leading to the problem of the understanding blindness and breakdown of the performing blindness. Informational expert systems can be formalized in the form of dedicated formal informational systems of understanding applying hermeneutic as well as direct parallel-cyclic modes for production of expertise. Misunderstanding can be formally treated as an informationally incompatible system with regard to the ruling understanding. It is shown how disorder and cultural understanding (informational cultivation) could be formally decomposed. At the end of the essay an informational formalization of the idea of god (universal information) in an algebraic manner is presented. The aim of the formalization of understanding by algebraic means is to step vigorously on the way to an informational theory of understanding in the living as well as artificial. By the presented algebraic approach, artificial understanding can be directed into a new enlightenment and applicative perspective. On the other hand, the formal approach of understanding could prepare the ground from which various philosophical excursions into the domain of understanding would become possible.

RAZUMEVANJE KOT INFORMACIJA II

Razumevanje kot informacija in njegova formalizacija lahko izpostavita različne informacijske koncepte; ti lahko bistveno spremenijo znanje o razumevanju kot osnovno in kot kar najbolj zapleteno področje informacijskih entitet. Razumevanje postane formalna informacijska entiteta, ki informira spontano in cirkularno, na hermenevtičen (zgodovinsko cikličen) ali paralelno-cikličen način. Razumevanje je mogoče formalno razstaviti v sestavljajoče komponente, npr. v občutenje, opazovanje, zaznavanje, snovanje in doumevanje, ki vse informirajo na lasten in prepleteno hermenevtičen način. Formalne aksiome razumevanja je mogoče razvijati dovolj natančno, oblikujoč osnovo za algebraično teorijo razumevanja. Intelogenco in intelektualizem je mogoče obravnavati kot distinktna vidika, ki prežemata druge komponente razumevanja. Nadalje je mogoče formalizirati hermenevtiko kot modus razumevanja z vpeljavo specifične strukture

informacijskega cikla v okviru razumevanja. S formalizacijo semiotike se uvaja t.i. semiotični tip razumevanja, ki upošteva sintakso, semantiko in pragmatiko kot distinktne, toda prepletene informacijske entitete. Formalizirati in preplesti je mogoče več vrst zgubljenosti in izgube razumevanja, kar privede do problema razumevajoče slepote in prekinjanja nastopajoče slepote. Informacijske ekspertne sisteme je mogoče formalizirati v obliki namenskih formalnih informacijskih sistemov razumevanja z uporabo hermenevtičnih kot tudi paralelno-cikličnih načinov produciranja ekspertize. Nerazumevanje je mogoče formalno obravnavati kot informacijsko nezdružljivi sistem glede na vladajoče razumevanje. Pokazano je, kako bi bilo mogoče formalno razstaviti moteno (patološko) in kulturno razumevanje (informacijsko kultiviranje). Na koncu spisa je prikazana še informacijska formalizacija ideje o bogu (univerzalne informacije) na algebraičen način. Namen prikazane formalizacije razumevanja z algebraičnimi pripomočki je primerna usmeritev na pot k informacijski teoriji razumevanja, ki bi lahko zajela tako živo kot umetno (arteficialno). S prikazanim algebraičnim pristopom je prav umetno razumevanje mogoče nanovo osvetliti in aplikativno omogočiti. Razen tega pa formalni koncept razumevanja pripravlja ozadje iz katerega so mogoči raznovrstni filozofski izleti v domene razumevanja.

Part Two

... There can be little doubt that Formalism is the most popular anti-Platonist position among practicing mathematicians. ...

Penelope Maddy (RCP) 1122

A Formal Theory of Understanding as Information

15. On the Sense of Formalization of Understanding

... according to Goedel, mathematical intuition inspires us to build up theories which are then justified by their ability to systemize all of mathematics. ...

Penelope Maddy (RCP) 1133

The question of the sense of formalization of understanding as information has to be put into consideration at the beginning of the discourse which follows, for understanding is the fundament of informational arising, from the most primitive informational form to the most complex informational processes of understanding. Is such formalization at all possible and, if it is, to which extent and practical (philosophical, mathematical, technological) relevance? What new quality of formal understanding does this theory bring to the surface and how can it impact our symbolic presentation and imagination? How does this theory root in the basic theory of information (for instance, in the so-called informational logic and informational algebra) which was described in (IL-I, IL-II, IL-III, IL-IV, IIA)?

Understanding as information is an ongoing, continuous informational process, that is, an implicit or explicit informational variable (entity, an acting or operating operand) which will be marked by the symbol $\mathfrak{U}$. Understanding $\mathfrak{U}$ as informational process understands or informs in an understanding way its own information (itself) and the arriving or to the understanding coming information, i.e., information coming into understanding. This style of explanation (expression) or of articulation of the problem of understanding is a specific style of linguistic expression and is necessary to bring the adequate formal apparatus into existence, to its semiotic appearance. In general, this, sometimes unconventional style of articulation, is characteristic for the entire realm of informational understanding of information, constituting the realm of informational philosophy and informational theory.

16. Understanding as a Formal Informational entity

... mathematics is neither a game with symbols nor pure metamathematics; rather, it is the study of logical consequences. Among philosophers, this view has been called If-thenism, and it has been embraced and ultimately rejected by three important figures in the philosophy of mathematics: Hilbert (before his program), Russel (before Principia), and Putnam (before his Platonism). ...

Penelope Maddy (RCP) 1124

Let us paraphrase the above quotation in the following way: informational theory is neither a game with informational operands and

informational operators nor pure metainformational theory; rather it is the study of informational (not only logical) consequences. Among philosophers, this view might be called Theoretic Informationalism, however, in cybernetics and informatics, it seems to be the way of postmodernistic understanding of information and the way of informational understanding of the cosmic (microcosmic and macrocosmic), the living (in biology, neural science) and the artificial (intelligent machines).

By marking of understanding as information by Gothic symbol $\mathfrak{U}$, we stress the ongoing informationally explicit or informationally implicit process of understanding. Thus, understanding $\mathfrak{U}$ will perform as regular informing of information, i.e., as a regular informational entity, variable, or operand, which informs itself and other information and is informed by itself and by other information. What is the meaning of this phenomenology in the case of understanding as information?

Generally, understanding $\mathfrak{U}$ informs and is informed in all possible ways, i.e.,

$$(\mathfrak{U}1) \qquad \mathfrak{U} \Leftrightarrow ((\mathfrak{U} \models) \vee (\models \mathfrak{U}))$$

Some comments to this formula are necessary:

(1) All operators ($\Leftrightarrow$, $\models$, and $\vee$) in this or any other informational formula are "informational", if there is not explicitly defined that they are particular, for instance, mathematical.

(2) Formula ($\mathfrak{U}1$) is completely open which is explicated by the right side of the operator $\Leftrightarrow$. That is, formula $\mathfrak{U} \models$ informs freely, i.e. spontaneously and circularly to something symbolized by the empty place on the right side of the operator $\models$. We can say that $\mathfrak{U}$ 'does not know yet' which informational entities somewhere could be informed by $\mathfrak{U}$, with the exception of $\mathfrak{U}$ itself. This yields,

$$(\mathfrak{U}2) \qquad \mathfrak{U} \Leftrightarrow (\mathfrak{U} \models \mathfrak{U})$$

(3) On the right side of operator $\Leftrightarrow$ in formula ($\mathfrak{U}1$) there is formula $\models \mathfrak{U}$ which symbolizes the completely free, i.e., spontaneous and circular informing of something (yet undetermined) to $\mathfrak{U}$, that is, in the way, where $\mathfrak{U}$ is informed by something not known yet. Formula $\models \mathfrak{U}$ hides the potentiality of $\mathfrak{U}$ to be informed by any information or the potentiality of any information to inform the understanding $\mathfrak{U}$ as information. Certainly, $\mathfrak{U}$ is informed by itself, thus, ($\mathfrak{U}2$) is informationally valid also in this particular case.

(4) Informational operator $\models$ is the most general informational operator (metaoperator, a kind of informational jockey operator) which can be replaced (particularized or, in a particular case, universalized) by any informational operator. Thus, operator $\models$ represents the entire class of possible informational operators which can be grouped as general ($\models$, $\nmid$, $\not\models$, $\not\nmid$), parallel ($\Vdash$, $\shortmid\!\!\shortmid$, $\not\Vdash$, $\not\shortmid\!\!\shortmid$), cyclic ($\vdash$, $\dashv$, $\nvdash$, $\not\dashv$), and parallel-cyclic operators ($\Vdash$, $\dashv\!\!\dashv$, $\nVdash$, $\not\dashv\!\!\dashv$) of informing and non-informing, respectively.

Instead of ($\mathfrak{U}1$), the so called system expression of understanding $\mathfrak{U}$ can be used, i.e.,

$$(\mathfrak{U}3) \qquad \mathfrak{U} \Leftrightarrow ((\mathfrak{U} \models); (\models \mathfrak{U}))$$

System $\mathfrak{U}$, i.e. understanding as information, is a composition of two distinct processes, $\mathfrak{U} \models$ and $\models \mathfrak{U}$, which can inform anything and can be informed by anything, respectively. Thus, informationally, the following is implied:

$$(\mathfrak{U}4) \qquad (\mathfrak{U} \models) \Rightarrow (\mathfrak{U} \models \mathfrak{U}); \quad (\models \mathfrak{U}) \Rightarrow (\mathfrak{U} \models \mathfrak{U})$$

Informational operator $\Rightarrow$ is read as 'implies/imply' or 'is/are implied'. Processes $\mathfrak{U} \models$ and $\models \mathfrak{U}$ imply informationally process $\mathfrak{U} \models \mathfrak{U}$ or, one can say, process $\mathfrak{U} \models \mathfrak{U}$ is implied informationally by process $\mathfrak{U} \models$ as well as by process $\models \mathfrak{U}$. Informational operator $\Rightarrow$ is not equivalent to the mathematical operator of implication, which is used, for instance, in an if-then clause.

The next question is, what does understanding as an acting form of information produce. As any informing, i.e. processing of information, understanding $\mathfrak{U}$ as informing is a component of understanding entity (operand) α as information. In this case one says that $\mathfrak{U}$ is a particular implicit informing of α and that α as information of understanding uses $\mathfrak{U}$ as its own processing, which performs the function of understanding. Thus, the so-called general informing of α, i.e., $\mathfrak{I}(\alpha)$, includes understanding $\mathfrak{U}(\alpha)$, yielding

$$(\mathfrak{U}5) \qquad \mathfrak{U}(\alpha) \subset \mathfrak{I}(\alpha)$$

Understanding $\mathfrak{U}$ of α is a specific process (subprocess) of informing $\mathfrak{I}$ of information α. This means that α can also inform in a non-understanding, misunderstanding, or disordering way, where the non-understanding component of α within $\mathfrak{I}(\alpha)$ is an informational difference occurring between $\mathfrak{U}(\alpha)$ and $\mathfrak{I}(\alpha)$. This difference can be marked by $\mathfrak{D}(\alpha)$, i.e.,

$$(\mathfrak{U}6) \qquad \mathfrak{D}(\alpha) \equiv \mathfrak{U}(\alpha) \setminus \mathfrak{I}(\alpha)$$

We see that understanding $\mathfrak{U}$ of α performs as regular information. Understanding $\mathfrak{U}$ as information informs and is informed as an autonomous informational entity, in a spontaneous and circular way, i.e.,

$$(\mathfrak{U}7) \qquad \mathfrak{U} \Leftrightarrow (\ldots ((\mathfrak{U} \models \mathfrak{U}) \models \mathfrak{U}) \ldots \models \mathfrak{U})$$

and within an informational realm α, by which it is impacted and which it impacts, as

$$(\mathfrak{U}8) \qquad \alpha, \mathfrak{U} \models \alpha, \mathfrak{U}$$

This yields

$$(\mathfrak{U}9) \qquad \alpha \models \alpha; \ \alpha \models \mathfrak{U}; \ \mathfrak{U} \models \alpha; \ \mathfrak{U} \models \mathfrak{U}$$

or according to ($\mathfrak{U}7$) also,

$$(\mathfrak{U}7') \qquad \mathfrak{U}(\alpha) \Leftrightarrow (\ldots ((\mathfrak{U}(\alpha) \models \mathfrak{U}(\alpha)) \models \mathfrak{U}(\alpha)) \ldots \models \mathfrak{U}(\alpha))$$

The four processes of ($\mathfrak{U}9$) can be expressed, for instance, in the sense of formula ($\mathfrak{U}7$), in the form:

(𝔄10)

(1) (·... ((α ⊨ α) ⊨ α) ... ⊨ α);
(2) (... ((α ⊨ 𝔄) ⊨ α, 𝔄) ... ⊨ α, 𝔄);
(3) (... ((𝔄 ⊨ α) ⊨ 𝔄, α) ... ⊨ 𝔄, α);
(4) (... ((𝔄 ⊨ 𝔄) ⊨ 𝔄) ... ⊨ 𝔄)

System (𝔄10) can be marked simply by a system of two entities (operands) (𝔄(α), α(𝔄)), with the meaning that α and 𝔄 create each other informationally and in themselves implicitly.

17. Informational Components of Understanding

> ... Platonism, of course, is the view that mathematics is the study of a mind-independent realm of abstract entities, that it is as much an objective science as astronomy, physics, or biology. At various points, these tenets conflict with each of the position considered so far - with Intuitionism on mind-independence, with Formalism on the meaningfulness of higher mathematics, with Logicism on the need for existence assumptions that go beyond pure logic - but the Platonist's traditional and starkest opponent is the Nominalist, who holds simply that there are no abstract entities. ...
>
> Penelope Maddy (RCP) 1130

What could in general the components of understanding be: thought, spirit, sense, wisdom, mind, consciousness, reason, comprehension, perception, conception, understanding itself, etc.? We see that all these components have a common meaning which is the consequence of their understanding, the process of their semantic and pragmatic investigation. However, to each of these components the so-called intelligence - a property of information as information - can be attributed as a general principle. Thus, understanding 𝔄 is a domain of informing of information joining all these components, which will be marked by $\mathfrak{A}_1$, $\mathfrak{A}_2$, ... , $\mathfrak{A}_n$.

What is the notion of informational component as an informational entity distinguished within another informational entity? What does the informational component constitute as an informational entity which arises within another arising entity?

An informational component is a unity (informational unit) which can be distinguished (observed, investigated, comprehended) as the unity in informational sense within another informational entity. A component as informational unity can be informationally rounded up, closed, or bounded within a composed informational entity. Informational distinguishing of components means observing the informational differences, setting the informational boundaries, determining the informational oneness, and marking the united component informationally. However, an informational entity as unity is not an informationally isolated form or process of information: it informs and is informed within its master entity, but can also inform directly itself and other information and can be informed directly by itself and other information (arising inside and outside of its master entity).

Let us mark canonically the possible components of understanding 𝔄 by $\mathfrak{A}_i$ (i = 1, 2, ... , n) and let these components be informationally active within 𝔄. Semantically, components of 𝔄 can be, for instance, on the level of a being's metaphysics as intelligence, thought, spirit, sense, wisdom, comprehension, perception, conception, and understanding by itself. The question is how do these components perform informationally within 𝔄.

Let us determine the performing of components $\mathfrak{A}_1$, $\mathfrak{A}_2$, ... , $\mathfrak{A}_n$ of understanding 𝔄, within 𝔄, in the following, basically parallel way:

(𝔄11)
$$(\mathfrak{A}_1 \models \mathfrak{A}) \Vdash \mathfrak{A}_1;$$
$$(\mathfrak{A}_2 \models \mathfrak{A}) \Vdash \mathfrak{A}_2;$$
$$\cdots\cdots\cdots$$
$$(\mathfrak{A}_n \models \mathfrak{A}) \Vdash \mathfrak{A}_n;$$
$$(\mathfrak{A} \models \mathfrak{A}_1) \Vdash \mathfrak{A};$$
$$(\mathfrak{A} \models \mathfrak{A}_2) \Vdash \mathfrak{A};$$
$$\cdots\cdots\cdots$$
$$(\mathfrak{A} \models \mathfrak{A}_n) \Vdash \mathfrak{A}$$

This is a parallel system within which processes $(\mathfrak{A}_i \models \mathfrak{A}) \Vdash \mathfrak{A}_i$; $(\mathfrak{A} \models \mathfrak{A}_i) \Vdash \mathfrak{A}$; i = 1, 2, ... , n inform in parallel and constitute the common or resultant understanding 𝔄. Simultaneously, components $\mathfrak{A}_1$, $\mathfrak{A}_2$, ... , $\mathfrak{A}_n$ inform particularly to understanding 𝔄, i.e., as $\mathfrak{A}_i \models \mathfrak{A}$ within $(\mathfrak{A}_i \models \mathfrak{A}) \Vdash \mathfrak{A}_i$; i = 1, 2, ... , n and understanding 𝔄 informs particularly to each component $\mathfrak{A}_1$, $\mathfrak{A}_2$, ... , $\mathfrak{A}_n$, i.e., as a process $\mathfrak{A} \models \mathfrak{A}_i$ within $(\mathfrak{A} \models \mathfrak{A}_i) \Vdash \mathfrak{A}$; i = 1, 2, ... , n. It can be imagined how this parallel system of understanding as informing of understanding components and understanding itself can become as complex as possible. Within this system, intelligence $\eta_{\mathfrak{A}}$ as information of understanding arises, characterized by an open and infinitely recursive formula of the form

(𝔄12) ... (... (($\mathfrak{A} \models \eta_{\mathfrak{A}}$) 𝔄) ... ⊨ $\eta_{\mathfrak{A}}$) ...

We see how informational playing of entities 𝔄 and $\eta_{\mathfrak{A}}$ can improve informational capabilities of both of them. Certainly, formula (𝔄12) is completely open for any other, outer informational impacting.

18. Axioms and Basic Rules of Understanding as Information (Formalization of Principles of Understanding)

> ... Logic is a matter of syntax, and syntax is a matter of convention. ...
>
> Penelope Maddy (RCP) 1126

18.1. Introduction.

The aim of this section is twofold: to show the concept of understanding in the elucidation of the so-called general (redefined) concept of information (POI), i.e., as an informational entity (for instance, informational operand and operator, in fact, informational process) and to formalize this concept by means of a distinguished, so-called informational language (formulas of operands, operators, and parentheses). The theory developed by this language should proceed from the general level, called metalevel of formulas; these can be particularized (decomposed) and again universalized (composed) stepwise, developing a formula or a system of formulas (for instance, a top-down and bottom-up design). At the beginning, a general syntax and semantics of operands, operators, and formulas of understanding as information is needed. Within this orientation it is possible to set some skeletal definitions.

(18.1) DEFINITION. The formulas for formatting understanding as informational operands are the following:

(1) $\mathfrak{A}$ marks an informational operand (entity) of understanding as information; the Gothic letter $\mathfrak{A}$ is the symbol of the so-called implicit informing of $\mathfrak{A}$, where $\mathfrak{A}$ is understood to be an implicit informational operator simultaneously, for instance, performing processes of understanding of information. Thus, the explicit role of understanding $\mathfrak{A}$ is the operand-like, while its implicit role is the operator-like.

(2) $(\mathfrak{A})$ is a parenthesized operand of understanding or parenthesized implicit operator of understanding.

(3) $\mathfrak{A} \models$ reads '$\mathfrak{A}$ informs'.

(4) $\models \mathfrak{A}$ reads '$\mathfrak{A}$ is informed'.

(5) $\mathfrak{A} \models \mathfrak{A}$ reads '$\mathfrak{A}$ informs $\mathfrak{A}$' or '$\mathfrak{A}$ is informed by $\mathfrak{A}$'. □

(18.2) DEFINITION. The informationally semantic perquisites to the previous definition are the following:

(1) $\mathfrak{A}$ can mark any operand or implicit operator of understanding, for instance, $\mathfrak{B}$, $\mathfrak{C}$, $\mathfrak{D}$, ..., indexed or not.

(2) $\mathfrak{A}$ can mark any multicomponent aggregate of understanding operands, i.e. $\mathfrak{A}_1$, $\mathfrak{A}_2$, ... , $\mathfrak{A}_n$ or $\mathfrak{A}_1$; $\mathfrak{A}_2$; ... ; $\mathfrak{A}_n$ or $\mathfrak{B}$, $\mathfrak{C}$, ... , $\mathfrak{D}$, etc.

(3) Symbol $\models$ is the general (explicit) operator of informing, called metaoperator; it can be replaced (particularized or universalized) by any operator. For instance, $\mathfrak{A} \models_T$ means '$\mathfrak{A}$ informs true'.

(4) Explicitly, $\models$ can be a unary, a binary, or a multiplex operator. In principle, $\models$ or its replacement is a multiplex operator. The case $\mathfrak{A} \models$ means that '$\mathfrak{A}$ informs to some yet unknown (or informationally irrelevant) and potentially unlimited number of operands'. Through this generalized concept, each informational formula (representing a simple or composite operand, i.e. operand entity) is open. It means, for instance, that in formula $\mathfrak{A} \models \mathfrak{B}$, entity $\mathfrak{A}$ can additionally inform, that is, $\mathfrak{A} \models$, and $\mathfrak{B}$ can additionally be informed, that is, $\models \mathfrak{B}$.

(5) Operand $\mathfrak{A}$ by itself (as an autonomous entity) has the property $\mathfrak{A} \models$ and $\models \mathfrak{A}$, i.e., performs as an open operand entity. Thus, $\mathfrak{A} \Leftrightarrow (\mathfrak{A} \models; \models \mathfrak{A})$ is an informational system where informational operator $\Leftrightarrow$ reads 'means'.

(6) All operators in a formula are informational. They can also be standard mathematical operators, if explicitly determined as such. Mathematical operators are particularizations of the informational ones. □

The previous two definitions constitute the formatting of the so-called operand formulas. In these formulas, operators $\models$ occur, for which it is possible to define the formatting formulas concerning operators. This might be senseful in processes of the so-called operator decomposition where the structure of an operator is becoming semantically relevant.

(18.3) DEFINITION. In general, we can introduce the following operator formatting formulas:

(1) $\models$ marks an operator (entity).

(2) In an operand formula of the general form $\mathfrak{A} \models$, operator $\models$ can be replaced in such a way that the new operand formula becomes
$\mathfrak{A} \models (\mathfrak{C} \models)$ or $(\mathfrak{A} \models \mathfrak{C}) \models$.
Thus,

$$\mathfrak{A} \models \Rightarrow ((\mathfrak{A} \models (\mathfrak{C} \models)); ((\mathfrak{A} \models \mathfrak{C}) \models))$$

(3) In an operand formula of the general form $\models \mathfrak{B}$, operator $\models$ can be replaced in such a way that the new operand formula becomes
$(\models \mathfrak{C}) \models \mathfrak{B}$ or $\models (\mathfrak{C} \models \mathfrak{B})$.
Thus,

$$\models \mathfrak{A} \Rightarrow (((\models \mathfrak{C}) \models \mathfrak{B}); (\models (\mathfrak{C} \models \mathfrak{B})))$$

(4) As one can see from (2) and (3), the principle of $\models$'s replacement is the following: $\models$ is replaced by $\models (\mathfrak{C} \models \xi)$ or by $(\xi \models \mathfrak{C}) \models$, where ξ marks the existing operand (also an empty operand place) of the original formula. This may be more evident if ξ is replaced by '...', thus, the replacement is $\models (\mathfrak{C} \models ...)$ or $(... \models \mathfrak{C}) \models$. □

(18.4) DEFINITION. In a formula, operator $\models$ can be replaced by the so-called operator composition of the form $\models \circ \models$, where $\circ$ is the sign of composition of operators' $\models$. A multiple composition must be parenthesized for the clear distinction of its parts; this might be useful in the process of decomposition of a composed operator. For instance, $(...((\models \circ \models) \circ \models)...) \circ \models$. □

A good example of operator composition is the case

$$\mathfrak{A} \models \mathfrak{B} \Leftrightarrow \mathfrak{A} \models_{\mathfrak{A}} \circ \models_{\mathfrak{B}} \mathfrak{B}$$

The interpretation of the right side of $\Leftrightarrow$ is the following: $\mathfrak{A}$ informs $\mathfrak{B}$, but only to the extent that $\mathfrak{A}$ can inform merely in the sense of $\mathfrak{A}$ itself, i.e. as $\mathfrak{A} \models_{\mathfrak{A}}$, and that $\mathfrak{B}$ can be informed merely in the sense of $\mathfrak{B}$, i.e. in the specific form $\models_{\mathfrak{B}} \mathfrak{B}$. This paradox leads to the decomposition of $\models$ in $\mathfrak{A} \models \mathfrak{B}$, to consider the components of $\models$, i.e. particularized operators $\models_{\mathfrak{A}}$ and $\models_{\mathfrak{B}}$. We recognize how each operator connecting two operands is, in its structure, always the affair (consequence, in fact understanding) of (belonging to) operands in

question, that is, a part of both operands concerning their common operator. In this respect, an operator symbolizes the relation concerning both operands and depending from both of them.

For instance, a mathematical formula a + b, with the informational meaning

$$a + b \Leftrightarrow a +_a \circ +_b b,$$

says that a and b as operands can be added, i.e., as formulas a $+_a$ and $+_b$ b guarantee this fact, however, in that mathematical case, operators $+_a$ and $+_b$ become identical. This property - the identity of components constituting an operator - is characteristic for mathematical operators. Therefore, in formula a + b, we never think in the sense of formula $a + b \Leftrightarrow a +_a \circ +_b b$.

A question has to be noted in regard to Definition (18.1) and Definition (18.3). What is the difference between these two definitions?

An operand formula of the form $\mathfrak{A} \models \mathfrak{B}$ can be transformed into the form $(\mathfrak{A} \models \mathfrak{C}) \models \mathfrak{B}$ by the application of rule (5) and (2) of Definition (18.1). On the other side, the same transformation of the form $\mathfrak{A} \models \mathfrak{B}$ can be obtained by the application of rule (4) of Definition (18.3), where $\models$ is replaced by the aggregate $(\ldots \models \mathfrak{C}) \models$. However, there is an essential semantic difference between both cases. In the first case, operand $\mathfrak{A}$ was decomposed into $(\mathfrak{A} \models \mathfrak{C})$, while in the second case operator $\models$ was decomposed as $(\ldots \models \mathfrak{C}) \models$. In this way, the new operand entity $\mathfrak{C}$ entered into new formula as a consequence of operand analysis, while in the second case, operand $\mathfrak{C}$ was a consequence of operator analysis. Of course, there may be no informational correlation between the both cases of appearance of $\mathfrak{C}$ in two different cases of decomposition.

18.2. An axiomatic basis of understanding

The intention of the philosophy of an axiomatic basis of understanding is to enable the development of informational formulas concerning understanding and being composed solely of operands, operators and pairs of parentheses. In the process of formula development, two subprocesses can be distinguished: formatting of formulas as determined in Section 18.1 and the semantic development of operands and operators concerning a "real" informational process of understanding, i.e., its needs, circumstances, and possibilities. The last subprocess embraces phenomenology of spontaneity, circularity, particularization, universalization, and like that, which is notionally concentrated within the so-called principles of information (POI), i.e. in the concept of informational arising. Informational operands, informational operators, as well as formulas themselves possess the nature of spontaneous and circular arising and are understood as arising entities.

For instance, in the realm of semantic development, informational operators can be decomposed in greater and greater details according to the proceeding of understanding; operands can be replaced by more detailed formulas, changed, or they can simply disappear; new formulas can be introduced into existing systems of formulas, concerning additional informational realities and details; systems of formulas can be paralleled by the use of particular operators and their circular perplexity can be developed unlimitedly. All these formal procedures are a consequence of the progressing understanding.

It is to stress that according to the principles of information (POI), operands as well as operators are arising (developing) entities. In this sense, these entities can and must be comprehended as arising (changing, growing, vanishing) entities. As we can imagine, the role of an informational operator is different in regard to a classical mathematical operator. This fact has to be taken into consideration when thinking and constructing informational operators. Some basic axioms will stress this fact over and over again. It is certainly necessary to step into this realm of thinking without any prejudice, for a symbol (also the mathematical one) can mark any abstract entity by which axiomatic systems can be constructed.

(18.2.1) AXIOM. The kernel idea of this axiom is the informing of an operand of understanding $\mathfrak{A}$. How does $\mathfrak{A}$ inform and how it is informed? The answer (or the assumption) is the formula

(a) $\quad \mathfrak{A} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A})$

Operand $\mathfrak{A}$ is in principle a completely open system, where operator $\models$ symbolizes the act of informing, that is, when entity $\mathfrak{A}$ informs, i.e., $\mathfrak{A} \models$, to yet unidentified entities and is informed, i.e., $\models \mathfrak{A}$, by yet unidentified entities (operands). Symbol $\Rightarrow$ reads 'implies'. In system (a), components $\mathfrak{A} \models$ and $\models \mathfrak{A}$ perform as two distinguished entities which can again inform and can be informed. Thus,

(a1) $\quad (\mathfrak{A} \models) \models; \models (\mathfrak{A} \models); (\models \mathfrak{A}) \models; \models (\models \mathfrak{A})$

is the system of the second rank of $\mathfrak{A}$'s informing. Inductively, the system of rank n of $\mathfrak{A}$'s informing can be constructed. This kind of recursion makes an operand entity (of understanding) completely open in transmitting information to and receiving information from yet unidentified operand entities. □

Applying Definition 1.4 to entities $\mathfrak{A} \models$ and $\models \mathfrak{A}$ yields

(a2) $\quad \mathfrak{A} \models \Leftrightarrow \mathfrak{A} \models_{\mathfrak{A}} \circ \models; \models \mathfrak{A} \Leftrightarrow \models \circ \models_{\mathfrak{A}} \mathfrak{A}$

This interpretation is senseful, for it shows the difference between composed operators $\models_{\mathfrak{A}} \circ \models$ and $\models \circ \models_{\mathfrak{A}}$, represented by $\models$ in case $\mathfrak{A} \models$ as well as in case $\models \mathfrak{A}$.

(18.2.2) AXIOM. An operand entity, marked by $\mathfrak{A}$, informs itself and is informed by itself:

(b) $\quad \mathfrak{A} \Rightarrow \mathfrak{A} \models \mathfrak{A}$

By this expression it is said that $\mathfrak{A}$ is also the mastery over itself, that $\mathfrak{A}$ 'arises' (grows, changes, vanishes in an informational way), i.e., informs actively (creatively)

within itself by itself. Because of (a), formula (b) remains open in the sense to impact and to be impacted by any other operand entity, in an informational way. □

(18.2.3) AXIOM. Let understanding $\mathfrak{A}$ inform understanding $\mathfrak{B}$, i.e., $\mathfrak{A} \models \mathfrak{B}$. The question, how can $\mathfrak{A}$ inform $\mathfrak{B}$, can be answered in the following way:

(c) $\mathfrak{A} \models \mathfrak{B} \Rightarrow (\mathfrak{A} \models \mathfrak{C}) \models \mathfrak{B}$;
(d) $\mathfrak{A} \models \mathfrak{B} \Rightarrow \mathfrak{A} \models (\mathfrak{C} \models \mathfrak{B})$;
(e) $\mathfrak{A} \models \mathfrak{B} \Rightarrow \mathfrak{A} \models_{\mathfrak{A}} \circ \models_{\mathfrak{B}} \mathfrak{B}$ □

For (c) and (d) the following (traditional) interpretation can be introduced:

(c1) $\mathfrak{A} \models \mathfrak{B} \Rightarrow (\exists\ \mathfrak{C}).((\mathfrak{A} \models \mathfrak{C}) \models \mathfrak{B})$;
(d1) $\mathfrak{A} \models \mathfrak{B} \Rightarrow (\exists\ \mathfrak{C}).(\mathfrak{A} \models (\mathfrak{C} \models \mathfrak{B}))$

In these formulas, the particular informational operator $\exists$ reads "informs the existence of' or 'is informed about the existence by'. For $\exists\ \mathfrak{C}$ is an open formula, it reads as '$\mathfrak{C}$ is informed of its existence by something undetermined yet'.

Traditionally, formulas (c1) and (d1) can be particularized even in more detail as follows:

(c2) $\mathfrak{A} \models \mathfrak{B} \Rightarrow ((\forall\ \mathfrak{A}, \mathfrak{B})\ \exists\ \mathfrak{C}).((\mathfrak{A} \models \mathfrak{C}) \models \mathfrak{B})$;
(d2) $\mathfrak{A} \models \mathfrak{B} \Rightarrow ((\forall\ \mathfrak{A}, \mathfrak{B})\ \exists\ \mathfrak{C}).(\mathfrak{A} \models (\mathfrak{C} \models \mathfrak{B}))$

In these formulas, aggregate (informational set) $\mathfrak{A}, \mathfrak{B}$ is an operand entity and $\forall\ \mathfrak{A}, \mathfrak{B}$ is read 'something informs each $\mathfrak{A}, \mathfrak{B}$'. Further, formula $(\forall\ \mathfrak{A}, \mathfrak{B})\ \exists\ \mathfrak{C}$ is read '$(\forall\ \mathfrak{A}, \mathfrak{B})$ informs the existence of $\mathfrak{C}$', or in short, 'each $\mathfrak{A}, \mathfrak{B}$ informs the existence of $\mathfrak{C}$'. Operator '.' has the meaning 'such that' or 'informs the possibility of'. Thus, $(\forall\ \mathfrak{A}, \mathfrak{B})\ \exists\ \mathfrak{C}$ as an operand entity informs the possibility of $(\mathfrak{A} \models \mathfrak{C}) \models \mathfrak{B}$ or $\mathfrak{A} \models (\mathfrak{C} \models \mathfrak{B})$.

There is possible to understand in which way regular informational operators $\forall$ and $\exists$ differ from the the traditional logical quantifiers 'for all' and 'there exists'. Informational operators $\forall$ and $\exists$ are in principle multiplex operators, in a particular case they can be binary or unary (but still open) operators (not only quantifiers).

(2.4) AXIOM. An operand entity, marked by $\mathfrak{A}$, is divisible in an informational way. This yields,

(f) $\mathfrak{A} \Rightarrow \mathfrak{A}_1, \mathfrak{A}_2, \ldots$ or $\mathfrak{A} \Rightarrow \mathfrak{A}_1; \mathfrak{A}_2; \ldots$

In principle, the divisibility of $\mathfrak{A}$ is not limited or predetermined. Components $\mathfrak{A}_1, \mathfrak{A}_2, \ldots$ of $\mathfrak{A}$ are informational parts (processes, forms) of $\mathfrak{A}$ as an operand entity, however, as operand entities within $\mathfrak{A}$ they can arise as any other operand entity. □

Instead of (f), there is possible to introduce (traditionally)

(f1) $\mathfrak{A}_1, \mathfrak{A}_2, \ldots \subset \mathfrak{A}$ or $\mathfrak{A}_1; \mathfrak{A}_2; \ldots \subset \mathfrak{A}$

and read 'entities $\mathfrak{A}_1, \mathfrak{A}_2, \ldots$ inform to be parts (components, subentities) of entity $\mathfrak{A}$'.

Axiom (f) enables the introduction of a formula system, marked by $\mathfrak{A}$, where $\mathfrak{A}$ can be a component of the system, it represents. This principle leads to the complexion of an informational entity, to its systematization (systemic structure and organization), in which components perform as autonomous, however, informationally connected entities within $\mathfrak{A}$.

18.3. The internal structure (informational cycle) of an operand entity.

The question of internal structure (or organization) of an operand entity $\mathfrak{A}$ concerns the right side of the open formula $\mathfrak{A} \Rightarrow \mathfrak{A} \models \mathfrak{A}$, in which informing of $\mathfrak{A}$ in (over) itself, i.e., $\mathfrak{A} \models \mathfrak{A}$, is open in the sense that $(\mathfrak{A} \models \mathfrak{A}) \models$ and $\models (\mathfrak{A} \models \mathfrak{A})$. What could be the structure of the internal informing of $\mathfrak{A}$ in regard to the informational arising of $\mathfrak{A}$?

The concept we attempt to introduce with the last question is called the informational cycle. Within this cycle, each operand entity informs, as we say, spontaneously and circularly, that is, in a cyclically spontaneous way. The spontaneous has the meaning of a determined, undetermined, and/or unforeseen informational arising of $\mathfrak{A}$, depending on informing of $\mathfrak{A}$ itself and on informing of operands informing $\mathfrak{A}$, from the standpoint, i.e. from the informational circumstances and possibilities of entity $\mathfrak{A}$.

(18.3.1) AXIOM. The informational cycle of an understanding operand entity $\mathfrak{A}$ is constituted by the following entities of $\mathfrak{A}$: informing of understanding $\mathfrak{A}$, marked by $\mathfrak{I}$, $\mathfrak{I}_{\mathfrak{A}}$, or $\mathfrak{I}(\mathfrak{A})$; counter-informing of understanding $\mathfrak{A}$, marked by $\mathfrak{C}$, $\mathfrak{C}_{\mathfrak{A}}$, or $\mathfrak{C}(\mathfrak{A})$; informational embedding or embedding of information into understanding $\mathfrak{A}$, marked by $\mathfrak{E}$, $\mathfrak{E}_{\mathfrak{A}}$, or $\mathfrak{E}(\mathfrak{A})$. Further, informing $\mathfrak{I}$ of $\mathfrak{A}$ produces (generates, informs) understanding $\mathfrak{A}$; counter-informing $\mathfrak{C}$ of $\mathfrak{A}$ produces counter-information γ of understanding (for instance, misunderstanding, disordered understanding, information of counter-understanding); embedding $\mathfrak{E}$ of $\mathfrak{A}$ embeds in an informational way the arisen counter-information γ and to $\mathfrak{A}$ arriving other information β, by producing the embedding information ε of understanding, that is information, needed to understand an informational entity by understanding. Merely by ε, the counter-arisen and the other, arrived information can be embedded into $\mathfrak{A}$ in an understanding way. We distinguish the following four cases of informing of entity $\mathfrak{A}$ within the so-called informational cycle: general, parallel, serial, and parallel-serial informing. □

(18.3.2) AXIOM. A general interpretation of the informational cycle of understanding. Two general cases of informational cycle of an understanding operand entity can be distinguished.

The general sequential (serial, cyclic) case can be expressed by the following, one-cycle formula:

(g1) $((((((\beta \models \mathfrak{A}) \models \mathfrak{I}) \models \mathfrak{C}) \models \gamma) \models \mathfrak{E}) \models \varepsilon) \models \mathfrak{A}$

This formula is cyclic in regard to $\mathfrak{A}$. For this formula it is also characteristic that the informing of the subsequent entity ($\mathfrak{I}$, $\mathfrak{C}$, γ, $\mathfrak{E}$, ε, $\mathfrak{A}$, etc. in the cycle) depends on all preceding entities, that is, that the history

(memory) of the cyclic process is preserved. Formula (g1) models only the so-called main cycle of an informational cycle within which all possible other subcycles can exist. For instance,

(g2) $((\beta \models \mathfrak{A}) \models \mathfrak{Z}) \models (\mathfrak{A}, \mathfrak{Z})$;

(g3) $(((\beta \models \mathfrak{A}) \models \mathfrak{Z}) \models \mathfrak{C}) \models (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C})$;

(g4) $((((\beta \models \mathfrak{A}) \models \mathfrak{Z}) \models \mathfrak{C}) \models \gamma) \models (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma)$;

(g5) $(((((\beta \models \mathfrak{A}) \models \mathfrak{Z}) \models \mathfrak{C}) \models \gamma) \models \mathfrak{E}) \models (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma, \mathfrak{E})$;

(g6) $((((((\beta \models \mathfrak{A}) \models \mathfrak{Z}) \models \mathfrak{C}) \models \gamma) \models \mathfrak{E}) \models \varepsilon) \models (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon)$

In these subcycles other partial cycles can exist, etc.

The general parallel (simultaneous) case of informing $\mathfrak{A}$ by β and by $\mathfrak{A}$ itself can be expressed in the most complex form by formula

(g7) $(\beta, \mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon) \models (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon)$

representing a parallel system of formulas

(g8) $\beta \models \mathfrak{A}, \beta \models \mathfrak{Z}, \beta \models \mathfrak{C}, \beta \models \gamma, \beta \models \mathfrak{E}, \beta \models \varepsilon,$
$\mathfrak{A} \models \mathfrak{A}, \mathfrak{A} \models \mathfrak{Z}, \mathfrak{A} \models \mathfrak{C}, \mathfrak{A} \models \gamma, \mathfrak{A} \models \mathfrak{E}, \mathfrak{A} \models \varepsilon,$
$\mathfrak{Z} \models \mathfrak{A}, \mathfrak{Z} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{Z} \models \gamma, \mathfrak{Z} \models \mathfrak{E}, \mathfrak{Z} \models \varepsilon,$
$\mathfrak{C} \models \mathfrak{A}, \mathfrak{C} \models \mathfrak{Z}, \mathfrak{C} \models \mathfrak{C}, \mathfrak{C} \models \gamma, \mathfrak{C} \models \mathfrak{E}, \mathfrak{C} \models \varepsilon,$
$\gamma \models \mathfrak{A}, \gamma \models \mathfrak{Z}, \gamma \models \mathfrak{C}, \gamma \models \gamma, \gamma \models \mathfrak{E}, \gamma \models \varepsilon,$
$\mathfrak{E} \models \mathfrak{A}, \mathfrak{E} \models \mathfrak{Z}, \mathfrak{E} \models \mathfrak{C}, \mathfrak{E} \models \gamma, \mathfrak{E} \models \mathfrak{E}, \mathfrak{E} \models \varepsilon,$
$\varepsilon \models \mathfrak{A}, \varepsilon \models \mathfrak{Z}, \varepsilon \models \mathfrak{C}, \varepsilon \models \gamma, \varepsilon \models \mathfrak{E}, \varepsilon \models \varepsilon$

However, in this parallel system some combinations of elementary formulas can be understood as particular (parallel) cycles of different length. For instance, some of the so-called $\mathfrak{A}$-cycles of the length 1, 2, ... , 6 are the following:

(g9/1) $(\mathfrak{A} \models \mathfrak{A})$;

(g9/2) $(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{A})$; $(\mathfrak{A} \models \mathfrak{C}, \mathfrak{C} \models \mathfrak{A})$;
$(\mathfrak{A} \models \gamma, \gamma \models \mathfrak{A})$; $(\mathfrak{A} \models \mathfrak{E}, \mathfrak{E} \models \mathfrak{A})$;
$(\mathfrak{A} \models \varepsilon, \varepsilon \models \mathfrak{A})$;

(g9/3) $(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{C} \models \mathfrak{A})$;
$(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{C} \models \mathfrak{A})$;
$(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{C} \models \mathfrak{A})$;
$(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{C} \models \mathfrak{A})$;
$(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{C} \models \mathfrak{A})$;

(g9/4) $(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{C} \models \gamma, \gamma \models \mathfrak{A})$;

(g9/5) $(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{C} \models \gamma, \gamma \models \mathfrak{E}, \mathfrak{E} \models \mathfrak{A})$;

(g9/6) $(\mathfrak{A} \models \mathfrak{Z}, \mathfrak{Z} \models \mathfrak{C}, \mathfrak{C} \models \gamma, \gamma \models \mathfrak{E}, \mathfrak{E} \models \varepsilon, \varepsilon \models \mathfrak{A})$;

Similarly, the so called $\mathfrak{Z}$-, $\mathfrak{C}$-, γ-, $\mathfrak{E}$-, and ε-cycles can be observed. □

(18.3.3) AXIOM. A parallel interpretation of the informational cycle of understanding. In the previous, general case we have observed the parallel and the cyclic nature of informing of understanding $\mathfrak{A}$. To explicate the parallel nature of informing we can introduce a general parallel informational operator of informing, marked by $\Vdash$. Then, we can simply continue the discussion presented in the previous, general case. The parallel sequential (cyclic) case can be expressed by the following, parallel one-cycle formula:

(h1) $((((((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{Z}) \Vdash \mathfrak{C}) \Vdash \gamma) \Vdash \mathfrak{E}) \Vdash \varepsilon) \Vdash \mathfrak{A}$

Though this formula is cyclic in regard to $\mathfrak{A}$, it is inwardly parallel in all its operand components. Thus, for instance, $\beta \Vdash \mathfrak{A}$ means that β informs to $\mathfrak{A}$ in parallel to all $\mathfrak{A}$'s cyclic components, i.e. to $\mathfrak{Z}$, $\mathfrak{C}$, γ, $\mathfrak{E}$, ε, and to $\mathfrak{A}$ as an informational entirety of these entities. We understand how a parallel cycle of the form (h1) can be complexly determined to perform in a parallel way in regard to its operand components.

For formula (h1) it is characteristic that the informing of the entities $\mathfrak{A}$, $\mathfrak{Z}$, $\mathfrak{C}$, γ, $\mathfrak{E}$, ε, $\mathfrak{A}$, etc. in the cycle depends on all preceding entities, that is, that the history (memory) of the parallel cyclic process is preserved. Again, formula (h1) models only the so-called main cycle of an informational cycle within which all possible other subcycles can exist. For instance,

(h2) $((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{Z}) \Vdash (\mathfrak{A}, \mathfrak{Z})$;

(h3) $(((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{Z}) \Vdash \mathfrak{C}) \Vdash (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C})$;

(h4) $((((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{Z}) \Vdash \mathfrak{C}) \Vdash \gamma) \Vdash (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma)$;

(h5) $(((((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{Z}) \Vdash \mathfrak{C}) \Vdash \gamma) \Vdash \mathfrak{E}) \Vdash (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma, \mathfrak{E})$;

(h6) $((((((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{Z}) \Vdash \mathfrak{C}) \Vdash \gamma) \Vdash \mathfrak{E}) \Vdash \varepsilon) \Vdash (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon)$

In these subcycles other partial parallel cycles can exist, etc. The last formulas are cases of extremely possible parallelism within an understanding operand $\mathfrak{A}$. □

(18.3.4) AXIOM. A cyclic interpretation of the informational cycle of understanding. In both general and parallel case we have observed the cyclic nature of informing. To explicate the pure cyclic nature of informing we can introduce a general cyclic informational operator of informing, marked by $\vdash$. The pure cyclic case can be expressed by the following, one-cycle formula:

(i1) $((((((\beta \models \mathfrak{A}) \vdash \mathfrak{Z}) \vdash \mathfrak{C}) \vdash \gamma) \vdash \mathfrak{E}) \vdash \varepsilon) \vdash \mathfrak{A}$

This formula is strictly cyclic in regard to understanding $\mathfrak{A}$, it is inwardly cyclic in regard to all its operand components. In this case, the outward informing entity β informs $\mathfrak{A}$ in a general manner, i.e., in the form $\beta \models \mathfrak{A}$. By definition, within the cycle (i1), other cycles of $\mathfrak{A}$'s components could be permitted. For instance,

(i2) $((\beta \models \mathfrak{A}) \vdash \mathfrak{Z}) \vdash (\mathfrak{A}, \mathfrak{Z})$;

(i3) $(((\beta \models \mathfrak{A}) \vdash \mathfrak{Z}) \vdash \mathfrak{C}) \vdash (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C})$;

(i4) $((((\beta \models \mathfrak{A}) \vdash \mathfrak{Z}) \vdash \mathfrak{C}) \vdash \gamma) \vdash (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma)$;

(i5) $(((((\beta \models \mathfrak{A}) \vdash \mathfrak{Z}) \vdash \mathfrak{C}) \vdash \gamma) \vdash \mathfrak{E}) \vdash (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma, \mathfrak{E})$;

(i6) $((((((\beta \models \mathfrak{A}) \vdash \mathfrak{Z}) \vdash \mathfrak{C}) \vdash \gamma) \vdash \mathfrak{E}) \vdash \varepsilon) \vdash (\mathfrak{A}, \mathfrak{Z}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon)$

The last formulas describe pure sequential cases within an understanding operand $\mathfrak{A}$. □

(18.3.5) AXIOM. A parallel-cyclic interpretation of the informational cycle of understanding. As understood, informing of understanding information is parallel as well as cyclic: in parallel informing, cycles can be formed and in cyclic informing, parallel informing can occur. This leads to the introduction of the parallel-cyclic operator $\Vdash$. The parallel-cyclic case can be expressed by the following, one-cycle formula:

(j1) $((((((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{I}) \Vdash \mathfrak{C}) \vdash \gamma) \Vdash \mathfrak{E}) \Vdash \varepsilon) \Vdash \mathfrak{A}$

This formula is strictly parallel-cyclic in regard to understanding $\mathfrak{A}$, it is inwardly and outwardly $(\beta \Vdash \mathfrak{A})$ parallel-cyclic in regard to all its operand components. By definition, within the parallel cycle (j1) other parallel cycles of $\mathfrak{A}$'s components could be permitted. For instance,

(j2) $((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{I}) \Vdash (\mathfrak{A}, \mathfrak{I})$;

(j3) $(((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{I}) \Vdash \mathfrak{C}) \Vdash (\mathfrak{A}, \mathfrak{I}, \mathfrak{C})$;

(j4) $((((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{I}) \Vdash \mathfrak{C}) \Vdash \gamma) \Vdash (\mathfrak{A}, \mathfrak{I}, \mathfrak{C}, \gamma)$;

(j5) $(((((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{I}) \Vdash \mathfrak{C}) \Vdash \gamma) \Vdash \mathfrak{E}) \Vdash (\mathfrak{A}, \mathfrak{I}, \mathfrak{C}, \gamma, \mathfrak{E})$;

(j6) $((((((\beta \Vdash \mathfrak{A}) \Vdash \mathfrak{I}) \Vdash \mathfrak{C}) \Vdash \gamma) \Vdash \mathfrak{E}) \Vdash \varepsilon) \Vdash (\mathfrak{A}, \mathfrak{I}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon)$

The last formulas describe parallel-sequential cases within an understanding operand $\mathfrak{A}$. □

Operator $\Vdash$ is the most complex case of operator $\models$. It explicitly expresses the parallel-cyclic nature of informing.

The principle of informing $\mathfrak{I}$ of understanding $\mathfrak{A}$, and within it the principles of counter-informing $\mathfrak{C}$ of counter-information γ and embedding $\mathfrak{E}$ of embedding information ε, constitute the basic concept of any operand entity, which by its informing is operative by itself in an informational way. Operators connect (couple) operands and are simultaneously parts of connected (coupled) operands (e.g., operator compositions). Up to now four general informational operators have been introduced: the general, $\models$, the parallel, $\Vdash$, the cyclic, $\vdash$, and the parallel-cyclic operator, $\Vdash$. Thus, we can introduce the following more precise definitions for the usage of operators concerning the outward and the inward informing of operands.

(18.3.6) DEFINITION. The following definitions can be introduced:

(k1) $\beta \models \mathfrak{A} \Leftrightarrow (\beta, \mathfrak{A}, \mathfrak{I}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon) \models (\mathfrak{A}, \mathfrak{I}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon)$;

(k2) $\beta \Vdash \mathfrak{A} \Leftrightarrow (\beta, \mathfrak{A}, \mathfrak{I}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon) \Vdash (\mathfrak{A}, \mathfrak{I}, \mathfrak{C}, \gamma, \mathfrak{E}, \varepsilon)$;

(k3) $\beta \vdash \mathfrak{A} \Leftrightarrow ((((((\beta \models \mathfrak{A}) \vdash \mathfrak{I}) \vdash \mathfrak{C}) \vdash \gamma) \vdash \mathfrak{E}) \vdash \varepsilon) \vdash \mathfrak{A}$;

(k4) $\beta \Vdash \mathfrak{A} \Leftrightarrow (\beta \Vdash \mathfrak{A}) \wedge (\beta \vdash \mathfrak{A})$

The right side of $\Leftrightarrow$ in (k1) is defined by (g8). The right side of $\Leftrightarrow$ in (k2) is defined by (g8), if operators $\models$ are replaced by operators $\Vdash$. The right side of $\Leftrightarrow$ in (k3) demonstrates a sort of pipeline principle in regard to $\mathfrak{A}$, that is, to components of its informational cycle. Parallel-serial informing (k4) is simply a mixture of parallel and cyclic informing. □

Definition 18.3.6 clears the sense of introduction of four informational operators, $\models$, $\Vdash$, $\vdash$, and $\Vdash$. Operator $\wedge$ means 'informs in an "and" fashion in one and another direction' (in regard to the left and the right operand).

Let us examine how an operand does not inform and how does it inform in another (alternative) way.

18.4. Informing and non-informing of understanding

What does in general the non-informing of an operand mean? In general, non-informing is nothing else than informing in a non-informing way. For instance, if things inform (impact other things and itself) in a thing-characteristic way, then things do not or cannot inform in a thing-non-characteristic (non-impacting) way. By non-informing of an operand its property of non-informing can be explicated. We can explicate, for instance, that an operand does not inform true. If operand $\mathfrak{A}$ does not inform in a certain way, we introduce $\mathfrak{A} \not\models$. The next question to this non-informing might be, in which particular way does $\mathfrak{A}$ not inform. The antisymmetric situation is the case of $\not\models \mathfrak{A}$, where $\mathfrak{A}$ is not informed or cannot be informed (impacted) in a particular way.

(4.1) DEFINITION. Operator $\not\models$ can be understood as a particularization of operator $\models$. The following equivalence can be introduced:

(11) $\alpha \not\models \beta \Leftrightarrow (\alpha \not\models_{\alpha} \circ \models_{\beta} \beta) \vee (\alpha \models_{\alpha} \circ \not\models_{\beta} \beta) \vee (\alpha \not\models_{\alpha} \circ \not\models_{\beta} \beta)$

Operator $\vee$ is an informational disjunction and thus,

(12) $(\alpha \not\models_{\alpha} \circ \models_{\beta} \beta)$; $(\alpha \models_{\alpha} \circ \not\models_{\beta} \beta)$; $(\alpha \not\models_{\alpha} \circ \not\models_{\beta} \beta) \Rightarrow \alpha \not\models \beta$

This is an informational formula where each left multiplex operand of operator $\Rightarrow$ implies $\alpha \not\models \beta$. □

18.5. Alternative informing of understanding

To make the discussion and construction of alternative cases of informing of understanding possible on the formal and the most general level, the so-called alternative operators can be introduced. It means that to each operator there exists an alternative operator with the following meaning: if the original operator, as we say, informs in one way, then the alternative operator, as we say, informs in another way. This approach enables to make the previous general definitions and axioms even more general.

(18.5.1) DEFINITION. The operator pairs of one and another case of informing are the following:

($\models$, $\dashv\!\models$) and ($\not\models$, $\not\dashv\!\models$) for general informing and general non-informing in one and another way, respectively;

($\Vdash$, $\dashv\!\Vdash$) and ($\not\Vdash$, $\not\dashv\!\Vdash$) for parallel informing and parallel non-informing in one and another way, respectively;

($\vdash$, $\dashv$) and ($\not\vdash$, $\not\dashv$) for cyclic informing and cyclic non-informing in one and another way, respectively; and

($\Vdash$, $\dashv\!\Vdash$) and ($\not\Vdash$, $\not\dashv\!\Vdash$) for parallel-cyclic informing and parallel-cyclic non-informing in one and another way respectively.

According to the type of understanding these operators can be appropriately particularized (indexed), for instance, $\models_{\mathfrak{A}}$, $\dashv\!\models_{\mathfrak{A}}$, $\not\models_{\mathfrak{A}}$, $\not\dashv\!\models_{\mathfrak{A}}$, $\Vdash_{\mathfrak{A}}$, $\dashv\!\Vdash_{\mathfrak{A}}$, $\not\Vdash_{\mathfrak{A}}$, $\not\dashv\!\Vdash_{\mathfrak{A}}$, $\vdash_{\mathfrak{A}}$, $\dashv_{\mathfrak{A}}$, $\not\vdash_{\mathfrak{A}}$, $\not\dashv_{\mathfrak{A}}$, $\Vdash_{\mathfrak{A}}$, $\dashv\!\Vdash_{\mathfrak{A}}$, $\not\Vdash_{\mathfrak{A}}$, $\not\dashv\!\Vdash_{\mathfrak{A}}$, where understanding $\mathfrak{A}$ appears as a general parameter of understanding (for instance, hermeneutic interpretation, semantic explanation, particular comprehension, informational understanding, etc.) □

(18.5.2) AXIOM. Axiom 18.2.1 can now be broadened and generalized by considering Definition 18.5.1 in the following form:

(m1) $\mathfrak{A} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A})$; $\mathfrak{A} \Rightarrow (\mathfrak{A} \not\models; \not\models \mathfrak{A})$;
$\mathfrak{A} \Rightarrow (\dashv\!\models \mathfrak{A}; \mathfrak{A} \dashv\!\models)$; $\mathfrak{A} \Rightarrow (\not\dashv\!\models \mathfrak{A}; \mathfrak{A} \not\dashv\!\models)$;

(m2) $\mathfrak{A} \Rightarrow (\mathfrak{A} \Vdash; \Vdash \mathfrak{A})$; $\mathfrak{A} \Rightarrow (\mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A})$;
$\mathfrak{A} \Rightarrow (\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash)$; $\mathfrak{A} \Rightarrow (\not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$;

(m3) $\mathfrak{A} \Rightarrow (\mathfrak{A} \vdash; \vdash \mathfrak{A})$; $\mathfrak{A} \Rightarrow (\mathfrak{A} \not\vdash; \not\vdash \mathfrak{A})$;
$\mathfrak{A} \Rightarrow (\dashv \mathfrak{A}; \mathfrak{A} \dashv)$; $\mathfrak{A} \Rightarrow (\not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv)$;

(m4) $\mathfrak{A} \Rightarrow (\mathfrak{A} \Vdash; \Vdash \mathfrak{A})$; $\mathfrak{A} \Rightarrow (\mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A})$;
$\mathfrak{A} \Rightarrow (\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash)$; $\mathfrak{A} \Rightarrow (\not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$;

(m5) $\mathfrak{A} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A}; \dashv\!\models \mathfrak{A}; \mathfrak{A} \dashv\!\models)$;
$\mathfrak{A} \Rightarrow (\mathfrak{A} \not\models; \not\models \mathfrak{A}; \not\dashv\!\models \mathfrak{A}; \mathfrak{A} \not\dashv\!\models)$;

(m6) $\mathfrak{A} \Rightarrow (\mathfrak{A} \Vdash; \Vdash \mathfrak{A}; \dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash)$;
$\mathfrak{A} \Rightarrow (\mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A}; \not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$;

(m7) $\mathfrak{A} \Rightarrow (\mathfrak{A} \vdash; \vdash \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv)$;
$\mathfrak{A} \Rightarrow (\mathfrak{A} \not\vdash; \not\vdash \mathfrak{A}; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv)$;

(m8) $\mathfrak{A} \Rightarrow (\mathfrak{A} \Vdash; \Vdash \mathfrak{A}; \dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash)$;
$\mathfrak{A} \Rightarrow (\mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A}; \not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$;

(m9) $\mathfrak{A} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \not\models; \not\models \mathfrak{A})$;
$\mathfrak{A} \Rightarrow (\dashv\!\models \mathfrak{A}; \mathfrak{A} \dashv\!\models; \not\dashv\!\models \mathfrak{A}; \mathfrak{A} \not\dashv\!\models)$;

(m10) $\mathfrak{A} \Rightarrow (\mathfrak{A} \Vdash; \Vdash \mathfrak{A}; \mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A})$;
$\mathfrak{A} \Rightarrow (\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash; \not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$;

(m11) $\mathfrak{A} \Rightarrow (\mathfrak{A} \vdash; \vdash \mathfrak{A}; \mathfrak{A} \not\vdash; \not\vdash \mathfrak{A})$;
$\mathfrak{A} \Rightarrow (\dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv)$;

(m12) $\mathfrak{A} \Rightarrow (\mathfrak{A} \Vdash; \Vdash \mathfrak{A}; \mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A})$;
$\mathfrak{A} \Rightarrow (\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash; \not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$;

(m13) $\mathfrak{A} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \not\models; \not\models \mathfrak{A};$
$\dashv\!\models \mathfrak{A}; \mathfrak{A} \dashv\!\models; \not\dashv\!\models \mathfrak{A}; \mathfrak{A} \not\dashv\!\models)$;

(m14) $\mathfrak{A} \Rightarrow (\mathfrak{A} \Vdash; \Vdash \mathfrak{A}; \mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A};$
$\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash; \not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$;

(m15) $\mathfrak{A} \Rightarrow (\mathfrak{A} \vdash; \vdash \mathfrak{A}; \mathfrak{A} \not\vdash; \not\vdash \mathfrak{A};$
$\dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv)$;

(m16) $\mathfrak{A} \Rightarrow (\mathfrak{A} \Vdash; \Vdash \mathfrak{A}; \mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A};$
$\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash; \not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$;

(m17) $\mathfrak{A} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \not\models; \not\models \mathfrak{A};$
$\dashv\!\models \mathfrak{A}; \mathfrak{A} \dashv\!\models; \not\dashv\!\models \mathfrak{A}; \mathfrak{A} \not\dashv\!\models;$
$\mathfrak{A} \Vdash; \Vdash \mathfrak{A}; \mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A};$
$\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash; \not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash;$
$\mathfrak{A} \vdash; \vdash \mathfrak{A}; \mathfrak{A} \not\vdash; \not\vdash \mathfrak{A};$
$\dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv;$
$\mathfrak{A} \Vdash; \Vdash \mathfrak{A}; \mathfrak{A} \not\Vdash; \not\Vdash \mathfrak{A};$
$\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \dashv\!\Vdash; \not\dashv\!\Vdash \mathfrak{A}; \mathfrak{A} \not\dashv\!\Vdash)$

Operand $\mathfrak{A}$ is in principle a completely open system, where operators $\models$, $\dashv\!\models$, $\not\models$, $\not\dashv\!\models$, $\Vdash$, $\dashv\!\Vdash$, $\not\Vdash$, $\not\dashv\!\Vdash$, $\vdash$, $\dashv$, $\not\vdash$, $\not\dashv$, $\Vdash$, $\dashv\!\Vdash$, $\not\Vdash$, and $\not\dashv\!\Vdash$ symbolize the acts of various natures of informing and non-informing, that is, when entity $\mathfrak{A}$ informs and non-informs, i.e., $\mathfrak{A} \models$, $\dashv\!\models \mathfrak{A}$, $\mathfrak{A} \not\models$, $\not\dashv\!\models \mathfrak{A}$, $\mathfrak{A} \Vdash$, $\dashv\!\Vdash \mathfrak{A}$, $\mathfrak{A} \not\Vdash$, $\not\dashv\!\Vdash \mathfrak{A}$, $\mathfrak{A} \vdash$, $\dashv \mathfrak{A}$, $\mathfrak{A} \not\vdash$, $\not\dashv \mathfrak{A}$, $\mathfrak{A} \Vdash$, $\dashv\!\Vdash \mathfrak{A}$, $\mathfrak{A} \not\Vdash$, and $\not\dashv\!\Vdash \mathfrak{A}$ to yet unidentified entities and is informed and non-informed, i.e., $\models \mathfrak{A}$, $\mathfrak{A} \dashv\!\models$, $\not\models \mathfrak{A}$, $\mathfrak{A} \not\dashv\!\models$, $\Vdash \mathfrak{A}$, $\mathfrak{A} \dashv\!\Vdash$, $\not\Vdash \mathfrak{A}$, $\mathfrak{A} \not\dashv\!\Vdash$, $\vdash \mathfrak{A}$, $\mathfrak{A} \dashv$, $\not\vdash \mathfrak{A}$, $\mathfrak{A} \not\dashv$, $\Vdash \mathfrak{A}$, $\mathfrak{A} \dashv\!\Vdash$, $\not\Vdash \mathfrak{A}$, and $\mathfrak{A} \not\dashv\!\Vdash$ by yet unidentified entities (operands), in one and another way, respectively. In system (m1)-(m16), components of the form $\mathfrak{A} \models$ and $\models \mathfrak{A}$ ($\models$ is for $\models$, $\dashv\!\models$, $\not\models$, $\not\dashv\!\models$, $\Vdash$, $\dashv\!\Vdash$, $\not\Vdash$, $\not\dashv\!\Vdash$, $\vdash$, $\dashv$, $\not\vdash$, $\not\dashv$, $\Vdash$, $\dashv\!\Vdash$, $\not\Vdash$, and $\not\dashv\!\Vdash$) perform as distinguished entities which can again inform and can be informed in the sense of (a1) in Axiom 3.1. Inductively, the system of rank n of $\mathfrak{A}$'s informing and non-informing, in one and another way, can be constructed. This kind of recursion makes an operand entity completely open in transmitting (informing) information to and receiving (informing) information from yet unidentified operand entities by various kinds of informing and non-informing. □

It is evident how an understanding operand $\mathfrak{A}$ can inform and can be informed generally, parallel, cyclically, parallel-cyclically, and alternatively. All these possibilities can be expressed explicitly by the use of the listed operators. Thus, informational operands of understanding can become as complex as possible by the recursive decomposition of formulas, i.e., of operands as well as operators.

18.6. Informational theories of understanding

Up to now we have presented a soft or general concept of the axiomatic base, giving the idea how in concrete theories these bases could be constructed according to the conceptualized needs. Theorems and their proofs will be informational formulas, where proofs are ways or scenarios representing the modes of getting theorems. A general informational theory of understanding can be developed ad infinitum, adding new and new theorems to the existing ones. Let us look at some examples.

(18.6.1) THEOREM. Some very basic theorems are:

(n1) Axiom (18.2.1) (a), that is,
$\mathfrak{A} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A})$
yields also the opposite formula, that is,
$(\mathfrak{A} \models; \models \mathfrak{A}) \Rightarrow \mathfrak{A}$
In this particular case, the meaning of an informational operand is
$\mathfrak{A} \Leftrightarrow (\mathfrak{A} \models; \models \mathfrak{A})$;

(n2) $\mathfrak{A} \models \Rightarrow \models \mathfrak{A}$; $\models \mathfrak{A} \Rightarrow \mathfrak{A} \models$; $\mathfrak{A} \models \not\equiv \models \mathfrak{A}$;

(n3) $\mathfrak{A} \dashv\!\models \Leftarrow \dashv\!\models \mathfrak{A}$; $\dashv\!\models \mathfrak{A} \Leftarrow \mathfrak{A} \dashv\!\models$; $\mathfrak{A} \dashv\!\models \not\equiv \dashv\!\models \mathfrak{A}$;

(n4) $\mathfrak{A} \models \Rightarrow \dashv\!\models \mathfrak{A}$; $\mathfrak{A} \models \Leftarrow \dashv\!\models \mathfrak{A}$; $\mathfrak{A} \models \not\equiv \dashv\!\models \mathfrak{A}$;

(n5) $\mathfrak{A} \models \Rightarrow \mathfrak{A} \dashv\!\models$; $\mathfrak{A} \models \Leftarrow \mathfrak{A} \dashv\!\models$; $\mathfrak{A} \models \not\equiv \mathfrak{A} \dashv\!\models$ □

PROOF. Proofs of the listed theorems are the following:

(n1') Operand $\mathfrak{A}$ informs and is informed and, if so, the informing of $\mathfrak{A}$ in the sense to inform and to be informed can be the sufficient condition to say that $\mathfrak{A}$ is informational operand. Thus, $(\mathfrak{A} \models; \models \mathfrak{A}) \Rightarrow \mathfrak{A}$. In this

particular case, the consequence of the original and opposite implication is $\mathfrak{A} \Leftrightarrow (\mathfrak{A} \models; \models \mathfrak{A})$.

(n2') If $\mathfrak{A}$ informs, then it is at least informed by itself, that is, $\mathfrak{A} \models \Rightarrow (\mathfrak{A} \models \mathfrak{A})$. Thus, $\models \mathfrak{A}$. If $\mathfrak{A}$ is informed, then it informs also itself, that is, $\models \mathfrak{A} \Rightarrow (\mathfrak{A} \models \mathfrak{A})$. Thus, $\models \mathfrak{A}$. To inform and to be informed can be completely different (particularized) operations. Thus, $(\mathfrak{A} \models) \not\equiv (\models \mathfrak{A})$. This theorem concerns the so-called informing in one way (operator $\models$ or its particularizations, respectively).

(n3') Theorem (n3) is the antisymmetric (alternative) case of theorem (n2) which concerns the so-called informing in another way (operator $\dashv$ or its particularizations, respectively). Operator $\Leftarrow$ has the meaning of antisymmetric (alternative) implication.

(n4') According to Axiom (18.5.2), (m5) or (m13), when operand $\mathfrak{A}$ is informing in one and/or another way, $\mathfrak{A} \models$ implies in an informational way $\dashv \mathfrak{A}$ and vice versa, however, $\mathfrak{A} \models$ and $\dashv \mathfrak{A}$ are not identical.

(n5') Theorem (n5) can be understood to be the implicatively alternative case of theorem (n4).

Certainly, theorems in the sense of Theorem 18.6.1 can be continued infinitely because of the principle of operator particularization and universalization. Q.E.D.

(18.6.2) THEOREM. According to Axiom 18.5.2, (m17),

$$
\begin{aligned}
(m18)(&\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \not\models; \not\models \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv;\\
&\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \models; \models \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv;\\
&\mathfrak{A} \vdash; \vdash \mathfrak{A}; \mathfrak{A} \nvdash; \nvdash \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv;\\
&\mathfrak{A} \vdash; \vdash \mathfrak{A}; \mathfrak{A} \nvdash; \nvdash \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv) \Rightarrow\\
(&\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \not\models; \not\models \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv;\\
&\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \models; \models \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv;\\
&\mathfrak{A} \vdash; \vdash \mathfrak{A}; \mathfrak{A} \nvdash; \nvdash \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv;\\
&\mathfrak{A} \vdash; \vdash \mathfrak{A}; \mathfrak{A} \nvdash; \nvdash \mathfrak{A}; \dashv \mathfrak{A}; \mathfrak{A} \dashv; \not\dashv \mathfrak{A}; \mathfrak{A} \not\dashv)
\end{aligned}
$$

where the occurring operators can be arbitrarily particularized and universalized. Operator of informational implication in one way, that is, $\Rightarrow$, can be replaced by operator of informational implication in another way, that is, $\Leftarrow$, to distinguish alternative implicative possibilities. □

PROOF. Theorem 18.6.2 is a generalization of the previous theorem. Axiomatically, any occurrence of the form 'to inform' or 'to be informed' in one or another way implies 'to inform' or 'to be informed' in one or another way. Further, the occurring operators can still be particularized and (again) universalized. Thus, for instance, $\mathfrak{A} \models \Rightarrow \mathfrak{A} \models$ is not a trivial implication, because, for instance, the case $\mathfrak{A} \models_\beta \Rightarrow \mathfrak{A} \models_\gamma$ can be reasonable in an informational way. The property of cross-implicative relations is the consequence of Axiom 18.5.2, (m17) and the previous definitions and axioms, determining the informational nature of an operand $\mathfrak{A}$. Theorem 18.6.2 says that informing and non-informing of $\mathfrak{A}$ is not limited in advance in any way and that all possible, unforeseeable forms of informing can occur. Q.E.D.

Why there is, for instance, nothing surprising in the informational implication of the form $\mathfrak{A} \models \Rightarrow \mathfrak{A} \not\models$, where $\models$ and $\not\models$ seem to oppose each other in this particular expression? Let us remind that $\models$ and $\not\models$ are metaoperators and that if $\mathfrak{A}$ informs in one way, it may not inform in another way. Thus, the possibility of $\mathfrak{A} \models \Rightarrow \mathfrak{A} \not\models$ always exists. In the opposite case, $\mathfrak{A}$ would be an almighty, omnipotent, timeless, and everywhere present information (for instance, God). It would simply mean that within $\mathfrak{A}$ all possible and impossible information is arising or that $\mathfrak{A}$ incorporates the arising and understanding of the entire informational cosmos.

Why the trivial case of $\mathfrak{A} \models \Rightarrow \mathfrak{A} \not\models$, in the sense of the classical mathematical logic, is not possible, for instance, when saying that $\models$ and $\not\models$ is one and same nature of an operator, where $\not\models$ is an ideal negation of $\models$ and vice versa? What is an ideal negation in the informational sense? Does it exist at all when considering all existing and possible and unforeseeable information? In informational sense, $\models$ and $\not\models$ are simply different operators, even not cognate in any sense in regard to informing and non-informing of an operand of understanding.

(18.6.3) THEOREM. Let us examine some formulas using binary and multiplex operators in the context (sense) of formulas $\xi \models \eta$ and $\eta \dashv \xi$:

$$
\begin{aligned}
(o1)\quad & \mathfrak{A} \models \mathfrak{B} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \models \mathfrak{A}; \mathfrak{A} \models \mathfrak{B};\\
&\qquad \mathfrak{B} \models; \models \mathfrak{B}; \mathfrak{B} \models \mathfrak{B};\\
&\qquad (\mathfrak{A} \models \mathfrak{B}) \models; \models (\mathfrak{A} \models \mathfrak{B});\\
&\qquad (\mathfrak{A} \models \mathfrak{B}) \models (\mathfrak{A} \models \mathfrak{B}))\\
(o2)\quad & ((\mathfrak{B} \dashv \mathfrak{A}) \dashv (\mathfrak{B} \dashv \mathfrak{A}); (\mathfrak{B} \dashv \mathfrak{A}) \dashv;\\
&\qquad \dashv (\mathfrak{B} \dashv \mathfrak{A});\\
&\qquad \mathfrak{B} \dashv \mathfrak{B}; \mathfrak{B} \dashv; \dashv \mathfrak{B};\\
&\qquad \mathfrak{B} \dashv \mathfrak{A}; \mathfrak{A} \dashv; \dashv \mathfrak{A}) \Leftarrow \mathfrak{B} \dashv \mathfrak{A}\\
(o3)\quad & (\mathfrak{A}, \mathfrak{B} \models \mathfrak{A}, \mathfrak{B}) \Rightarrow\\
& \mathfrak{B} \models \mathfrak{A}; \mathfrak{A} \dashv \mathfrak{B} \Leftarrow (\mathfrak{B}, \mathfrak{A} \dashv \mathfrak{B}, \mathfrak{A})\\
(o4)\quad & \mathfrak{A} \models \mathfrak{B} \Rightarrow (\mathfrak{A} \models \mathfrak{A}, =); (\mathfrak{B}, \mathfrak{A} \dashv \mathfrak{A}) \Leftarrow \mathfrak{B} \dashv \mathfrak{A}\\
(o5)\quad & \mathfrak{A} \models \mathfrak{B} \Rightarrow (\mathfrak{A}, \mathfrak{B} \models \mathfrak{B}); (\mathfrak{B} \dashv \mathfrak{B}, \mathfrak{A}) \Leftarrow \mathfrak{B} \dashv \mathfrak{A}\\
(o6)\quad & \mathfrak{A} \models \mathfrak{B} \Rightarrow (\ldots((\mathfrak{A} \models \mathfrak{B}) \models \mathfrak{B})\ldots \models \mathfrak{B});\\
& (\mathfrak{B} \dashv \ldots(\mathfrak{B} \dashv (\mathfrak{B} \dashv \mathfrak{A}))\ldots) \Leftarrow \mathfrak{B} \dashv \mathfrak{A}\\
(o7)\quad & (\mathfrak{A} \models \mathfrak{B}) \not\models_{\Rightarrow} (\mathfrak{B} \models \mathfrak{A}); (\mathfrak{A} \dashv \mathfrak{B}) \not\dashv_{\Leftarrow} (\mathfrak{B} \dashv \mathfrak{A})
\end{aligned}
$$

PROOF. Proofs of the listed theorems are the following:

(o1') In $\mathfrak{A} \models \mathfrak{B}$, operands $\mathfrak{A}$ and $\mathfrak{B}$ are autonomous entities; thus, $\mathfrak{A}$ implies $\mathfrak{A} \models; \models \mathfrak{A}; \mathfrak{A} \models \mathfrak{A}$ and $\mathfrak{B}$ implies $\mathfrak{B} \models; \models \mathfrak{B}; \mathfrak{B} \models \mathfrak{B}$. A similar implication exists for $\mathfrak{A} \models \mathfrak{B}$ as an operand unity. Besides, $\mathfrak{A} \models \mathfrak{B}$ in the narrower sense, where $\mathfrak{A}$ informs (for instance, understands understanding) $\mathfrak{B}$, on the right side of $\Rightarrow$, is evident. So, $\mathfrak{A} \models \mathfrak{B}$ on the left side of $\Rightarrow$ has a broader, informational meaning, resulting from the previous axiomatic basis.

(o2') This is a pure antisymmetric case of (o1) with the aim to remind us on the alternative nature of informational implication in this particular case.

(o3') Since formula $(\mathfrak{A}, \mathfrak{B} \models \mathfrak{A}, \mathfrak{B})$ represents a system of four formulas, i.e., $\mathfrak{A} \models \mathfrak{A}; \mathfrak{A} \models \mathfrak{B}; \mathfrak{B} \models \mathfrak{A}; \mathfrak{B} \models \mathfrak{B}$, the first part of theorem holds implicatively in one way. The second part of theorem holds similarly for informing in another way.

(o4') There can be seen from previous definitions that formula $\mathfrak{A} \models \mathfrak{B}$ implies in one way $\mathfrak{A} \models \mathfrak{A}$; $\mathfrak{A} \models \mathfrak{B}$. In this case, $\mathfrak{A} \models \mathfrak{A}$ is implied in one way by $\mathfrak{A}$ itself. Similarly, formula $\mathfrak{B} \dashv \mathfrak{A}$ implies in another way the formula aggregate $\mathfrak{B} \dashv \mathfrak{A}$; $\mathfrak{A} \dashv \mathfrak{A}$. In this case, $\mathfrak{A} \dashv \mathfrak{A}$ is implied in another way by $\mathfrak{A}$ itself.

(o5') This theorem concerns $\mathfrak{B}$ in a similar manner as theorem (o4) concerns $\mathfrak{A}$.

(o6') In formula $\mathfrak{A} \models \mathfrak{B}$, operand $\mathfrak{A}$ performs as a thing which informs in a thing-characteristic way, operand $\mathfrak{B}$ is in fact the observer of $\mathfrak{A}$'s informing in a $\mathfrak{B}$'s characteristic way, and operator $\models$ is a mode of informational (operational) equipment (composition $\models_{\mathfrak{A}} \circ \models_{\mathfrak{B}}$) between $\mathfrak{A}$ and $\mathfrak{B}$. Since $\mathfrak{B}$ can observe its own informing, it can observe the process $\mathfrak{A} \models \mathfrak{B}$, thus, $(\mathfrak{A} \models \mathfrak{B}) \models \mathfrak{B}$, etc. On the other hand, $\mathfrak{A}$ is not necessarily aware of $\mathfrak{B}$'s observing of operand $\mathfrak{A}$, because $\mathfrak{A} \models$ is a completely open informing of $\mathfrak{A}$ in regard to possible observers. Therefore,

(o8) $(\mathfrak{A} \models \mathfrak{B}) \not\models_{\Rightarrow} (\ldots((\mathfrak{A} \models \mathfrak{B}) \models \mathfrak{A})\ldots \models \mathfrak{A})$;
$(\mathfrak{A} \dashv \ldots(\mathfrak{A} \dashv (\mathfrak{B} \dashv \mathfrak{A}))\ldots) \not\dashv_{\Leftarrow} (\mathfrak{B} \dashv \mathfrak{A})$

is regular, if there is not an explicit informing on the sense of formula $\mathfrak{B} \models \mathfrak{A}$. The similar can be said for the case of informing in another way (operator $\dashv$). The proof of (o6) is an evident deduction on the basis of the previous definitions and theorems.

In the discussed meaning of formula $\mathfrak{A} \models \mathfrak{B}$ a comment has to be added in regard to the meaning of operator $\models$. The role of the observer is in fact on the $\mathfrak{A}$-s side, if $\models$ is read as 'understands'. In this case $\mathfrak{A}$ is observing $\mathfrak{B}$ in an understanding way, that is, $\mathfrak{A}$ understands $\mathfrak{B}$.

(o7') Operators $\not\models_{\Rightarrow}$ and $\not\dashv_{\Leftarrow}$ are examples of particularized operators of non-informing with the meaning does not inform implicatively in one or another way, respectively. Thus, formula $\mathfrak{A} \models \mathfrak{B}$ does not imply $\mathfrak{B} \models \mathfrak{A}$ in the one way of informing and formula $\mathfrak{B} \dashv \mathfrak{A}$ does not imply $\mathfrak{A} \dashv \mathfrak{B}$ in the other way of informing. Evidently, this can be deduced from the previous theorems. Instead of operator particularizations also operator compositions for $\not\models_{\Rightarrow}$ and $\not\dashv_{\Leftarrow}$ could be introduced, for instance, $\not\models \circ \Rightarrow$ and $\Leftarrow \circ \not\dashv$, respectively. As we see, operator particularizations can always be understood as operator compositions and vice versa. In case with a composite operator, for instance, $\xi \models_1 \circ \models_2 \eta$, the influence of operands ξ and η can be considered, for instance, in the form $\xi\ (\models_{1,\xi} \circ \models_{1,\eta}) \circ (\models_{2,\xi} \circ \models_{2,\eta})\ \eta$, where operator between operands ξ and η is a complex composition. Another possibility of such operator composition would be $(\models_{1,\xi} \circ \models_{2,\xi}) \circ (\models_{1,\eta} \circ \models_{2,\eta})$, and all other possible combinations, for instance. Q.E.D.

We see how the discussed principles of particularization and universalization, of operand and operator substitution, of generic (also recursive) properties of operands and operators, for instance,

$$\mathfrak{A} \Rightarrow (\ldots((\mathfrak{A} \models \mathfrak{A}) \models \mathfrak{A})\ldots \models \mathfrak{A})$$

constitute the realm of the understanding in an informational way. The principled (and advanced) generic recursiveness of informational formulas enables the conceptualization of operands (i.e. formulas) marking, for instance, discourse, time, intelligence, understanding, etc. as information in the "redefined" or "rethought" sense of information. Thus, the question arises: Where are the possible advances of this conceptualization (and universalization) in the sense of mathematical disciplinarity? Would it be, for instance, permissive to say, that a mathematical function (formula) is intelligent, if it is capable to observe (inform, counter-inform, and embed) itself to some extent?

(1∂.6.4) THEOREM. Some basic (systemized) theorems concerning informing and non-informing of understanding in one and another way are the following:

(p1) $(\mathfrak{A} \models \mathfrak{B};\ \mathfrak{B} \dashv \mathfrak{A}) \Rightarrow$
$(\mathfrak{A} \models \mathfrak{B};\ \mathfrak{B} \dashv \mathfrak{A};\ \mathfrak{A} \models\!\models \mathfrak{B};\ \mathfrak{B} \dashv\!\dashv \mathfrak{A};$
$\mathfrak{A} \vdash \mathfrak{B};\ \mathfrak{B} \dashv \mathfrak{A};\ \mathfrak{A} \models \mathfrak{B};\ \mathfrak{B} \dashv\!\dashv \mathfrak{A})$

(p2) $(\mathfrak{A} \not\models \mathfrak{B};\ \mathfrak{B} \not\dashv \mathfrak{A}) \Rightarrow$
$(\mathfrak{A} \not\models \mathfrak{B};\ \mathfrak{B} \not\dashv \mathfrak{A};\ \mathfrak{A} \not\models\!\models \mathfrak{B};\ \mathfrak{B} \not\dashv\!\dashv \mathfrak{A};$
$\mathfrak{A} \not\vdash \mathfrak{B};\ \mathfrak{B} \not\dashv \mathfrak{A};\ \mathfrak{A} \not\models \mathfrak{B};\ \mathfrak{B} \not\dashv\!\dashv \mathfrak{A})$

(p3) $(\mathfrak{A} \models \mathfrak{B};\ \mathfrak{B} \dashv \mathfrak{A}) \Rightarrow (\mathfrak{A} \not\models \mathfrak{B};\ \mathfrak{B} \not\dashv \mathfrak{A})$

(p4) $(\mathfrak{A} \not\models \mathfrak{B};\ \mathfrak{B} \not\dashv \mathfrak{A}) \Rightarrow (\mathfrak{A} \models \mathfrak{B};\ \mathfrak{B} \dashv \mathfrak{A})$ □

PROOF. These theorems are only semantically grouped previous theorems with the aim to explicate some common and opposite properties of informing and non-informing of understanding $\mathfrak{A}$, in regard to another understanding information $\mathfrak{B}$ (that is autonomous or componential in regard to $\mathfrak{A}$), in one and another way.

(p1') General informing in one or another way implies general, parallel, cyclic, and parallel-cyclic informing of understanding in one and another way.

(p2') General non-informing of understanding in one or another way implies general, parallel, cyclic, and parallel-cyclic non-informing of understanding in one and another way.

(p3') General informing of understanding in one or another way implies general, parallel, cyclic, and parallel-cyclic non-informing of understanding in one and another way.

(p4') General non-informing of understanding in one or another way implies general, parallel, cyclic, and parallel-cyclic informing of understanding in one and another way.

However,

$$\neg((\mathfrak{A} \models \mathfrak{B};\ \mathfrak{B} \dashv \mathfrak{A}) \Leftrightarrow (\mathfrak{A} \not\models \mathfrak{B};\ \mathfrak{B} \not\dashv \mathfrak{A}));$$

there does not exist (symbol $\neg$ marks the operator of informational negation) an equivalent informational meaning between informing and non-informing of understanding in one and another way. Q.E.D.

The phenomenon of informational recursion was already shown. An understanding informational operand $\mathfrak{A}$ informs and is informed. Its informing, $\mathfrak{A} \models$ and $\models \mathfrak{A}$, informs too and is informed too, that is, $(\mathfrak{A} \models) \models$, $\models (\mathfrak{A} \models)$, etc., ad infinitum. Similar can be said for the case $\mathfrak{A} \models \mathfrak{B}$, etc. Thus, informational recursion of understanding can be resumed by the following theorem:

(18.6.5) THEOREM. A recursive series of recursive theorems of understanding is, for instance, the following:

(q1) $\mathfrak{A} \Rightarrow (\mathfrak{A} \models; \models \mathfrak{A})$;
(q2) $\mathfrak{A} \models \Rightarrow ((\mathfrak{A} \models) \models; \models (\mathfrak{A} \models))$;
(q3) $\models \mathfrak{A} \Rightarrow ((\models \mathfrak{A}) \models; \models (\models \mathfrak{A}))$; ...
(q4) $\mathfrak{A} \Rightarrow (\mathfrak{A} \not\models; \not\models \mathfrak{A})$;
(q5) $\mathfrak{A} \not\models \Rightarrow ((\mathfrak{A} \not\models) \not\models; \not\models (\mathfrak{A} \not\models))$;
(q6) $\not\models \mathfrak{A} \Rightarrow ((\not\models \mathfrak{A}) \not\models; \not\models (\not\models \mathfrak{A}))$; ...
(q7) $(\mathfrak{A} \dashv; \dashv \mathfrak{A}) \Leftarrow \mathfrak{A}$;
(q8) $((\mathfrak{A} \dashv) \dashv; \dashv (\mathfrak{A} \dashv)) \Leftarrow \mathfrak{A} \dashv$;
(q9) $((\dashv \mathfrak{A}) \dashv; \dashv (\dashv \mathfrak{A})) \Leftarrow \dashv \mathfrak{A}$; ...
(q10) $(\mathfrak{A} \not\dashv; \not\dashv \mathfrak{A}) \Leftarrow \mathfrak{A}$;
(q11) $((\mathfrak{A} \not\dashv) \not\dashv; \not\dashv (\mathfrak{A} \not\dashv)) \Leftarrow \mathfrak{A} \not\dashv$;
(q12) $((\not\dashv \mathfrak{A}) \not\dashv; \not\dashv (\not\dashv \mathfrak{A})) \Leftarrow \not\dashv \mathfrak{A}$; ...

Symbol '...' marks the possibility of theorem generation ad infinitum. Operators $\models$, $\not\models$, $\dashv$, and $\not\dashv$ can be replaced by their parallel ($\Vdash$, $\not\Vdash$, $\dashv\!\!\dashv$, $\not\dashv\!\!\dashv$), cyclic ($\vdash$, $\not\vdash$, $\dashv$, $\not\dashv$), and parallel-cyclic counterparts ($\Vdash$, $\not\Vdash$, $\dashv\!\!\dashv$, $\not\dashv\!\!\dashv$), respectively. Mixed cases of showed recursion among operator counterparts are possible. □

PROOF. The proof of these theorems is evident, for it proceeds from the previous definitions. For instance, we see how (q3) can be continued ad infinitum:

$(\models \mathfrak{A}) \models \Rightarrow (((\models \mathfrak{A}) \models) \models; \models ((\models \mathfrak{A}) \models))$;
$(((\models \mathfrak{A}) \models) \Rightarrow ((((\models \mathfrak{A}) \models) \models) \models; \models (((\models \mathfrak{A}) \models) \models))$

etc. Q.E.D.

18.7. Informational algebras of understanding

Already by the principle of particularization and universalization, different informational algebras of understanding can be generated from existing mathematical algebras. Thereupon, these algebras can be developed in a further manner by using other informational principles (POI), that is, axioms and theorems of informational logic, informational algebra (IIA), informational theory (redetermined theory of information), etc.

Let us show a set of rules (axioms) for deduction of propositions as it appears within the classical mathematical logic. The rules of propositional language (concerning understanding as information) are, for instance, the following:

(r1) $\mathfrak{A} \Rightarrow (\mathfrak{B} \Rightarrow \mathfrak{A})$;
(r2) $(\mathfrak{A} \Rightarrow (\mathfrak{A} \Rightarrow \mathfrak{B})) \Rightarrow (\mathfrak{A} \Rightarrow \mathfrak{B})$;
(r3) $(\mathfrak{A} \Rightarrow \mathfrak{B}) \Rightarrow ((\mathfrak{B} \Rightarrow \mathfrak{C}) \Rightarrow (\mathfrak{A} \Rightarrow \mathfrak{C}))$;
(r4) $(\mathfrak{A} \wedge \mathfrak{B}) \Rightarrow \mathfrak{A}$;
(r5) $(\mathfrak{A} \wedge \mathfrak{B}) \Rightarrow \mathfrak{B}$;
(r6) $(\mathfrak{A} \Rightarrow \mathfrak{B}) \Rightarrow ((\mathfrak{A} \Rightarrow \mathfrak{C}) \Rightarrow (\mathfrak{A} \Rightarrow (\mathfrak{B} \wedge \mathfrak{C})))$;
(r7) $\mathfrak{A} \Rightarrow (\mathfrak{A} \vee \mathfrak{B})$;
(r8) $\mathfrak{B} \Rightarrow (\mathfrak{A} \vee \mathfrak{B})$;
(r9) $(\mathfrak{A} \Rightarrow \mathfrak{C}) \Rightarrow ((\mathfrak{B} \Rightarrow \mathfrak{C}) \Rightarrow ((\mathfrak{A} \vee \mathfrak{B}) \Rightarrow \mathfrak{C}))$;
(r10) $(\mathfrak{A} \equiv \mathfrak{B}) \Rightarrow (\mathfrak{A} \Rightarrow \mathfrak{B})$;
(r11) $(\mathfrak{A} \equiv \mathfrak{B}) \Rightarrow (\mathfrak{B} \Rightarrow \mathfrak{A})$;
(r12) $(\mathfrak{A} \Rightarrow \mathfrak{B}) \Rightarrow ((\mathfrak{B} \Rightarrow \mathfrak{A}) \Rightarrow (\mathfrak{A} \equiv \mathfrak{B}))$;
(r13) $(\mathfrak{A} \Rightarrow \mathfrak{B}) \Rightarrow ((\neg \mathfrak{B}) \Rightarrow (\neg \mathfrak{A}))$;
(r14) $\mathfrak{A} \Rightarrow (\neg (\neg \mathfrak{A}))$;
(r15) $(\neg (\neg \mathfrak{A})) \Rightarrow \mathfrak{A}$

Let us use this set of rules to produce informational theorems by means of the previous informational axioms and theorems. Propositional operands $\mathfrak{A}$, $\mathfrak{B}$, and $\mathfrak{C}$ pass over to informational operands; the similar happens to the propositional operators $\Rightarrow$, $\wedge$, $\vee$, $\equiv$, and $\neg$. This procedure is legal in an informational way, for operands and operators can be universalized from the propositional into the informational ones.

(18.7.1) THEOREM. As a consequence of rules (r1)-(r15), some informational axioms of understanding can be the following:

(r1') $\mathfrak{A} \Rightarrow (\mathfrak{B} \models \mathfrak{A})$;
(r2') $(\mathfrak{A} \models (\mathfrak{A} \models \mathfrak{B})) \Rightarrow (\mathfrak{A} \models \mathfrak{B})$;
(r3') $(\mathfrak{A} \models \mathfrak{B}) \Rightarrow ((\mathfrak{B} \models \mathfrak{C}) \Rightarrow (\mathfrak{A} \models \mathfrak{C}))$;
(r4') $(\mathfrak{A} \models \mathfrak{B}) \Rightarrow \mathfrak{A}$;
(r5') $(\mathfrak{A} \models \mathfrak{B}) \Rightarrow \mathfrak{B}$;
(r6') $(\mathfrak{A} \models \mathfrak{B}) \Rightarrow ((\mathfrak{A} \models \mathfrak{C}) \Rightarrow (\mathfrak{A} \models \mathfrak{B}, \mathfrak{C}))$;
(r7') $\mathfrak{A} \Rightarrow (\mathfrak{A} \models \mathfrak{B})$;
(r8') $\mathfrak{B} \Rightarrow (\mathfrak{A} \models \mathfrak{B})$; [eq. r1']
(r9') $(\mathfrak{A} \models \mathfrak{C}) \Rightarrow ((\mathfrak{B} \models \mathfrak{C}) \Rightarrow (\mathfrak{A}, = \models \mathfrak{C}))$;
(r10') $(\mathfrak{A} \equiv \mathfrak{B}) \Rightarrow (\mathfrak{A} \models \mathfrak{B})$;
(r11') $(\mathfrak{A} \equiv \mathfrak{B}) \Rightarrow (\mathfrak{B} \models \mathfrak{A})$;
(r12') $(\mathfrak{A} \models \mathfrak{B}) \Rightarrow ((\mathfrak{B} \models \mathfrak{A}) \Rightarrow (\mathfrak{A}, \mathfrak{B} \models \mathfrak{A}, =))$;
(r13') $(\mathfrak{A} \models \mathfrak{B}) \Rightarrow ((\not\models \mathfrak{B}) \Rightarrow (\mathfrak{A} \not\models))$;
(r14') $\mathfrak{A} \Rightarrow (\not\models (\not\models \mathfrak{A}))$;
(r15') $(\not\models (\not\models \mathfrak{A})) \Rightarrow \mathfrak{A}$ □

PROOF. The set of listed theorems of understanding is not informationally systematic, for it is directly deduced from the mathematical (propositional) algebraic system. Proofs are the following:

(r1") There is $\mathfrak{A} \Rightarrow (\models \mathfrak{A})$ and $\mathfrak{B} \models \mathfrak{A}$ is one of the possibilities of $\models \mathfrak{A}$, which is an open formula.

(r2") If $\mathfrak{A}$ informs the informing $\mathfrak{A} \models \mathfrak{B}$, then $\mathfrak{A}$ informs $\mathfrak{B}$ in an informational way.

(r3") The informing $\mathfrak{A} \models \mathfrak{B}$ implies also informing $\mathfrak{A} \models \mathfrak{C}$, if $\mathfrak{B}$ informs $\mathfrak{C}$.

(r4") Informing $\mathfrak{A} \models \mathfrak{B}$ implies $\mathfrak{A}$ as an informational operand.

(r5") Informing $\mathfrak{A} \models \mathfrak{B}$ implies $\mathfrak{B}$ as an informational operand.

(r6") Informing $\mathfrak{A} \models \mathfrak{B}$ implies $\mathfrak{A} \models \mathfrak{B}, \mathfrak{C}$, if $\mathfrak{A}$ informs $\mathfrak{C}$.

(r7") There is $\mathfrak{A} \Rightarrow (\mathfrak{A} \models)$ and $\mathfrak{A} \models \mathfrak{B}$ is one of the possibilities of $\mathfrak{A} \models$, which is an open formula.

(r8") This theorem is equivalent to (r1').

(r9") Informing $\mathfrak{A} \models \mathfrak{C}$ implies $\mathfrak{A}, \mathfrak{B} \models \mathfrak{C}$, if $\mathfrak{B}$ informs $\mathfrak{C}$.

(r10") Particular informing $\mathfrak{A} \equiv \mathfrak{B}$ implies $\mathfrak{A} \models \mathfrak{B}$, because $\mathfrak{A} \equiv \mathfrak{B}$ is only a particular informing of other possible informing of $\mathfrak{A}$ to $\mathfrak{B}$. Formula $\mathfrak{A} \equiv \mathfrak{B}$ means that $\mathfrak{A}$ informs $\mathfrak{B}$ in an equivalent way (or to be equivalent) and vice versa. On the other hand, operator $\equiv$ can be universalized into operator $\models$.

(r11") Because of informational equivalence between $\mathfrak{A}$ and $\mathfrak{B}$, formula $\mathfrak{A} \equiv \mathfrak{B}$ implies also $\mathfrak{B} \models \mathfrak{A}$; in this respect, a joint theorem of (r10') and (r11') would be appropriate, for instance:

$\mathfrak{A} \equiv \mathfrak{B} \Rightarrow (\mathfrak{A} \models \mathfrak{B};\ \mathfrak{B} \models \mathfrak{A})$

(r12") Informing $\mathfrak{A} \models \mathfrak{B}$ implies $\mathfrak{A}, \mathfrak{B} \models \mathfrak{A}, \mathfrak{B}$, if $\mathfrak{B} \models \mathfrak{A}$; this is a trivial consequence proceeding out of basic definitions.

(r13") Informing $\mathfrak{A} \models \mathfrak{B}$ implies $\mathfrak{A} \not\models \mathfrak{B}$, where operators $\models$ and $\not\models$ are general operators of informing (understanding) and non-informing (non-understanding). Further, informing $\mathfrak{A} \not\models \mathfrak{B}$ implies $\not\models \mathfrak{B}$ and $\mathfrak{A} \not\models$; thus, $(\not\models \mathfrak{B}) \Rightarrow (\mathfrak{A} \not\models)$ for the case $\mathfrak{A} \models \mathfrak{B}$.

(r14") Informing of $\mathfrak{A}$ implies $\models \mathfrak{A}$ ($\mathfrak{A}$ cannot inform in all possible ways). Non-informing $\models \mathfrak{A}$ can always (by definition) be expanded into $\models (\models \mathfrak{A})$. Thus, $\mathfrak{A} \Rightarrow (\models (\models \mathfrak{A}))$.

(r15") The reverse informational implication to (r14') is evident. As soon as $\mathfrak{A}$ appears in a formula, this formula implies the occurring (arising) of operand $\mathfrak{A}$. Q.E.D.

The presented set of theorems is in no way an adequate selection for the demonstration of the appropriateness of an informational theory of understanding or any other general theory of informing. Implication as a form of mathematical inference belongs to the most primitive means in informational sense. The so-called higher informational modi (rules of inference) are, for instance, not only modus ponens and modus tollens, but also modus rectus, obliquus, procedendi, operandi, vivendi, necessitatis, possibilitatis, etc. which are described into more detail in (IL-IV).

18.8. Conclusion

On the basis of the exposed definitions, axioms, and theorems, informational theories of understanding for various purposes, needs, and applications can be generated in(de)finitely. Mathematical theories appear to be only particular cases of informational theories. The essential distinction between mathematical and informational theories of understanding might be the following:

(1) The nature of mathematical operators is informationally static, determined firmly, that is, functional-deterministically, algorithmically, set-theoretic-definitely, algebraically, arithmetically, etc. The nature of informational operators of understanding is dynamic (arising, developing, generative). This principle sets an informational operator of understanding as a variable, similar to the operand variable. In the case of the notion of explicit and implicit informational operator we are confronted with the so-called "large" transcendence in regard to the mathematical notion of operator and function, respectively. This transcendence brings a new perspective into understanding of the multiplex nature of operators.

(2) The nature of mathematical operands is either constant (a value) or variable (a member of a set of values of different nature). The nature of informational operands is like an arising formula which in its cycles of arising is spontaneously developing as a consequence of operand's own and of other operands impacting. In this way, formulas, representing operands, are dynamic, spontaneous structures, behaving as informational entities by themselves.

(3) Mathematical expressions (symbols, formulas, functors, etc.) are meant to be fixed formal aggregates (symbolic compositions, functional structures). Informational formulas of understanding are fixed structures only exceptionally, particularly, otherwise they are dynamic, performing as arising informational operands.

(4) The entire mathematical knowledge can be incorporated into the informational concept in a senseful way and, maybe, some new understanding and possibilities of mathematical advance in informational sense can be found. It is to believe that informational concept can offer new ways of mathematical thinking and, particularly, of the informational rethinking of the realm of mathematics.

(5) Although the exposed thinking in this essay will cause a demonstrative opposition of mathematicians, it will, maybe, stimulate also the necessary rethinking of some mathematical foundations, bring a new spirit, and open the possibilities of a new constructivism on the way to living (neuronal) processes in the future scientific investigation.

19. Intelligence as Understanding and vice versa

At the beginning, let us state the question concerning intelligence and intellectualism as understanding.

Intelligence becomes a technical term (for instance, as understood in the artificial intelligence community) which stresses the meaning of intelligence as knowledge, ability to learn, information, news, technically bounded expertise, and that like. Sometimes, to be intelligent means to be informed in the usual, however, not informationally redetermined way (POI) and not predominantly in the sense of the highest possible and autonomous human creativity, but first of all in the manner of socially integrative ideology. If we exclude the meaning of intelligence as the basic eternal quality of divine Mind, intelligence becomes a regular (artificial, autopoietic) system performance, irrespective of the place of its occurrence, for instance, within technological systems as well as in the domain of regular animal and human behavior, thinking, and acting. It may be said, that intelligence as information appears as a component of understanding as information on the lowest possible level of understanding, on the border separating the so-called intelligent and non-intelligent systems. Intelligence as information becomes just the criterion of this border where real understanding begins.

On the other side, intellectualism as information pervades the realm of understanding as its autonomous and mainly creative power in understanding information. Intellectualism as information appears as a higher form of intelligence, with the emphasized cognitively creative component of understanding. While in its regular states of informing, understanding performs intelligently, so in its breakdown phases, for instance discovering its own loseableness, understanding performs in an intellectual way. If intelligence is a regular performance of understanding as information,

then intellectualism is an exceptional informational capability, by which the ruling intelligence of understanding is broken down, proceeding into a new state (or essentially different form) of intelligence within the informationally arising understanding. Thus, intellectualism as information appears as an informationally regulating power over intelligence.

Intelligence and intellectualism can be understood to be distributed within understanding as information, that is, they can arise within any component of understanding. So, let us present the formal side of the concept.

Let $\mathfrak{U}$ mark understanding, $\mathfrak{I}$ intelligence or informing of $\mathfrak{U}$ in an intelligent way, and $\mathfrak{I}^+$ intellectualism or informing of $\mathfrak{U}$ in an intellectual way. Then,

$(\mathfrak{I}1)\quad \mathfrak{I} \subset \mathfrak{I}^+;\ \mathfrak{I},\ \mathfrak{I}^+ \subset \mathfrak{U}$

At the first glance, intelligence $\mathfrak{I}$ is subordinated to intellectualism $\mathfrak{I}^+$ and both are subordinated to understanding $\mathfrak{U}$. In some cases, intellectualism $\mathfrak{I}^+$ can be subordinated to intelligence $\mathfrak{I}$, thus,

$(\mathfrak{I}2)\quad \mathfrak{I}^+ \subset \mathfrak{I};\ \mathfrak{I},\ \mathfrak{I}^+ \subset \mathfrak{U}$

The domination of intellectualism $\mathfrak{I}^+$ over intelligence $\mathfrak{I}$ in case ($\mathfrak{I}1$) constitutes the so-called intellectual mode of understanding, while the domination of intelligence $\mathfrak{I}$ over intellectualism $\mathfrak{I}^+$ in case ($\mathfrak{I}2$) constitutes the so-called pseudo-intellectual or simply intelligent mode of understanding. In the second case, intellectualism $\mathfrak{I}^+$ is in fact suppressed by intelligence $\mathfrak{I}$, thus, it can be omitted from the further investigation.

The general (or simple) intelligent model of an understanding informing $\mathfrak{U}$ becomes

$(\mathfrak{I}3)\quad (((\mathfrak{U} \models \mu) \models \mathfrak{I}) \models \eta) \models \mathfrak{U}$

where $\models$ reads 'informs in an understanding way', μ is the meaning informed (produced) by understanding $\mathfrak{U}$, and η is intelligent information as produced in the cycle of understanding ($\mathfrak{I}3$). If necessary, this model can be particularized in detail according to the needs or concrete requirements. For instance, engineering or artificial intelligence models will proceed from these simple circumstances.

The general intellectual model of an understanding informing $\mathfrak{U}$ is an informational expansion of the general intelligent model ($\mathfrak{I}3$), that is,

$(\mathfrak{I}4)\quad (((((\mathfrak{U} \models \mu) \models \mathfrak{I}) \models \eta) \models \mathfrak{I}^+) \models \eta^+) \models \mathfrak{U}$

In this intellectual model, intelligence ($\mathfrak{I}$, η) is the condition sine qua non, where intellectualism ($\mathfrak{I}^+$, η^+) arises out of intelligence as its necessary informational background. In this model, η^+ marks the so-called intellectual information, produced by intellectualism $\mathfrak{I}^+$. Thus, for instance, a primitive intellectual model of the form

$(\mathfrak{I}4')\quad (((\mathfrak{U} \models \mu) \models \mathfrak{I}^+) \models \eta^+) \models \mathfrak{U}$

either implicitly assumes the existence of intelligence or is simply only an intelligent system of the form ($\mathfrak{I}3$), that is, a quasi-intellectual system.

In the informational circle of intellectual understanding ($\mathfrak{I}4$), intellectualism arises out of intelligence, that is, is informed by intelligence, however, via understanding $\mathfrak{U}$ controls or informationally impacts intelligence within the understanding cycle. In a similar way, intellectualism informs understanding, however, it is essentially controlled by the informing of understanding within the understanding cycle. The mechanism of the informational cycle enables that intellectualism and intellectual information impact any informational component of the understanding cycle, for instance, additionally in a parallel way, that is,

$(\mathfrak{I}5)\quad \mathfrak{I}^+,\ \eta^+ \models \mathfrak{U},\ \mu,\ \mathfrak{I},\ \eta,\ \mathfrak{I}^+,\ \eta^+$

Intellectualism can be understood to be the highest level of informing within understanding $\mathfrak{U}$, that is, the most creative, decisive, and actual process of informing which can control, impact, and influence all in the understanding involved informational components of understanding.

Let us analyze two partial cycles (loops) within the informational cycle of understanding, concerning intelligence and intellectualism, that is, their possible mutual impacting and the dominance of the one over the other. For instance,

$(\mathfrak{I}6)\quad (((\mathfrak{I} \models \eta) \models \mathfrak{I}^+) \models \eta^+) \models \mathfrak{I};$
$\mathfrak{I}^+,\ \eta^+ \subset \mathfrak{I},\ \eta$

or

$(\mathfrak{I}7)\quad (((\mathfrak{I}^+ \models \eta^+) \models \mathfrak{I}) \models \eta) \models \mathfrak{I};$
$\mathfrak{I},\ \eta \subset \mathfrak{I}^+,\ \eta^+$

The case ($\mathfrak{I}6$) might be called quasi-intellectual, while the case ($\mathfrak{I}7$) is a proper intellectual case, in which intellectualism dominates over intelligence. It may happen that both partial cycles exist within understanding and are instantaneously dominating, depending from the circumstances. In the case of the intelligent dominance ($\mathfrak{I}6$), understanding loses its essential creative, critical, breakdown, also radically changing informing, so it can proceed into a stable, blind state of loseableness of understanding. If intellectualism interrupts the state of blindness or loseableness of understanding, intelligence strengthens the blindness or loseableness and performs in a counter-informational way against the intellectualism.

In the intelligent case, intelligence dominates over intellectualism, preserving the informing of intelligence dominant in regard to intellectualism. This is the well-known case of system intelligence (systemic uniformity, stupidity) in which intellectual breakdowns are minimal and exceptional. Intelligent system is lowly creative and weakly critical, and thus intelligently regular.

Intellectual system senses the loseableness of understanding by breakdowns of system blindness and changes essentially the performance of intelligence. As mentioned, both

systems can exist within understanding, opposing each other and performing counter-informational against each other, improving each other informationally, that is in an intelligent and intellectual way simultaneously. Thus, the informational conservatism of intelligence is pervaded by the informational radicalism of intellectualism and vice versa. Understanding as information is the framework of this intelligent and intellectual game.

20. Meaning as a Result of Understanding

> ... in one case we take language as conveying information and instruction; in the other case we leave aside the individual participant and see the process of conversation and understanding as a distributed, coherent events shared among participants, where meaning and understanding is relative to the recursive process of interpretation within the conversational unit. ...
>
> Francisco J. Varela (PBA) 269

What is meaning in regard to understanding? How does meaning arise by understanding in a complex, distributive, and distributed informational cycle of understanding? How the complexity of understanding can be considered in a cyclic and a parallel way simultaneously? Is understanding as an arising information the most complex form of informing which can be imagined, although such a form of understanding cannot be performed physically by a human autopoietic system? Does it mean that a much more complex artificial informational machine could beat a human communicating community in regard to the quality and complexity of understanding? Why we are putting such questions right at this place of our conversation? Let us look at the following formal informational concept of understanding.

At the beginning of our understanding discourse or discourse of understanding we have to put the question of the complexity of an ongoing understanding. What are the informational components of understanding? They can arise and disappear in a spontaneous and circular way. In this point we can turn to ourselves, to the image we have on that how our minds build up the informational phenomenology of understanding. The very classical and basic components of understanding $\mathfrak{U}$ can be, for instance:

the sensing $\mathfrak{S}$ producing the information of sensing (or sense), marked by σ;

the observing $\mathfrak{S}^+$ which may appear to be a higher form of sensing $\mathfrak{S}$, producing the information of observing (or observation), marked by σ^+;

the perceiving $\mathfrak{P}$, producing the information of perceiving (or perception), marked by π;

the conceiving $\mathfrak{C}$, producing the information of conceiving (or conception), marked by γ;

the comprehending $\mathfrak{C}^+$ which may appear to be a higher form of conceiving $\mathfrak{C}$, producing the information of comprehending (or comprehension), marked by γ^+;

the understanding $\mathfrak{U}$, producing the information of understanding (or meaning), marked by μ.

It is to say that these components of understanding are rather arbitrary than systematic, for in a concrete case, other components may appear and the above introduced ones disappear. The listed components can merely form a solid and evident ground of our further discussion.

Thus, for the basic aggregate of understanding we can set roughly

$$(\mu 1) \quad \mathfrak{U} = (\mathfrak{U}, \mathfrak{S}, \mathfrak{S}^+, \mathfrak{P}, \mathfrak{C}, \mathfrak{C}^+ \models \mathfrak{U}, \mathfrak{S}, \mathfrak{S}^+, \mathfrak{P}, \mathfrak{C}, \mathfrak{C}^+)$$

or in a more complex form

$$(\mu 2) \quad \mathfrak{U} = (\mathfrak{U}, \mu; \mathfrak{S}, \sigma; \mathfrak{S}^+, \sigma^+; \mathfrak{P}, \pi; \mathfrak{C}, \gamma; \mathfrak{C}^+, \gamma^+ \models \mathfrak{U}, \mu; \mathfrak{S}, \sigma; \mathfrak{S}^+, \sigma^+; \mathfrak{P}, \pi; \mathfrak{C}, \gamma; \mathfrak{C}^+, \gamma^+)$$

Formula (μ2) is a general scheme or scenario of understanding that can be elaborated (developed, constructed) into detail, that is, adding new components and considering the listed ones in a spontaneous and circular way.

Let us analyze the listed components of understanding in an informational way. Sensing $\mathfrak{S}$ as informing of sense σ is sensed by the outward (arriving, environmental) information α as well as the inward, that is, understanding information, say, β. This process of sensing produces information of sensing σ, that is,

$$(\mu 3) \quad (\alpha, \beta \models \mathfrak{S}) \models \sigma$$

Thus, through β the process of understanding will be closed in one or another way. Operand β will mark the distinct components of understanding, taking part in the production of sensing information.

Observing $\mathfrak{S}^+$ will depend on the results of sensing and on some understanding components, marked by β, and will produce information of observing σ^+, for instance,

$$(\mu 4) \quad ((((\alpha, \beta \models \mathfrak{S}) \models \sigma), \beta) \models \mathfrak{S}^+) \models \sigma^+$$

We see how the complexity raises already at the second component of understanding. Operands β at different points of the cycle may mark different partial (componential) entities of understanding, re-entering from the side of the resulting understanding. Besides this cyclic complexity also parallel informational links among components of understanding can exist, for instance in the sense of formula (μ2), where general operator $\models$ can be replaced by its parallel or parallel-cyclic particularization, that is, by $\Vdash$ or $\models\!\!\!\models$, respectively.

In our case, perceiving $\mathfrak{P}$ is on the third hierarchical level of understanding, thus, it will depend consequently on results of sensing and observing and, additionally to that, on some understanding components, marked by β. Such complex perceiving will produce information of perceiving (observation) π, yielding formula

$$(\mu 5) \quad (((((((\alpha, \beta \models \mathfrak{S}) \models \sigma), \beta) \models \mathfrak{S}^+) \models \sigma^+), \beta) \models \mathfrak{P}) \models \pi$$

We see how the construction of the main informational cycle of understanding $\mathfrak{U}$ is a rather canonical way of hierarchical structure, which can be generalized by the form

$$(\alpha, \beta \models \gamma) \models \delta$$

where, successively, α is α (outward information) or formula (μi), β is any possible component combination of $\mathfrak{U}$ (μ2), γ is the next hierarchical component of $\mathfrak{U}$, for instance, $\mathfrak{U}_1$, $\mathfrak{U}_2$, ..., and δ is to this component belonging "meaning", say, μ_1, μ_2, ... Thus,

$$(\mu_{i+1}) \quad ((\mu i), \mathfrak{U} \models \mathfrak{U}_{i+1}) \models \mu_{i+1}; \quad i = 2, 3, \ldots$$

In our case, i = 2, 3, 4, 5, 6, 7 corresponds to the sequence α, $\mathfrak{S}$, $\mathfrak{S}^+$, $\mathfrak{P}$, $\mathfrak{C}$, $\mathfrak{C}^+$, where index i+7 marks $\mathfrak{U}$; further, μ_3, μ_4, ... μ_8 are markers for σ, σ^+, π, γ, γ^+, μ, respectively.

Continuing our case, on the fourth hierarchical level of understanding is conceiving $\mathfrak{C}$. According to the previous discussion, there is,

$$(\mu 6) \quad (((((((((\alpha, \beta \models \mathfrak{S}) \models \sigma), \beta) \models \mathfrak{S}^+) \models \sigma^+), \beta) \models \mathfrak{P}) \models \pi), \beta) \models \gamma$$

Further, on the fifth hierarchical level is comprehending $\mathfrak{C}^+$, which yields

$$(\mu 7) \quad ((((((((((((\alpha, \beta \models \mathfrak{S}) \models \sigma), \beta) \models \mathfrak{S}^+) \models \sigma^+), \beta) \models \mathfrak{P}) \models \pi), \beta) \models \gamma), \beta) \models \mathfrak{C}^+) \models \gamma^+$$

On the last hierarchical level is understanding $\mathfrak{U}$ and at this point finally the meaning μ is coming into existence:

$$(\mu 8) \quad ((((((((((((((\alpha, \beta \models \mathfrak{S}) \models \sigma), \beta) \models \mathfrak{S}^+) \models \sigma^+), \beta) \models \mathfrak{P}) \models \pi), \beta) \models \gamma), \beta) \models \mathfrak{C}^+) \models \gamma^+), \beta) \models \mathfrak{U}) \models \mu$$

In this formula, β can be replaced by $\mathfrak{U}$ in the sense of formula (μ2), which yields

$$(\mu 8') \quad ((((((((((((((\alpha, \mathfrak{U} \models \mathfrak{S}) \models \sigma), \mathfrak{U}) \models \mathfrak{S}^+) \models \sigma^+), \mathfrak{U}) \models \mathfrak{P}) \models \pi), \mathfrak{U}) \models \gamma), \mathfrak{U}) \models \mathfrak{C}^+) \models \gamma^+), \mathfrak{U}) \models \mathfrak{U}) \models \mu$$

At this formula we see how the complexity of understanding becomes right on the level of only several hierarchical components an extremely perplexed informational process.

The spontaneity and cyclicity, and the parallelism of these phenomena of informing within an understanding process can be formalized in several ways. For instance, if an external information α is coming into an understanding consideration, we can roughly use the following (initial) scheme:

$$(\mu 9) \quad (\alpha \models \mathfrak{S}) \models \sigma; \quad \text{[sensing phase]}$$
$$(\sigma \models \mathfrak{S}^+) \models \sigma^+; \quad \text{[observing phase]}$$
$$(\sigma^+ \models \mathfrak{P}) \models \pi; \quad \text{[perceiving phase]}$$
$$(\pi \models \mathfrak{C}) \models \gamma; \quad \text{[conceiving phase]}$$
$$(\gamma \models \mathfrak{C}^+) \models \gamma^+; \quad \text{[comprehending phase]}$$
$$(\gamma^+ \models \mathfrak{U}) \models \mu \quad \text{[understanding phase]}$$

Out of this initial scheme more and more reversal (recursive), complex, and perplexed network of understanding can be developed (projected and thereupon constructed) and, of course, informationally particularized to the needed extent. Implicitly (or functionally), the act of understanding can be marked by the conventional symbolism, for instance, $\mathfrak{U}(\mu(\alpha))$, where meaning μ depends on α and understanding $\mathfrak{U}$ is a function of the arisen meaning. Simultaneously, $\mu(\mathfrak{U}(\alpha))$ can be considered. Etc.

21. Hermeneutics as a Modus of Understanding

> ... Hermeneutics is the way [...] understanding enlightens itself; ...
>
> M. Heidegger (BT) 450

In principle, hermeneutics as a modus of understanding is characterized by the cyclicity of understanding in a specific sense, that is, to consider previous or even in the distant past arisen informational cycles of understanding of distinct informational entities. Hermeneutics is one of the specific aspects of understanding of understanding, that is, of self-understanding as information, putting into the foreground the question of the historical conditionality of understanding. Understanding is a process of informing-historical informational occurring of understanding with its own (derivative) way of interpretation. As a hermeneutically distributed process of understanding, hermeneutics pervades the interpretation of understanding components, emphasizing the historicity of understanding. Additionally, this historical interpretation of understanding can stress, for instance, a specific practical cognition, good sense, authenticity, explaining, interpreting, and meaning of the past by the present understanding.

Hermeneutics is nothing else than a specific, historically concerning, individual, and pragmatic orientation of understanding. It concerns the interpretation of understanding through its informational history, through that what occurred informationally in the previous informational cycles of understanding. Hermeneutics is the study of informational occurrence or informing, is interpretation of informing within informational cycles, and, in this respect, is also re-interpretation of understanding in cycles before through a historical look, by which the ongoing understanding is disclosed as simultaneousness of past, present, and future.

Let β be a previous information and let $\mathfrak{B}$ be its previous understanding, which produced the meaning μ_β. The problem of hermeneutics is to understand (explain, clarify, reconstruct) the process of previous understanding (in a global form), as

$$(H1) \quad (\beta \models \mathfrak{B}) \models \mu_\beta$$

by the present (state of) understanding $\mathfrak{U}$. Thus, in a linear way,

(H2) $(((\beta \models \mathfrak{B}) \models \mu_\beta) \models \mathfrak{U}) \models \mu$

In this form, $(\beta \models \mathfrak{B}) \models \mu_\beta$, as a whole, is the object of the present understanding $\mathfrak{U}$. The interpretation of $(\beta \models \mathfrak{B}) \models \mu_\beta$ is taking place within the so-called hermeneutical cycle, that is, as

(H3) $(\mu, ((\beta \models \mathfrak{B}) \models \mu_\beta) \models \mathfrak{U}) \models \mu$

where μ is the present meaning of $(\beta \models \mathfrak{B}) \models \mu_\beta$, that is, concerning not only the previous information β, but also the previous way of understanding $\mathfrak{B}$, and through that the entire previous process of understanding $(\beta \models \mathfrak{B}) \models \mu_\beta$. The most indicative point of this understanding investigation is the difference $\delta(\mu, \mu_\beta)$ occurring between μ and μ_β. Thus, consequently,

(H4) $((\mu, ((\beta \models \mathfrak{B}) \models \mu_\beta) \models \mathfrak{U}) \models \mu) \Rightarrow \delta(\mu, \mu_\beta)$

Of course, this difference can be considered in the process of understanding in a more complex manner, for instance, as

(H5)
$((((\mu, ((\beta \models \mathfrak{B}) \models \mu_\beta) \models \mathfrak{U}) \models \mu) \Rightarrow \delta(\mu, \mu_\beta)) \models \mathfrak{U}) \models \mu$

where (H4) becomes a subcycle of the main cycle of understanding. The main cycle considers the arising difference $\delta(\mu, \mu_\beta)$ within the subcycle, and so forth.

In fact, any cycling of understanding of some information - and understanding of something is always an informational cycling - is hermeneutical in itself. For instance, understanding within a being's metaphysics is always hermeneutical. For instance, if α is understood to be a regularly arising (on-line, real-time) information, then its understanding automatically concerns also its history, that is,

(H6) $(\alpha, \mu \models \mathfrak{U}) \models \mu$, where
$(\alpha \models \alpha) \models \alpha$

"guarantees" the history of α. In this process, the meaning μ produced by $\mathfrak{U}$ is informationally approaching to the informing (that is, to the essence, Being, entity, information) of α.

22. Formalization of Semiotics as a Particular Form of Understanding

Semiotics is a concept of understanding and, in this way, it can be seen as a part of understanding, that is, the so-called semiotic understanding. Semiotic understanding is structured (constructed artificially) in the following way: semiotics $\underline{s}$ is an informational set of perplexed semiotic components, that is, $\underline{s} = (s, s^+, p)$, where s marks the syntax, s^+ the semantics, and p the pragmatics. The informing part of semiotic understanding is the semiotic informing $\underline{S}$, which is an informational set of perplexed informing components, that is, $\underline{S} = (S, S^+, P)$, where S marks syntactic informing, S^+ semantic informing, and P pragmatic informing of understanding.

Semiotics $\underline{s}$ in a narrower sense belongs to the category of meaning μ and the semiotic informing $\underline{S}$ to the understanding $\mathfrak{U}$. It is supposed that an input information α has its own semiotic structure, for instance, $(\underline{S}_\alpha, \underline{s}_\alpha)$. By means of semiotic understanding $\underline{S}$ the input information α is analyzed syntactically, semantically, and pragmatically and then by semiotic synthesis, by means of semiotics $\underline{s}$, the meaning $\mu(\alpha)$ of α is generated in a syntactical, semantical, and pragmatical way. In this process, the standard components of understanding (for instance, sensing, observing, perceiving, conceiving, and comprehending) are controlled in a semiotic way, within two cyclically successive steps of understanding: by a semiotic way of informing $\underline{S}$ and then by a semiotic standardization step of the produced meaning. Through this process, semiotics performs the control over the function of understanding within an semiotic understanding system, that is,

(S1) $(\alpha, \mu(\alpha) \models \underline{S}) \models_{\underline{s}} \mu(\alpha)$

In this system, $\underline{S}$ performs analytically and then together with $\underline{s}$ synthetically, constructing the meaning of α. System (S1) understands α syntactically as $s(\alpha)$, semantically as $s^+(\alpha)$, and pragmatically as $p(\alpha)$, where α appears syntactically as $s_\alpha(\alpha)$, semantically as $s^+_\alpha(\alpha)$, and pragmatically as $p_\alpha(\alpha)$, that is, as $\alpha = (s_\alpha(\alpha), s^+_\alpha(\alpha), p(\alpha))$ or

(S2) $((\alpha \models S_\alpha(\alpha)) \models s_\alpha(\alpha)) \models \alpha$;
$((\alpha \models S^+_\alpha(\alpha)) \models s^+_\alpha(\alpha)) \models \alpha$;
$((\alpha \models P_\alpha(\alpha)) \models p_\alpha(\alpha)) \models \alpha$

Formula (S1) can be decomposed in the following way, for instance:

(S3) $((s_\alpha(\alpha), s^+_\alpha(\alpha), p_\alpha(\alpha)),$
$(\mu(s_\alpha(\alpha)), \mu(s^+_\alpha(\alpha)), \mu(p_\alpha(\alpha))) \models$
$(S, S^+, P)) \models_{(s, s^+, p)}$
$(\mu(s_\alpha(\alpha)), \mu(s^+_\alpha(\alpha)), \mu(p_\alpha(\alpha)))$

By this formula, semiotic kinds of meaning and semiotic components of informing and semiotics in narrower sense become diversely perplexed. Thus, syntax can influence semantics as well as pragmatics, semantics can influence syntax as well as pragmatics, and pragmatics can influence syntax as well as semantics on the level of semiotic components of meaning. We recognize how semiotic informational entities can be understood in a semiotic way.

23. The Loseableness and Losing of Understanding

The loseableness and loosing of understanding are basic informational phenomena for ascertaining of deficiency occurring during the process of understanding. Loseableness and losing of understanding can concern in

principle all distinctions, that is, components of understanding and, thus, for instance, sensing, observing, perceiving, conceiving, and comprehending within the process of understanding may become insensitive, deficiently observable, insufficiently perceivable, inadequately conceivable, and miscomprehensive, respectively. In course of its arising, understanding can lose or fail certain essential informing, so it cannot progress in some possible domains or open new horizons of distinct understanding. Thus, after and during the informing of understanding, its deficiencies can be sensed, observed, perceived, conceived, and comprehended, constituting the so-called loseableness and losing of understanding as information.

The components of loseableness and losing of understanding can be marked by indexed $\mathfrak{L}$ (loseableness) and $\mathfrak{R}$ (losing) for their occurrences as informing and, further, by indexed λ (loseableness) and ρ (losing) for their occurrences as information, resulting from such informing. For instance, according to Heideggerian concepts of loseableness (UAI), we can introduce:

$\mathfrak{L}_{ww}$ and λ_{ww} for informing and information of understanding lost in what is encountered within-the-world;
$\mathfrak{L}_{eq}$ and λ_{eq} for informing and information of understanding lost in equipment;
$\mathfrak{L}_{ed}$ and λ_{ed} for informing and information of understanding lost in everydayness;
$\mathfrak{L}_{fc}$ and λ_{fc} for informing and information of understanding lost in factual circumstances;
$\mathfrak{L}_{ir}$ and λ_{ir} for informing and information of understanding lost in irresoluteness;
$\mathfrak{L}_{ja}$ and λ_{ja} for informing and information of understanding lost in just-always-already-alongside;
$\mathfrak{L}_{mp}$ and λ_{mp} for informing and information of understanding lost in the making-present of the "to-day";
$\mathfrak{L}_{pt}$ and λ_{pt} for informing and information of understanding lost in possibilities which trust themselves upon one;
$\mathfrak{L}_{pu}$ and λ_{pu} for informing and information of understanding lost in publicness;
$\mathfrak{L}_{si}$ and λ_{si} for informing and information of understanding lost in something with which information is concerned;
$\mathfrak{L}_{ty}$ and λ_{ty} for informing and information of understanding lost in the "they";
$\mathfrak{L}_{ts}$ and λ_{ts} for informing and information of understanding lost in the they-self;
$\mathfrak{L}_{we}$ and λ_{we} for informing and information of understanding lost in the world of equipment; and
$\mathfrak{L}_{ow}$ and λ_{ow} for informing and information of understanding lost in one's world.

Further, according to Heideggerian concepts of what understanding can lose (UAI), we can introduce:

$\mathfrak{R}_{ar}$ and ρ_{ar} for informing and information of understanding losing its aroundness;
$\mathfrak{R}_{ba}$ and ρ_{ba} for informing and information of understanding losing its basis;
$\mathfrak{R}_{Be}$ and ρ_{Be} for informing and information of understanding losing its Being;
$\mathfrak{R}_{Bt}$ and ρ_{Bt} for informing and information of understanding losing the Being of its "there";
$\mathfrak{R}_{Bw}$ and ρ_{Bw} for informing and information of understanding losing its Being-in-the-world;
$\mathfrak{R}_{ec}$ and ρ_{ec} for informing and information of understanding losing its equipmental character;
$\mathfrak{R}_{fo}$ and ρ_{fo} for informing and information of understanding losing its force;
$\mathfrak{R}_{ge}$ and ρ_{ge} for informing and information of understanding losing its genuineness;
$\mathfrak{R}_{ic}$ and ρ_{ic} for informing and information of understanding losing its indigenous character;
$\mathfrak{R}_{iv}$ and ρ_{iv} for informing and information of understanding losing its involvement-character;
$\mathfrak{R}_{is}$ and ρ_{is} for informing and information of understanding losing itself;
$\mathfrak{R}_{rh}$ and ρ_{rh} for informing and information of understanding losing its readiness-to-hand;
$\mathfrak{R}_{ti}$ and ρ_{ti} for informing and information of understanding losing its time; etc.

How can now these Heideggerian notions concerning understanding be involved in a formal way? How to build up formulas considering this sort of informational phenomenology within understanding? If loseableness can be comprehended as a state of lostness of understanding within itself, then loosing is a process of diminishing, vanishing, or disappearing of appropriate understanding. The loseableness is similar to informational cycling, for instance, in its own and outward insensibility, non-conditionality, uncommonness, unreasonableness, hopelessness, and unsuccessfulness. The loseableness of understanding can be constituted by a closed, firm, and self-sufficient informational cycling, being informationally isolated for the acceptance of other, different, and distinct information. In this way, loseableness of understanding supports itself and arises into its own realm of loseableness.

Let us introduce $\mathfrak{L}_i$, λ_i for $i \in \{ww, eq, ed, fc, ir, ja, mp, pt, pu, si, ty, ts, we, ow\}$ and $\mathfrak{R}_k$, ρ_k for $k \in \{ar, ba, Be, Bt, Bw, ec, fo, ge, ic, iv, is, rh, ti\}$. Let us see how a particular loseableness $\mathfrak{L}_i$, λ_i and a particular losing $\mathfrak{R}_k$, ρ_k is/are involved during the process of understanding $\mathfrak{U}$.

Let us suppose the basic informational cycles within understanding $\mathfrak{U}$ in the following form:

($\mathfrak{L}1$) $(\mathfrak{U} \models \mathfrak{L}_i) \models \mathfrak{U}$;
($\mathfrak{R}1$) $(\mathfrak{U} \models \mathfrak{R}_k) \models \mathfrak{U}$

or in a more complex form

($\mathfrak{L}2$) $((\mathfrak{U} \models \mathfrak{L}_i) \models \lambda_i) \models \mathfrak{U}$;
($\mathfrak{R}2$) $((\mathfrak{U} \models \mathfrak{R}_k) \models \rho_k) \models \mathfrak{U}$

Considering meaning μ produced by understanding $\mathfrak{U}$, we can introduce

$(\Omega 3)$ $(((\mu \models \mathfrak{U}) \models \Omega_i) \models \lambda_i) \models \mu;$
$(\Re 3)$ $(((\mu \models \mathfrak{U}) \models \Re_k) \models \rho_k) \models \mu$

etc. In (Ω_3) and $(\Re_3)$ meaning μ dominates over its understanding $\mathfrak{U}$, that is, $\mathfrak{U} \subset \mu$. In the other case these roles can be changed, for instance,

$(\Omega 4)$ $((((\mathfrak{U} \models \mu) \models \Omega_i) \models \lambda_i) \models \mathfrak{U}) \Rightarrow (\mu \models \mathfrak{U});$
$(\Re 4)$ $((((\mathfrak{U} \models \mu) \models \Re_k) \models \rho_k) \models \mathfrak{U}) \Rightarrow (\mu \models \mathfrak{U})$

Thus, we can resume these cases in a more complete form in the following sense:

$(\Omega\Re 1)$ $((\Omega 1), (\Re 1)) \Rightarrow (\Omega_i, \Re_k \subset \mathfrak{U});$
$((\Omega 2), (\Re 2)) \Rightarrow ((\Omega_i, \lambda_i, \Re_k, \rho_k \subset \mathfrak{U});$
$(\Omega_i \subset \lambda_i); (\Re_k \subset \rho_k));$
$(\Omega\Re 3)$ $((\Omega 3), (\Re 3)) \Rightarrow$
$((\mathfrak{U}, \Omega_i, \lambda_i, \Re_k, \rho_k \subset \mu);$
$(\mathfrak{U}, \Omega_i \subset \lambda_i); (\mathfrak{U}, \Re_k \subset \rho_k);$
$(\mathfrak{U} \subset \Omega_i); (\mathfrak{U} \subset \Re_k))$

Cases $(\Omega 4)$ and $(\Re 4)$ imply more detailed consequences, that is,

$(\Omega\Re 4)$ $((\Omega 4), (\Re 4)) \Rightarrow$
$((\mu, \Omega_i, \lambda_i, \Re_k, \rho_k \subset \mathfrak{U});$
$(\mu, \Omega_i \subset \lambda_i); (\mu, \Re_k \subset \rho_k);$
$(\mu \subset \Omega_i); (\mu \subset \Re_k))$

in which roles of μ $(\Omega\Re 3)$ and $\mathfrak{U}$ are exchanged (mutually replaced). It seems that in the intelligent case, meaning subordinates understanding to meaning $(\Omega\Re 3)$ and that in the intellectual case, understanding subordinates meaning to understanding $(\Omega\Re 4)$. We recognize how the dominance of components of meaning and understanding can be varied and become embedded in the one way or another.

Through the losing of understanding some informational distinctions disappear from the horizon of understanding and thus $\mathfrak{U}$ is becoming poorer and poorer in an understanding way. How can this disappearing of understanding be sensible by a formula?

For instance, losing $\Re_k$, producing the information of losing ρ_k, diminishes (reduces) into a steady form of losing $\bar{\Re}_k$, producing $\bar{\rho}_k$ as a linear process of diminishing understanding, that is,

$(\Re 5)$ $(\models (((\Re_k \models \rho_k) \models \bar{\Re}_k) \models \bar{\rho}_k) \models$

within an informational cycle. By a careful observing, even the beginning of this diminishing process can be sensed, for instance, in the form

$(\Re 6)$ $(\models (((\Re_k \models \rho_k) \llcorner \bar{\Re}_k) \models \bar{\rho}_k) \models$

where $\llcorner$ marks the operator of informational beginning and where entities $\bar{\Re}_k$ and $\bar{\rho}_k$ come into existence and begin to inform within the entire informational cycle of understanding.

Linear cyclic segments $(\Re 5)$ and $(\Re 6)$ bring to the surface another considerable phenomenon of the concept of informational cycle in general. These segments hint to the following general philosophy of a cycle: in general case, linear segments of a cycle can be formulated as

(s1) $(\models \mathfrak{U}) \models$ or
(s2) $\models (\mathfrak{U} \models)$

Both formulas are open in the sense to be informed and to inform, however, in the essentially different ways. In (s1), understanding $\mathfrak{U}$ informs as a consequence of the entire history of $\mathfrak{U}$, of all the informational impacts onto $\mathfrak{U}$. In fact, formally, $\mathfrak{U}$ does not inform directly, but through the entity $(\models \mathfrak{U})$, that is, as $(\models \mathfrak{U}) \models$. And this formula says explicitly that $(\models \mathfrak{U})$ informs.

In case (s2), $\mathfrak{U}$ informs from itself other entities by the open formula $(\mathfrak{U} \models)$ and only that what was informed by $\mathfrak{U}$ can be informed by something else. Within a cycle, this informational segments become

(t1) $(\mathfrak{B} \models \mathfrak{U}) \models \mathfrak{B}$ or
(t2) $\mathfrak{B} \models (\mathfrak{U} \models \mathfrak{B})$

These are two essentially different concepts of informational cycle. The first formula can imply $\mathfrak{U} \subset \mathfrak{B}$, and the second formula, $\mathfrak{B} \subset \mathfrak{U}$, and so forth.

24. A Formalization of the Blindness and the Breakdown of Understanding

> ... Indeed the three schools of thought with which most of us began our official philosophizing about mathematics - Intuitionism, Formalism, and Logicism - all stand in fundamental disagreement with Platonism. Nevertheless, various versions of Platonistic thinking survive in contemporary philosophical circles. ...
>
> Penelope Maddy (RCP) 1121

Blindness of understanding is a course of the process of understanding which lacks the perceiving of external informational disturbances, performing as breakdowns of this course and causing essential changes which can arise into new orientations. A blind understanding is an informationally steady process pervaded, for instance, with its specific truth, knowledge, belief, myth, worship, and cult. This course of understanding considers the care of its improvement and reality only in an unrevealed and unexplained way, being safely closed into its own spontaneous circulation, so that outward information, irrespective of its character, intensity, importance, contents, or meaning is predominantly sensed as informational noise. By definition, informational noise ν performs as inessential information for its observer (understanding). In this way, informational noise cannot inform the concerned information in a way other than a noisy way. Thus, the main axiom concerning informational noise ν in regard to understanding $\mathfrak{U}$ becomes

(B1) $(\nu \models \mathfrak{U}) \Rightarrow \mathfrak{U}$

The analysis of the left side of this implication yields

(B2) $((\nu \models_{\nu} \circ \models_{\mathfrak{A}} \mathfrak{A}) \models (\nu \models_{\mathfrak{A}} \mathfrak{A})) \Rightarrow (\models_{\mathfrak{A}} \mathfrak{A})$

In the case of informational noise, there is

(B3) $(\models \circ \models_{\mathfrak{A}} \mathfrak{A}) \Rightarrow (\models_{\mathfrak{A}} \mathfrak{A})$

explicating the informational closeness of blinded understanding. A blind understanding as information performs insensible to its outward informational domain.

Blindness B as an implicit informational component of understanding $\mathfrak{A}$ is a hidden and partly unaware component of understanding and can be implicitly distributed property of other understanding components, thus, each of them is insensitive to an essential extent in sensing, observing, perceiving, conceiving, and comprehending. The blindness of understanding as an informational phenomenology contributes to the closed spontaneity and circularity in the form of firm, solid, insensible informing of understanding. On the other side, a certain extent of blindness of understanding guarantees the stability of the informational cycling, keeping the produced meaning μ as a safe, reliable, and believable information. However, blindness B appears always as an implicit, unsensitized, distributed, unaware, and unconscious informational component of understanding $\mathfrak{A}$. Therefore, breaking the blindness of $\mathfrak{A}$ means always breaking down (or interrupt) the ruling, the dominant orientation of $\mathfrak{A}$ by an outward informational impulse.

While circulating in an instantaneously stable cycle, understanding $\mathfrak{A}$ can be broken down by the arising of the broken-down blindness $\mathfrak{B}$, caused by the breaking impulse, called the breakdown information β. Thus,

(B4) $\beta \llcorner (((B \models \mathfrak{B}) \models b) \subset \mathfrak{A});$
$(\mathfrak{B} \llcorner B) \models \mathfrak{B}$

describes that β begins (informational operator $\llcorner$ marks the beginning) the process in which broken-down blindness $\mathfrak{B}$ arises out of blindness B, producing the information of breakdown b within understanding $\mathfrak{A}$. At the beginning, the broken-down blindness $\mathfrak{B}$ becomes a weak, but already distinctive component of $\mathfrak{A}$, which begins to destroy cyclically the ruling blindness B more and more into the sense of $\mathfrak{B}$.

25. Formalization of an Informational Expert Systems

Informational expert systems (IES's) constitute a new, the so-called informational approach to the projection (in German, der Entwurf, i.e. in the sense of throwing out; in English, project, design, construction) of living and artificial expert systems. An IES can be classified as a system containing the property of the so-called expert understanding of an outward information by some (or all) of its components. According to this informational containment, IES's can be in the range of the primitively informational to the most complexly and perplexedly informational ones. Now-a-days, expert systems do not reach the level of the most primitive IES at all.

Let α be information coming into the expert consideration (understanding) by an IES. Operand α can certainly be only the marker of a complex, multi-componential set of informational operands, for instance, $\alpha_1, \ldots, \alpha_m$. Let $E_1, \ldots, E_n$ be already available but also arising expert modes of understanding of an IES. Further let $e_1, \ldots, e_n$ be expert informational entities concerning the input information α and produced (informed) by the expert modes of understanding $E_1, \ldots, E_n$. Further, let E be the master (integrative) expert mode of understanding of an IES and let e mark the master (integrative) expert information (resulting expertise) concerning α. Under these circumstances various IES's can be determined, for instance, marking them by IES's of rank 0, 1, 2, and so forth, according to their informational power, that is, according to the extent of their informational (expert) complexity, perplexity, and hierarchical structure. Within this classification, linear and circular IES's can be distinguished.

The most primitive, linear IES would be the one with the empty set of the expert modes of understanding $E_1, \ldots, E_n$. This is the linear IES of rank 0:

(E1) $\alpha \models E; E \models e$

Rank 0 marks a kind of improper linear IES. The next step is the linear IES of rank 1

(E2) $\alpha \models E_1, \ldots, E_n;$
$E_1 \models e_1; \ldots; E_n \models e_n;$
$e_1, \ldots, e_n \models E;$
$E \models e$

The choice and arising of $E_1, \ldots, E_n$ of an IES depends on the informational nature of α, that is, IES is informationally sensitive to the nature of input information in respect to its organization. That means that an implicitly or for the observer not visible structure of IES exists which decides on the choice and arising of the expert modes of understanding $E_1, \ldots, E_n$. Further, it is to stress that $E_1, \ldots, E_n$ and E are proper informational entities which inform, counter-inform and embed information according to the principles of information. So, $E_1, \ldots, E_n$ and E are proper modes of understanding in an informational way. In case (E2), α is an outward information, however, the produced $e_1, \ldots, e_n$ and e are data (messages), that is passive informational entities (products).

The next step of development of linear IES's would be to let $e_1, \ldots, e_n$ and e co-inform in the system, but still in a linear informational way, that is, without explicit informational expert cycles.

The most powerful or, in fact, the proper IES's can be constructed by the introduction of the expert (conceptual) circularity into the game of producing a real expertise of the accessed outward information α. Thus, a linear IES (E1) can be expanded into the circular IES of rank 0 only by the assumption that entity E within the system $\alpha \models E; E \models e$ performs

in an informationally circular way. This is certainly true, for E is a regular informational entity.

Similarly as in the linear case, rank 0 marks a kind of improper circular IES. Thus, the next step is the circular IES of rank 1, that is,

(E3) $\alpha \models E_1, \ldots, E_n;$
$E_1 \models e_1; \ldots; E_n \models e_n;$
$e_1, \ldots, e_n \models E;$
$E \models E_1, \ldots, E_n;$
$E \models e$

The expert cycle in (E3) is a distributed form of cyclicity in regard to E, that is

(E4) $E_i \models e_i; e_i \models E; E \models E_i; i = 1, \ldots, n$

This type of cyclicity differs essentially from the case of the so-called continuous (a history concerning or hermeneutical) cyclicity, which would be

(E5) $((E_i \models e_i) \models E) \models E_i; i = 1, \ldots, n$

The distributivity of the master expert understanding E is reflected in $i = 1, \ldots, n$.

The next innovative step in the direction of the circular, on-line (real-time) IES's of the second rank would be the following: let in (E3), $e_1, \ldots, e_n$ and e be proper informational entities which inform, counter-inform, and embed information. In this case, these informational entities can inform in a proper way to the particular, first-level expert modes of understanding, that is,

(E6) $e, e_1, \ldots, e_n \models E_1, \ldots, E_n$

This system represents a complete mode of cross informing among the left and the right entities of operator $\models$ in (E6), that is,

(E7) $e \models E_1; \ldots; e \models E_n;$
$e_1 \models E_1; \ldots; e_1 \models E_n;$
. . . . . .
$e_n \models E_1; \ldots; e_n \models E_n$

Further, because of the attribute on-line or real-time, the input information α, considered in an expert way, must be observed continuously and in distributed manner, for instance, as

(E8) $(\alpha \models E_i) \models e_i; i = 1, \ldots, n$

Thus, the complete formula of the on-line, circular, and distributed IES of rank 2 becomes

(E9) $(\alpha \models E_i) \models e_i; i = 1, \ldots, n;$
$e_1, \ldots, e_n \models E;$
$E \models E_1, \ldots, E_n;$
$E \models e;$
$e, e_1, \ldots, e_n \models E_1, \ldots, E_n$

However, we recognize how this formula is still reduced in the sense of understanding of α. For proper understanding of α more and more historicity of the previous (historically conditioned) understanding of α has to be taken into consideration. A higher rank of the continuous and distributed understanding of α in the described way would be, for instance,

(E10) $(((\alpha \models E_i) \models e_i) \models E) \models e;$
$i = 1, \ldots, n$

where n is certainly a independent variable! The last formula is an approach to a parallel IES with much more proper (historical) understanding of α.

By the previous discussion, the feeling of possibility of the projection of multi-level IES's is coming into the consciousness. A multi-level IES of rank i would be simply in introducing the particular modes of expert understanding, for instance, in the form

(E11) $E_{ik}; i = 1, \ldots, m; k = 1, \ldots, n$

This introduction would yield, for instance,

(E12)
$E_{11}, \ldots, E_{1x};$ [first level]
$E_{21}, \ldots, E_{2y};$ [second level]
. . . .
$E_{i1}, \ldots, E_{iz};$ [i-th level]

In this system, i, x, y, and z function as constructively independent variables. Their concrete occurrence can be conditioned by the nature of α and by the mode of understanding (the step of development) of an IES. Complexity of components, their perplexity, and number of levels are not foreseeable in advance. This limitlessness could lead to IES's of unforeseeable power which could exceed the today imagined or expected possibilities.

26. The Formal Notion of Misunderstanding

In the realm of understanding, the problem of misunderstanding can arise as an informational difference within understanding among two or more parallel processes of understanding. If α is information which comes in understanding consideration of two processes of understanding, say $\mathfrak{A}$ and $\mathfrak{B}$, then the resulting meanings $\mu_{\mathfrak{A}}(\alpha)$ and $\mu_{\mathfrak{B}}(\alpha)$ of α can be incompatible in an informationally understanding way, that is, the informational (understanding) difference $\delta(\mu_{\mathfrak{A}}(\alpha), \mu_{\mathfrak{B}}(\alpha))$ becomes an informationally distinctive (sensible, observable, perceivable, conceivable, and comprehensible) entity.

The distinction in understanding of α arises out of the understanding processes, for instance,

($\mathfrak{M}1$) $((\alpha, \mu_{\mathfrak{A}}, \mu_{\mathfrak{B}} \models \mathfrak{A}) \models \mu_{\mathfrak{A}}) \models \delta_{\mathfrak{A}}(\mu_{\mathfrak{A}}, \mu_{\mathfrak{B}}));$
($\mathfrak{M}2$) $((\alpha, \mu_{\mathfrak{B}}, \mu_{\mathfrak{A}} \models \mathfrak{B}) \models \mu_{\mathfrak{B}}) \models \delta_{\mathfrak{B}}(\mu_{\mathfrak{B}}, \mu_{\mathfrak{A}}))$

where $\mu_{\mathfrak{A}} = \mu_{\mathfrak{A}}(\alpha)$, $\mu_{\mathfrak{B}} = \mu_{\mathfrak{B}}(\alpha)$, $\delta_{\mathfrak{A}}(\mu_{\mathfrak{A}}, \mu_{\mathfrak{B}}) = \delta_{\mathfrak{A}}(\mu_{\mathfrak{A}}(\alpha), \mu_{\mathfrak{B}}(\alpha))$, and $\delta_{\mathfrak{B}}(\mu_{\mathfrak{B}}, \mu_{\mathfrak{A}}) = \delta_{\mathfrak{B}}(\mu_{\mathfrak{B}}(\alpha), \mu_{\mathfrak{A}}(\alpha))$. We see how formulas ($\mathfrak{M}1$) and ($\mathfrak{M}2$) are cross-correlated in the production of informational differences $\delta_{\mathfrak{A}}$ and $\delta_{\mathfrak{B}}$, in different ways of understanding of α. In the case of misunderstanding, differences $\delta_{\mathfrak{A}}$ and $\delta_{\mathfrak{B}}$

may informationally exclude each other, for instance by

$(\mathfrak{M}3)$ $((\delta_{\mathfrak{A}}, \delta_{\mathfrak{B}} \models \mathfrak{A}) \models \varepsilon_{\mathfrak{A}}$;
$(\mathfrak{M}4)$ $((\delta_{\mathfrak{B}}, \delta_{\mathfrak{A}} \models \mathfrak{B}) \models \varepsilon_{\mathfrak{B}}$

where the difference reports $\varepsilon_{\mathfrak{A}}$ and $\varepsilon_{\mathfrak{B}}$ concerning α are unacceptable for $\mathfrak{A}$, $\mathfrak{B}$, or both of them. Thus, these reports do not impact the processes of understanding in a proper way, that is,

$(\mathfrak{M}5)$ $\varepsilon_{\mathfrak{B}}(\alpha) \not\models \mathfrak{A}$; $\varepsilon_{\mathfrak{A}}(\alpha) \not\models \mathfrak{B}$

Thus, misunderstanding of other understanding of information may function in the way of unacceptable informational difference or, in an extreme case, as informational noise.

27. The Formal Expression of Disorders of Understanding

What is the difference between misunderstanding and disorder of understanding? In medical sense, disorder can point into the domain of the pathologic. If misunderstanding performs as a characteristic, distinct understanding, then disorder may fail in its own understanding, recognizing, for instance, its own confusion, faulty, and inconsistent understanding.

A disorder of understanding can be sensed by the understanding itself as well as by other kinds of understanding. A disordered meaning of information α is produced, for instance, by

(D1) $(\alpha, d \models \mathfrak{D}) \models d$

where $\mathfrak{D}$ marks the disordered understanding within an understanding system $\mathfrak{A}$ and d is the produced disordered meaning of α. Thus,

(D2) $(((\alpha, d \models \mathfrak{D}) \models d) \subset ((\alpha, \mu \models \mathfrak{A}) \models \mu)) \Rightarrow \delta(\mu, d)$

where the informational difference $\delta(\mu, d)$ between the regular meaning μ and the disordered meaning d can be observed.

28. Formalization of Cultural Understanding

Culture as information of understanding roots in cultural media (social information), communication (transient information), and historical memory (preceding information), that is, in cult, myth, worship, belief, knowledge, and so forth. It projects (builds up and constructs) the fundamental information of the past (hermeneutics) for the being in the present, establishing the sacred, frequently untouchable belief and faith, which impact the behavior. Culture is accumulated understanding rooting in life, experience, and survival of the previous and present epochs, left (inherited) in forms, styles, habits, experiences, and so forth, constituting the tales of understanding of preceding epochs. This information essentially impacts the present, ruling and, in evolutionary terms, genetic information that governs the behavior of beings, populations, and arising of evolutionary changes (for instance, in the mind-brain domain).

In general, culture as understanding is arising of intellectual, aesthetic, and moral information with expert care and training. As social information, it concerns the integrated pattern of knowledge, belief, and behavior that is impacted by human and populational capability or incapability of survival. As historical memory, culture informs the succeeding generations, but not unconditionally, depending on the instantaneous circumstances of surviving.

Cult, as the focusing component of culture as information and of culture as information of understanding of living beings and their populations, concerns care (Being) and adoration. Cult as information is, for instance, a great devotion to idea (ideology), intellectual fad (culturing or "culturism"), or intellectual movement (intellectualism). In Latin, cultus means also the way of life, that is, understanding and performing the life according to the instantaneously ruling understanding of culture.

How can culture as an informational phenomenon of understanding be formalized? Which are the cultural components and how do they constitute informationally the cultural cycle of understanding?

Let understanding $\mathfrak{A}$ of culture C as informing produce (sense, observe, perceive, conceive, comprehend) the cultural information c. Further, let us, for instance, introduce the following markers for further culturally componential or subcultural informational entities:

C^+ marks cultivating and c^+ marks cult;
M marks mythologizing and m marks myth;
W marks worshiping (ritualizing) and w marks worship (the object of worshiping, for instance, ritual or ritualism);
B marks believing and b marks belief (ideology);
K marks knowing (scientizing, expertizing) and k marks knowledge (science, expertise).

Let us introduce the following rough (linear) hierarchy:

(C1) $((((((K, k) \subset (B, b)) \subset (W, w)) \subset (M, m)) \subset (C^+, c^+)) \subset (C, c)) \subset (\mathfrak{A}, \mu)$

This hierarchy is a nested structure where at the lowest level is knowledge, belief is above it, and both are beneath worship, etc. On the top is understanding, and beneath it is culture as understanding of cultural information, etc. Formula (C1) is only an example of linearly nested hierarchies. Of course, as any other informational entities, hierarchies can be structured circularly too, for instance,

(C2) $(((((((\mathfrak{A}, \mu) \models (K, k)) \subset (B, b)) \subset (W, w)) \subset (M, m)) \subset (C^+, c^+)) \subset (C, c)) \subset (\mathfrak{A}, \mu)$

where linear hierarchy (C1) is informed (that is, performed) circularly by $(\mathfrak{A}, \mu)$ at the top of this hierarchy. According to this cyclic hierarchy, the main informational cycle of

cultural understanding could be

(C3) $(((((((\mathfrak{A}, \mu) \models (K, k)) \models (B, b)) \models (W, w)) \models (M, m)) \models (C^+, c^+)) \models (C, c)) \models (\mathfrak{A}, \mu)$

To this cycle, at the first glance, formula

(C4) $(\mathfrak{A}, \mu, C, c, C^+, c^+, M, m, W, w, B, b, K, k) \Vdash (\mathfrak{A}, \mu, C, c, C^+, c^+, M, m, W, w, B, b, K, k)$

can be imagined that enables the choice of the most complex and perplexed, that is, parallel-cyclic informing among the active cultural components and their understanding. To the listed cultural components the general (basic) components of understanding (sensing, observing, perceiving, conceiving, comprehending) can be added establishing more and more complex system of cultural understanding.

29. Formalizing the Idea of God as Informational Understanding

> ... God is infinite intellect in which knower and known are identical. The distinction we draw between knower and known is transcended in God. The suspicion, drawn out into a thesis and strongly insisted on by some philosophers, that thought and object of thought must always be distinct, is one more product of our experience of finite thought; ...
>
> T. Gornall (POG) 77

God as information (GI) is the consequence of human historicity (historically embedded information), of biologic (autopoietically possible) and cultural evolution (culturally developmental information) of human's nervous system, of its brain and mind (cultural memory). Following T. Gornall (POG, 80-81) "... God's knowledge is all derived from Himself, never from any other source. ... God's knowledge is God's being, and His being is by definition a se, underived, its own explanation in every way. In knowing Himself, God knows all possibilities and all actuality, because these both depend directly on Him. ..."

Let us derive the following five axioms (on informational existence of God) which we will formalize in a straight-forward manner:

(1) God is the marker for supreme (the most universal) information.

(2) God is everywhere (a universal space-informational distributivity and parallelism).

(3) God knows and sees (understands and senses) all things (information).

(4) God has no beginning, He is, and He always will be (God as timeless information).

(5) God is creator (informer) of all things (information).

Let us formalize these axioms in the way of informational logic. Let god as information be marked by g and let god as informing of information g, that is, godding or deifying in a god-informational manner, be marked by G. Further, let $\forall$ mark informational operator with the meaning 'informs that all' and let '.' mark informational operator with the meaning 'inform in such a way that'.

1. God is the marker for supreme (the most universal) information. The kernel of this formula is that god is supreme information in regard to any information, that is,

(G1) $(\forall \alpha).(\alpha \subset g)$

This explicitly linear (or implicitly circular) formula reads as something informs all entities (operands) α that they (always) inform to g in a subordinated informational way or that g is informationally supreme to any α. We shall recognize later which kind of information could be the informer of this message which explicates the real nature of operator $\forall$, that is, its left side.

2. God is everywhere (a universal space-informational distributivity and parallelism). It means, with other words, god is informed from everywhere (everything) and it informs everywhere (to everything), or formally,

(G2) $(\forall \alpha).((\alpha \Vdash g); (g \Vdash \alpha))$

This explicitly linear formula says that each distinct α informs g and is informed by g in a parallel, that is simultaneous way. First, god g appears as an extremely, ultimately parallel receptor to which all existing and arising α's inform in parallel and simultaneously. But, simultaneously, god g impacts all existing and arising α's in a parallel, i.e., simultaneous informational way. It means that there does not exist an informational space, domain, or realm to which g does not inform continuously and from which g is not being continuously informed.

3. God knows and sees (understands and senses) all things (information). In short: god understands everything. Informing of god g is, in fact, its understanding G (godding, deifying), that is, informing of god g as information. In this respect, already components of G as understanding (its sensing $\mathfrak{S}$, observing $\mathfrak{S}^+$, perceiving $\mathfrak{P}$, conceiving $\mathfrak{C}$, and comprehending $\mathfrak{C}^+$) are supreme informational entities of G. God g senses everything means, for instance,

(G3) $(\forall \alpha).(\alpha \models (\mathfrak{S} \subset G))$

and, thus,

(G4) $(((((((((
[1] $(((\forall \alpha).(\alpha, g \models \mathfrak{S})) \models \sigma) \models$
[2] $\mathfrak{S}^+) \models \sigma^+) \models$
[3] $\mathfrak{P}) \models \pi) \models$
[4] $\mathfrak{C}) \models \gamma) \models$
[5] $\mathfrak{C}^+) \models \gamma^+) \models$
[6] $G) \models g$

In this way, god g appears as the meaning of godding G. The distinct phases of arising of g are explicated by [1], ... , [6] (sensing, observing, perceiving, conceiving, comprehending, understanding). This cyclic

scheme of understanding of g is hermeneutical.

4. God has no beginning, He is, and He always will be (God as timeless information). If god g has no beginning, we can put

(G5) $\not\models (\llcorner (G \models g))$

or a better determined form

(G6) $\not\models (g \llcorner (G \models g))$

or a completely determined formula

(G7) $g \not\models (g \llcorner (G \models g))$

If the question arises, who could begin the arising of god g, the answer is, probably god g by himself: $g \llcorner (G \models g)$. However, god g says that he has no beginning, thus, (G7).

This discussion brings to the surface another question: who else or what else is the informer in case $\models \alpha$ except α itself? And anti-symmetrically: who else or what else could be informed in case $\alpha \models$ except α itself? This question can particularly concern cases in which the so-called logical quantifiers $\forall$ and $\exists$ are used. Thus, some previous formulas, (G1), (G2), and (G3), respectively, can be "deified" in the following way:

(G8) $(g \forall \alpha).(\alpha \subset g)$;

(G9) $(g \forall \alpha).((\alpha \Vdash g); (g \Vdash \alpha))$;

(G10) $(g \forall \alpha).(\alpha \models (\mathfrak{G} \subset G))$

In this way, god g as information informs all "things" (informational entities) on his supremeness, "everywhereness", and universal understanding, respectively.

If god always will be, he has no end. This informational fact can be expressed in a godly expanding way by

(G11) $\not\models (\urcorner (G \models g))$ or
$\not\models (g \urcorner (G \models g))$ or
$g \not\models (g \urcorner (G \models g))$

where operator $\urcorner$ has the meaning 'ends' or 'is ended by'.

The question we have to answer is, how does god exist if he does not have a beginning. We see that god by itself informs that he does not have a beginning (G7) or more distinctly,

(G12) $g \not\models (\llcorner g)$

It means that g does not inform its beginning, however, he evidently exists, since he carries out its informing on its non-beginning and non-ending.

5. God is creator (informer) of all things (information). More exactly, god is co-creator of all things (information). This informational ability of him is hidden in (G2). While god as information is everywhere, he informs everything.

We see that the discourse on formal informational understanding of the idea of god is only a formal beginning. The aim of this discourse is to show how god as information can be made informationally regular, without any further non-informational speculation. This discourse shows how a formal informational language can enter into some critical informational domains, disclosing them informationally.

30. Conclusion of Part Two

What might be true for philosophers, if could it be true at all, "that thought and object of thought must always be distinct," does not endure the objections coming from the postmodern theory of information, of human informational common sense, experience, and evolution in understanding of human mind. Formally, an informational operand α is a uniqueness, that is, the duality of informationally active (operational) and informationally passive (operand) components. This fact comes to the surface in the process of decomposition of α, and is deterministically constituted by the basic property of any informational entity, which is in $\alpha \models$ and $\models \alpha$. In the same way, any informational operator is operator-operand dual, where this duality comes to the surface as the operator-operand decomposition of an informational operator. This understanding of information contributes also essentially to the problem of understanding as information.

Understanding as information brings to the surface the connectedness and perplexity of operand and operator processes. Things which exist (including information as a thing) inform and are informed simultaneously and circularly repeatedly) in a physical, chemical, biological, etc. or, in general, in an informational way, concerning mutually impacted things by themselves as well as their observers. And this becomes the central point of informational understanding of things, which is understanding as information.

Formalization brings to the surface not only the possibility of formal informational concepts, but can immensely change the knowledge of understanding as a basic and the most complex informational realm. Through formalization, understanding becomes a formal category (informational mode or modus of informing) that can inform spontaneously and circularly and in hermeneutic, that is, historically cyclic way and/or (simultaneously) in a (non-historical) perplexedly parallel, that is, a direct parallel-cyclical way (without the explicit hermeneutic circles). Understanding can always be formally decomposed into its various components, from sensing, observing, perceiving, conceiving, and comprehending which may inform in their own hermeneutic cycles to the informational phenomena of intelligence being controlled by different sorts of intellectualism.

Formal axioms of understanding can be developed in a straight-forward manner as a basic set of rules constituting a particular algebraic theory of understanding. Intelligence and intellectualism can be introduced as separate and/or distributed components (properties) pervading other components of understanding. Within this system of understanding, meaning arises as a relevant understanding information concerning some initial, input, or from the outward arriving information. It was shown how hermeneutics as a modus of understanding can be formalized in the form of historically all-embracing cycles of understanding.

Semiotic forms of information and their understanding deserve a separate and exhaustive formal informational treatment. For this purpose the notion of the so-called semiotic understanding can be introduced which, in some details, approaches the semiotic structure and organization of the input information being investigated by this type of understanding. Since the semiotic information constitutes the major part of our linguistic domain, the semiotic approach of understanding could substantially influence the design of informational systems concerning the semiotic nature of information (for instance, expert systems and other dictionary and language understanding systems).

An important aspect of contemporary living and artificial understanding systems is the consciousness of the existence of various informational phenomena of loseableness, losing, blindness, and breakdown within informational systems. For instance, it was shown how distinct forms of loseableness and losing of understanding can be classified and introduced into the game of understanding. On the other hand, a form of the blindness of understanding can become evident, so it can be broken down in the form of a relieving information impacting the dismission of the ruling informational blindness. The richness of the Heideggerian concepts of loseableness and losing of understanding can form the base from which particular formal cases of these phenomena can be conceptualized and adequately introduced into the design of living and artificial understanding system.

Misunderstanding can be treated as a strange, unknown, inconsistent, or unacceptable way of producing the meaning, a sort of counter-understanding which does not fit the convention or intelligence of the ruling understanding. Thus, misunderstanding can be on the way to counter-understanding which has to be informationally embedded into understanding. Understanding and misunderstanding are two poles of the perceiving and conceiving in the process of understanding. Therefore the produced meanings can be in a considerable semantical and pragmatical dissonance. On the other hand, understanding and misunderstanding can mutually reveal loseableness and losing of each other and point to the informational blindness and in this way perform as breakdown processes to each other.

Understanding disorder can be understood as the understanding malfunction on the level of autopoietic systems which can threaten the survival of an autopoietic system. In many cases, understanding disorder is classified in the sense of the pathologic. On the other hand, disordered understanding can be a part of the biological evolution taking place in a biological species.

Cultural information can be characterized by specific knowledge, belief, worship, myth, and cult within which it informs spontaneously and circularly. To these informational components, cultural information informs culturally in the sense of sensing, observing, perceiving, conceiving, and comprehending of cultural information. Cultural information may propose and dispose specific intelligence and intellectualism being distributed and dwelling within the cultural information itself. In this way, culture can preserve and stimulate the ongoing cultic, sacred, and cultivating style of informing.

The idea of God belongs to the oldest types of cultural information. This essay offers only a first trial for formalization of this idea by informational algebraic means. This trial of logical formalization is certainly not the first and the last one. God is a living proof of the existence of the most universal information within human mind.

Formalization of understanding as information is on the way to an informational theory of understanding in the living and the artificial. By this approach, probably, also the artificial understanding can come to its new enlightenment and perspective. This formal approach can also prepare the ground for various philosophical excursions of the possible in the domain of understanding.

References

(POG) Gornall, T., A Philosophy of God (The Elements of Thomist Natural Theology), Sheed and Ward, New York 1962.

(RCP) Maddy, P., The Roots of Contemporary Platonism, The Journal of Symbolic Logic 54 (1989) 1121-1144.

(EPD) Heidegger, M., Das Ende der Philosophie und die Aufgabe des Denkens, Zur Sache des Denkens, Niemeyer, Tuebingen 1969.

(PBA) Varela, F.J., Principles of Biological Autonomy, North Holland, New York (1979).

(POI) Železnikar, A.P., Principles of Information, Cybernetica 31 (1988), 99-122.

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

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

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

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

(IIA) Železnikar, A.P., An Introduction to Informational Algebra, Informatica 14 (1990) 1, 7-28.

(UAI) Železnikar, A.P., Understanding as Information, Part One, Informatica 14 (1990) 3, 8-30.

(GI) Železnikar, A.P., God as Information, to appear in A.P. Železnikar, On the Way to Information, Ljubljana, 1991.

THE RELATION SCHEME EXTENSIONS[1]

INFORMATICA 4/90

Svetlana Kuzmanov
Faculty of Economics,
Subotica, Moše Pijade 9
Pavle Mogin
Faculty of Technical Science,
Novi Sad, Veljka Vlahovića 3

Keywords: database scheme, design, relation scheme, extension, functional dependency, projection, instance.

ABSTRACT. A formal technique for the solution of the problem whether relation scheme (R',F') extends relation scheme (R,F) is presented in the paper. It is done by introducing the notion of EXT - suspected relation scheme and by using special recursive Horn - clauses. The results are expressed in the theorems which give sufficient conditions, defined on the level of attributes and functional dependencies, for the statement that (R',F') extends (R,F). Besides theoretical, the practical significance of the presented results in the database scheme design and transformation processes is expected.

SADRZAJ. U radu je data osnova za definisanje formalne tehnike resavanja problema da li sema relacije (R',F') prosiruje semu (R,F). Ovo je uradjeno uvodjenjem pojma EXT-suspektne seme relacije i koriscenjem specijalnih rekurzivnih Horn-ovih klauzula. Rezultati su izrazeni putem teorema koje daju dovoljne uslove, definisane na nivou obelezja i funkcionalnih zavisnosti, za tvrdjenje da (R',F') prosiruje (R,F). Pored teorijskog, ocekuje se i prakticni znacaj prezentiranih rezultata u projektovanju i procesu transformacije sema relacionih baza podataka.

1. INTRODUCTION

The concept of relation scheme extension within relational database theory was considered by Sciore E., Beeri C. and Kifer M., and Mogin P. and Kuzmanov S.. The idea is originated by E. Sciore who considered the relation schemes extensions in the transformation context of a dependent into an independent database (db) scheme, where the allowed set of dependencies contains first only functional dependencies (fd) and then multivalued dependencies (mvd) [Sc]. This idea was further developed in [BeKi1] and [BeKi2], where it was considered in the presence of mvd's from the aspect of transformation of a scheme possessing the intersection property, into an acyclic scheme, by adding new attributes, fd's and mvd's to the initially defined attribute and dependency sets. In the papers of Beeri and Kifer, the concept of extension is discussed in the scope necessary for the complete solution of the above mentioned transformation. The definition of relation scheme extension is introduced in [Sc] through a more general term of the two relational schemes compatibility, there is a number of features regarding compatibility and extension proved in the paper and implicit definition of the db scheme extension given. The

1 Research presented in the paper was supported by the Science Council of SAP Vojvodina

explicit db scheme extension definition was given in [KuMo1] and some related properties proved in the context of transformation of a not independent into an extended independent db scheme. Concerning the concept of extension, there remained, however, a number of open problems [Sc].

The notions of relation schemes extensions and compatibility in the presence of fd's are closely related to the questions of fd family closeness under the projection and join operators, respectively. In fact, the equivalence of corresponding definitions can be easily proved. Above questions were considered in [GiZa] and [Hu]. One of typical results in [GiZa] is a characterization for when the fd families are closed under projection and join. Analogous problem for more general class of implicational dependency families were studied in [Hu]. But it remains open whether this results are suitable enough for practical applications.

In this paper a formal technique for easy checking whether one of the given relation schemes extends the other is presented. The result is expressed through sufficient condition on the level of attributes and fd's for the statement that (R',F') extends (R,F). Such technique may be of crucial importance in the db scheme design and transformation processes and in the integrity- checking mechanisms.

The paper consists of five sections including this Introduction and Conclusion.

Section 2 contains notes on terminology and basic definitions. The concept of the relation scheme extension from the aspect of its role in the relational db theory is discussed in Section 3. The main result of the paper is given in Section 4. The concept of EXT - suspected relation scheme (R',F') with respect to the scheme (R,F) is introduced there. The set of the EXT - suspected models $M=\{M_l \mid l=1,p\}$ is assigned to the EXT - suspected scheme (R',F'). By means of recursive Horn - clauses the attribute sets P_0 and P_K are defined. All this provides the basis for defining conditions over the relation schemes (R',F') and (R,F), on the level of attributes and fd's, to conclude whether (R',F') extends (R,F) or not. The theorem T1 gives the solution for the special case (p=1), while the main result is contained in the theorem T2, providing sufficient condition over set of fd's F for the statement about extension. The final section summarizes previous considerations.

2. BASIC CONCEPTS AND TERMINOLOGY

A basic acquaintance with relational database theory as in [Ull] and predicate calculus on the level of [Sh] is assumed. All necessary definitions follow.

Relational scheme (R,F) consists of the finite attribute set R and dependency set F, where only fd's are allowed. Each attribute from R will be assumed to be domen infinite [Hu]. $F|_Z$ denotes the restriction of fd set F^+ to the attribute set Z. SAT(R,F) denotes the set of all relations satisfying every fd in F.

The consideration of relation "is extension" between two relation schemes is the main topic of the paper. Unformally, the relation scheme (R',F') "is extension" of the scheme (R,F) if $R' \supset R$ and every relation $r' \in SAT(R',F')$ contains one relation $r \in SAT(R,F)$. For each relation $r \in SAT(R,F)$ there must be also found at least one relation $r' \in SAT(R',F')$ which contains all data from r.

Beside the relation "is extension", between two relation schemes the relation "is compatible" can be also defined in the similar sense but it is out of the scope of this paper.

Definition D1. The relation scheme (R',F') is said to be the extension of the scheme (R,F) in the notation $(R',F') \supset (R,F)$ if:

$$R \subset R', \tag{2.1}$$

$$(\forall r \in SAT(R,F))(\exists r' \in SAT(R',F'))(r=\Pi_R(r')) \tag{2.2}$$

and

$$(\forall r' \in SAT(R',F'))(\exists r \in SAT(R,F))(r=\Pi_R(r')). \tag{2.3}$$

3. THE ROLE OF RELATION SCHEME EXTENSION CONCEPT IN RELATIONAL DATABASE THEORY

According to the stated in the Introduction, the term db scheme extension was introduced with purpose to support transformation theory of the initially given schemes with some undesirable property into schemes without them. Namely, the transformation procedure of the given scheme was formulated and it was checked whether it results in the db scheme which is the extension of the given one [Sc].

· Beside the previous, the concept of extension may play some other roles in answering the question whether, in general, one db scheme can be replaced by another one, without loss of information from any instance of the replaced scheme.

The above mentioned roles of the extension concept are illustrated by the following examples.

Example 3.1. Consider the relational scheme (R,F) with attributes C(ourse), D(ept), F(aculty), R(oom), and F = { C → D, F → D, FC → R}. Courses are offered by only one department, each faculty member teaches a course in a single room. This is a well known not independent scheme [Sc]. By adding attributes and fd's it can be transformed into an independent scheme (R',F'). New attribute is N(umber), and F'={DN → C,C → DN,F → D,NF → R}. To prove that the transformation is correct, (R',F')⊃(R,F) should be shown. Possible solutions are given in [Sc] and [Ku].

Example 3.2. Relation schemes (R,F) and (R',F') are considered, where R=ABCDIJMNE, F={AB → E, CD → E, IJ → E, MN → E}, R'=ABCDIJMNEGHK, F'={ACI → G, JM → G, BDJ → G, AJM → H, CIN → H, M → K, BGHK → E, AB → E, CD → E, IJ → E, MN → E}. It is apparent that R⊂R' holds. The question arises whether each relation r∈SAT(R,F) can be replaced by some r'∈SAT(R',F'). This is possible if (R',F')⊃(R,F) holds! The problem can be solved by direct application of the definition D1 for each particular case (instance), or by the use of general, formal technique based on the checking of condition on the higher abstraction level, that is, on the level of attributes and fd's. The first approach means the work with instances which leads to the time complexity issue and cannot be considered satisfying. In this example the use of D1 leads to the conclusion that ¬((R',F')⊃(R,F)), because there exists the relation r∈SAT(R,F), Figure 3.1, such that the relation r'∈SAT(R',F'), where $r=\Pi_R(r')$ does not exist . This can be shown by the extension of the relation r tuples by values for G,H,K from R'\R. Regardless of the way the values for the attributes from R'\R are chosen, the application of CHASE $_{F'}$ algorithm on the relation r' results in $r \neq \Pi_R(r')$.□

r =

A	B	C	D	E	I	J	M	N
3	1	3	0	1	3	6	2	0
3	0	3	0	2	3	4	4	0
0	1	0	5	3	0	4	4	0
3	0	7	0	4	8	6	2	8
0	1	7	5	4	8	4	2	9

Figure 3.1

The complexity of the problem solving approach in given examples at the level of instances points to the possible advantages of a formal determinant technique, as an alternative. Such approach may prove its value in the application of non-standard design procedures of the relational db schemes. By non-standard design procedure it is thereby understood a procedure which supports modification of the initially defined attribute and dependency sets aiming at achieving desirable db scheme properties (like independence, acyclity). At designing very large db schemes, it seems very logical to design mutually independent parts and merge them afterwards into a conceptual db scheme. This merging of independently designed parts, which can be conditionally considered as separate db schemes, leads to the problem of their compatibility. We believe that this problem could be solved by using the solution of the extension problem. The above mentioned motivates further coping with still open problems regarding relation scheme extension and compatibility. The following section offers a solution of the sufficient condition at the level of attributes and fd's for the statement that one relation scheme extends the other.

4. THE RELATION SCHEME EXTENSION DECISION PROBLEM

This section provides a basis for the defining a formal technique for the relation scheme extension problem solution on the level of attributes and fd's. Main theorem (T2) contains the most important result of the relation scheme extension problem, giving sufficient conditions for the statement (R',F')⊃(R,F). First, there follow lemmas and definitions of terms related to the notion of EXT suspected relation scheme model, and then formulations and proofs of the extension theorems.

Let (R,F) and (R',F') be two relation schemes satisfying the following initial assumptions:

$$R \subset R' \tag{4.1}$$

and,

$$F \subseteq F'|_R \,. \tag{4.2}$$

According to the assumption (4.1) the relation scheme extension deserves attention only in the case when the attribute set of one relation scheme presents the real subset of the other relation scheme attribute set.

If $\neg(F \leq F'|_R)$, then according to the definition D1 one can easily conclude that $\neg((R',F') \supset (R,F))$ because $\neg(2.3)$ is equivalent to $\neg(4.2)$. So the assumption (4.2) eliminates that case from further consideration.

Lemma L1. Let (R',F') and (R,F) be two relation schemes and let (4.1) and (4.2) hold. The statement

$$\neg((R',F') \supset (R,F)) \qquad (4.3)$$

is true iff

$$(\exists r \in SAT(R,F))(\forall r' \in SAT(R',F'))(r \neq \Pi_R(r')) \qquad (4.4)$$

holds.

Proof. The proof of lemma follows immediately.□

The meaning of the lemma L1 is to point out the fact that to prove (4.3) it is necessary and sufficient to show that (4.4) holds, reducing the problem to the existence of a counter example for the statement $(R',F') \supset (R,F)$. If such a counter example exists, one can conclude that the statement (4.3) is correct.

Lemma L2. Let (R,F) and (R',F') be two relation schemes satisfying (4.1) and (4.2). If

$$\neg(\exists Z \subseteq R' \setminus R)(\exists Y,V,X \subseteq R)(Z \subseteq (Y)^+_{F'} \wedge V \subseteq (XZ)^+_{F'}) \qquad (4.5)$$

for $Z, Y, V \neq \emptyset$, then

$$(R^{\downarrow},F') \supset (R,F)$$

holds.

Proof. To prove the statement of lemma, with respect to (4.1) and (4.2) it is necessary to show that

$$(\forall r \in SAT(R,F))(\exists r' \in SAT(R',F'))\ (r = \Pi_R(r')) \qquad (4.6)$$

is satisfied.

Let the assumption (4.5) holds, and (4.6) not, namely, let

$$(\exists r \in SAT(R,F))(\forall r' \in SAT(R',F'))\ (r \neq \Pi_R(r')). \qquad (4.7)$$

The conclusion on contradiction is made by consideration of the relation $r=\{t_i \mid i=1,n\} \in SAT(R,F)$ and r'' over $R^{\downarrow}$ defined by

$$r''=\left\{t' \mid \varphi(t,t')\right\}, \qquad (4.8)$$

where the predicate formula φ is defined as follows

$$\varphi(t,t')=(\forall t \in r)\ (\exists t' \in r'')\ (t=\Pi_R(t')) \wedge (\forall t' \in r'')\ (\exists t \in r)\ (t=\Pi_R(t')) \qquad (4.9)$$

where

$$(\forall t'_i,t'_j \in r'', i \neq j)\ (t'_i(R' \setminus R) \neq t'_j(R' \setminus R)$$

holds.

Let $r'=CHASE_F,(r'')$. Regarding the assumption (4.7), the application of the $CHASE_F$, algorithm to r" causes the change of a value of some attribute from R. Let the attribute which changed its value be $V \subseteq R$ where $V \neq \emptyset$. In the other words

$$(\exists t'^{*}_i \in r')(\exists t_i \in r)\ (t_i = t'_i[R] \wedge t_i[V] \neq t'^{*}_i[V]) \qquad (4.10)$$

must hold.

Note that if the number of tuples in the relation r' is less than the number in relation r", that effect might have been caused only by the fact that during the application of CHASE some attribute from R changed its value.

Generally, for the change of V value there must exist an fd $W \rightarrow V$ in $(F')^+$. That fd can not belong to F^+, otherwise $r \in SAT(R,F)$ would not hold. Hence $\neg(W \subseteq R)$. Let $W=XZ$, where $X \subseteq R$, $Z \subseteq R' \setminus R$ and $Z \neq \emptyset$. According (4.9) and (4.10), during the application of $CHASE_F$, to r", the attribute Z also changed its value, namely $t'_i[Z] \neq t'^{*}_i[Z]$. This change could be caused only if there was an fd $Y \rightarrow Z$ in $(F')^+$, where $Y \subseteq R$, $Y \neq \emptyset$. From this, one can conclude that

$$(\exists Z \subseteq R' \setminus R)\ (\exists X,Y,V \subseteq R)\ (Z \subseteq (Y)^+_{F'} \wedge V \subseteq (XZ)^+_{F'})$$

holds, which is contradictory to (4.5), thus proving the lemma.□

Example 4.1. Let the relation schemes (R,F), $R=ABCDE$, $F=\{A \rightarrow C,\ B \rightarrow C,\ DE \rightarrow A\}$, and (R',F'), $R'=ABCDEG$, $F'=\{A \rightarrow C,\ B \rightarrow C,\ DE \rightarrow A,\ B \rightarrow G\}$ be given. According to the condition (4.5), the statement $(R',F') \supset (R,F)$ is true because there exist $Z=G$ and $Y=B$ such that $G \subseteq B^+_{F'}$, but there is no V in R such that $V \subseteq (XZ)^+_{F'}$ holds.□

Example 4.2. Let the relation schemes (R,F), $R=ABMLKI$, $F=\{AB \rightarrow M,\ M \rightarrow L,\ LK \rightarrow I\}$ and (R',F'), $R'=ABMLKIG$, $F'=\{AB \rightarrow M,\ M \rightarrow L,\ LK \rightarrow I,\ IG \rightarrow A\}$ be given. There, as well, with respect to (4.5) one can conclude that $(R',F') \supset (R,F)$ holds, as for $X=I$, $Z=G$ and $V=A$ where $A \subseteq (IG)^+_{F'}$, there is no Y such that $G \subseteq (Y)^+_{F'}$ holds.□

Definition D2. The relation scheme (R',F') is EXT - suspected with respect to the scheme (R,F) if the assumptions (4.1) and (4.2) are satisfied and if

$$(\exists Z\subseteq R'\setminus R)(\exists Y,V,X\subseteq R)\ (Z\subseteq(Y)^{+}_{F'}\wedge V\subseteq(XZ)^{+}_{F'}) \quad (4.11)$$

for $Z,Y,V\neq\emptyset$ holds.

Obviously, (4.11) is equivalent to the negation of (4.5). Lemma L2 provides minimal necessary condition to doubt that (R',F') extends (R,F). In further, it will be assumed that $\neg(Z\subseteq X^{+}_{F'})$, and if $X\neq 0$, $\neg(X\subseteq Z^{+}_{F'})$, and that all fd's belong to the nonredundant covering which is left reduced. □

Definition D3. The set of EXT - suspected models $M=\{M_l \mid l=1,p\}$ is assigned to the EXT - suspected scheme (R',F') with respect to (w.r.t) (R,F), where

$$M_l = (XUVZ,\ G'),\ G' = F'|_{XUVZ},\ G = F'|_{XUV}$$

and the following holds:

$$(Z\subseteq R'\setminus R)\ (Z=\{Z_k \mid k=1,m\}), \quad (4.12)$$

$$(\forall Z_k\subseteq Z)\quad (\exists Y_k\subseteq P(R))\ ^{2}$$

$$(Y_k=\{Y^i_k \mid Y^i_k \rightarrow Z_k \in G',\ i=1,n_k\}), \quad (4.13)$$

and

$$(V\subseteq R)\ (XZ \rightarrow V \in G'), \quad (4.14)$$

with

$$U=\bigcup_{k=1}^{m}\ \bigcup_{i=1}^{n_k} Y^i_k\ .$$

Due to the simplicity, the following notations are introduced

$$Y=\{Y_k \mid Y_k\in P(R)\wedge k=1,m\};$$

$$Q=\{Y_k \mid Y_k\in Y\wedge n_k>1\},\ \#Q=q,$$

and

$$J=\bigcup_{Y_k\in Y\setminus Q} Y_k\ . \quad (4.15)$$

There follow definitions of the sets P_0 and P_k, $k=1,q$ which will be used in proving the theorems T1 and T2. The above mentioned sets are defined by means of recursive Horn clauses.

The recursive Horn clause is the first order calculus predicate formula equivalent to the formula

$$R_1 \wedge R_2 \wedge \ldots \wedge R_n \Rightarrow R,$$

where the formula appearing in consequent appears at least once in the antecedent. The formula is bounded if there is at least one finite derivation $R_1,R_2,\ldots,R_n\vdash R$, $n\in N$. Formula R is closed if no variable is free in R. Only the closed formulae are used in the paper, meaning that all relevant theorems are valid.

The first order calculus which is the basis for Horn clauses definition in the paper is the predicate calculus with equality. The additional quantificator $(\exists!u)$ for the variable u is introduced into it in the following way:

$(\exists!u)A(u)$ is the replacement for
$(\exists u)A(u)\wedge(A(u)\wedge A(v) \Rightarrow u=v)$,

where A is an arbitrary formula. In the paper, the quantificator "!" is used with the meaning "not more than one" defined in the following way:

$(!u)$ A(u) is the replacement for
$(\exists!u)A(u) \vee (\forall u)\neg A(u)$.

Definition D4. The formula $\phi(A,k,i)$ over the domains of its arguments R, $\{1,\ldots,q\}$, N, respectively, is defined in the following way:

$$\phi(A,k,i)=$$

$$(A\in(XJP_o^{i-1})^{+}_{F}\wedge(!Y^j_k\in Y_k)\neg(A\in(P_k^{i-1}Y^j_k)^{+}_{F}) \quad (4.16)$$

where

$$(\forall k\in\{1,\ldots,q\})(P^0_k=\emptyset)\ \text{and}\ P^0_0=(XJ)^{+}_{F}\ , \quad (4.17)$$

$$P^i_k=\{A \mid \phi(A,k,i)\}\ .\ \square \quad (4.18)$$

Definition D5. The formula $\gamma(B,i)$ over the domains of its arguments R and N, respectively, is defined in the following way:

$$\gamma(B,i)=$$

$$(\exists Y_h\in Q)(\forall Y^j_h\in Y_h)(B\in(P_h^{i-1}Y^j_h)^{+}_{F}), \quad (4.19)$$

where

$$P^i_o=\{B \mid \gamma(B,i)\}\ .\ \square \quad (4.20)$$

The following recursive Horn clauses are considered

$$\phi(A,k,p) \Rightarrow \phi(A,k,q) \quad (4.21)$$

and

$$\gamma(B,p) \Rightarrow \gamma(B,q),\ \text{where}\ p<q. \quad (4.22)$$

Lemma L3. Recursive Horn clauses (4.21) and (4.22) are bounded.

Proof. Can be found in [Ku]. □

[2] The partitive set of the set R is denoted by $P(R)$.

The direct consequence of this lemma is:

$$(\forall l \in \{0,1,\ldots,q\})(\exists n \in N)\ (P_l^n = P_l^{n+1} = \ldots = P_l). \quad (4.23)$$

The sets P_0 and P_k, k=1,q are used in the formulation and proof of theorem T1.

Example 4.3. Let us consider the relation schemes (R,G) and (R',G') where

R=ABCDEFGIJKLV,
G={CE → V, BDF → V, GI → V, AGJ → V,
AGL → V, E → A, F → A. CI → A, E → B,
AG → B, DJ → B, AG → C, BF → C, J → C,
AG → D, CE → D, I → D, BF → L, BI → L,
CE → L, AJ → L, FL → I, AGL → I,
EL → J},
R'=ABCDEFGIJKLV$Z_1Z_2Z_3$,
G'={ABCD$Z_1Z_2Z_3$ → V, E → A, F → A, CI → A,
E → B, AF → B, DJ → B, J → C, AF → C,
BG → C, I → D, AF → D, CE → D, E → Z_1,
F → Z_1, G → Z_1, I → Z_2, J → Z_2, K → Z_3,
L → Z_3, BG → L, CE → L, BI → L, AJ → L,
FL → I, AGL → I, EL → J}.

Therefrom X=ABCD, Y_1= {E,F,G}, Y_2= {I,J} and Y_3={K,L}. By applying D4 and D5 the inferring sequence showed in Figure 4.1 is obtained.

1.	P_0^0={A,B,C,D}	P_1^0={ }	P_2^0={ }	P_3^0={ }
2.	P_0^1={ }	P_0^1={A}	P_2^1={C,D}	P_3^1={ }
3.	P_0^2={A,C,D}	P_1^2={A,B}	P_2^2={C,D,A,B}	P_3^2={ }
4.	P_0^3={A,B,C,D,L}	P_1^3={A,B,C}	P_2^3={C,D,A,B}.	P_3^3={ }
5.	P_0^4={A,B,C,D,L}	P_1^4={A,B,C,D,L}	P_2^4={A,B,C,D,L}	P_3^4={ }
6.	P_0^5={A,B,C,D,L,V}	P_1^5={A,B,C,D,L}	P_2^5={A,B,C,D,L}	P_3^5={ }
7.	P_0^6={A,B,C,D,L,V}	P_1^6={A,B,C,D,L,V}	P_2^6={A,B,C,D,L}	P_3^6= { }

P_0={A,B,C,D,L,V}, P_1={A,B,C,D,L,V},
P_2={A,B,C,D,L} P_3={ }.

Figure 4.1.

For the simple example where R=XABV, G={XA →V, XB →V}, R'=XABVG, G'={A →G, B →G, XG →V}, it is rather simple to conclude, by using the definition D1, that (R,G) c (R',G') does not hold. With respect to the problem complexity, it is almost impossible to get a definite answer in the similar way for the example 4.3. The results given below, by means of the theorems T1 and T2, make a formal basis for the solution of the problem whether (R,F) c (R',F') holds, thus simplifying it.

The theorem T1 deals with the special case where p from D3 equals to one.

Theorem T1. Let (R',G') be EXT - suspected relation scheme with respect to the scheme (R,G), where R=XUV and R'=RZ. The statement

$$(R',G') > (R,G) \quad (4.24)$$

is true if

$$V \subseteq (P_0 XJ)_G^+ \quad (4.25)$$

holds.

Proof. Let (4.25) hold, and (4.24) not. According to the statement of the lemma L1, ¬(4.24) implies

$$(\exists r \in SAT(R,G))(\forall r' \in SAT(R',G')\ (r \neq \Pi_R(r')) \quad (4.26)$$

Let the relation $r=\{u_i \mid i=1,n\}$ such that $r \in SAT(R,G)$ be considered. The fact that for the given relation r holds

$$(\forall r' \in SAT(R',G'))\ (r \neq \Pi_R(r') \quad (4.27)$$

will be represented, without the loss of generality, by introducing relation r" over R' which is constructed according to (4.8) and (4.9).

For $r' = CHASE_F(r")$ holds

$$r \neq \pi_R(r'). \quad (4.28)$$

Statement (4.28) implies

$$(\exists u_i, u_j \in r)\ (u_i[V] \neq u_j[V]) \text{ and}$$

$$(\exists u_i', u_j' \in r',\ u_i' \neq u_j')$$

$$(u_i'[R\backslash V] = u_i[R\backslash V] \wedge u_j'[R\backslash V] = u_j[R\backslash V])$$

and

$$u_i'[V] = u_j'[V]. \quad (4.29)$$

According to (4.13)

$$u_i'[XZ]=u_j'[XZ]\ . \qquad (4.30)$$

As $\neg(Z \subseteq X_{G'}^{+})$, and with respect to (4.13), (4.9) and (4.30) it folows that $(\forall Z_k \in Z)$ holds

$$(\exists Y_k^l \subseteq Y_k)\ (u_i[Y_k^l]=u_j[Y_k^l]) \qquad (4.31)$$

or

$$(\exists Y_k^1, Y_k^2, \ldots, Y_k^l \in Y_k,\ 1<l\leq n_k)$$

with

$$(u_i[Y_k^1]=u_{i_1}[Y_k^1], u_{i_1}[Y_k^2]=u_{i_2}[Y_k^2], \ldots,$$

$$u_{i_{l-1}}[Y_k^l]=u_j[Y_k^l]$$

$$u_i \neq u_{i_1};\ u_{i_n} \neq u_{i_{n+1}};\ u_{i_{l-1}} \neq u_j;$$

$$n=1,2,\ldots,l-2). \qquad (4.32)$$

As $\neg(X \subseteq Z_{G'}^{+})$, (4.30) implies

$$u_i[X]=u_j[X]. \qquad (4.33)$$

Let

$$L=\{Y_k \in Y \mid (\exists Y_k^l \in Y_k)\ (u_i[Y_k^l]=u_j[Y_k^l]), \qquad (4.34)$$

i.e. L is the set of Y_k's for which (4.31) holds. Note that for every Y_k such that $n_k=1, Y_k \in L$ must hold and so for the set J from (4.15) holds

$$u_i[J]=u_j[J]\ . \qquad (4.35)$$

Regarding the P_0 definition, (4.31) and (4.32) there follows

$$u_i[P_0] \leftrightarrows u_j[P_0], \qquad (4.36)$$

so that the conditions (4.33), (4.34), (4.35) and (4.36) imply

$$u_i[XJP_0Y_1^{i_1}\ldots Y_k^{i_k}]=u_j[XJP_0Y_1^{i_1}\ldots Y_k^{i_k}]\ , \qquad (4.37)$$

where $\{Y_1,\ldots,Y_k\}=L\setminus\{Y_h \mid n_h=1\}$.

With respect to (4.27) and (4.37) there follows

$$\neg(V \subseteq (XJP_0Y_1^{i_1}\ldots Y_h^{i_h})_G^{+}\ , \qquad (4.38)$$

which further implies

$$\neg(V \subseteq (XJP_0)_G^{+}\ . \qquad (4.39)$$

The statement (4.39) contradicts to the initial assumption, which proves the theorem.□

Example 4.4. Let (R',G') and (R,G) be given as in the example 4.3. We have directly that (4.25) holds and therefore the statement $(R,G) \subset (R',G')$ is valid.□

The generalization of theorem T1 is the following theorem.

Theorem T2. Let (R',F') EXT - suspected relation scheme with respect to (R,F) be. The sufficient condition for the statement

$$(R',F') \supset (R,F) \qquad (4.40)$$

is that $(\forall M_I \in M)$ holds

$$V \subseteq (P_0XJ)_F^{+}\ . \qquad (4.41)$$

Proof. Similar to the proof of the previous theorem.□

5. CONCLUSION

The paper provides sufficient condition for the statement $(R',F') \supset (R,F)$ i. e. the relation scheme (R',F') extends the scheme (R,F) with respect to the set of fd's. The condition is defined over the set of models, assigned to the EXT - suspected scheme (R',F') by means of the sets P_0 and P_k determined by the recursive Horn clauses. Beside theoretical, the result may have practical significance, making a basis for determining formal technique which would be used in the steps of relational db scheme design methodology for instance, in the transformation process of the initially defined db scheme into a new better one. Then, in the process of merging of the independently designed parts into an integral db scheme. There remain some open questions. First is how to formulate an algorithm, to support the above mentioned formal technique. The other could be the question how to extend the achieved results to db schemes.

6. REFERENCES

[BeKi1] Beeri C., Kifer M. "Elimination of Intersection Anomalies from Database Schemes", JACM 1986.

[BeKi2] Beeri C., Kifer M. "Uniqueness of the Elimination of Intersection Anomalies from Database Schemes", JACM 1986.

[GiZa]..Ginsburg S., Zaiddan M. "Properties of Functional Dependencie Families" JACM ,July 1982.

[Hu] Hull R. "Finitely Specifiable Implicational Dependency Families",JACM, April 1984.

[Ku] Kuzmanov S. "The Contributions to the Theory of Independent and Acyclic Relational Database Scheme Extensions", Ph. Diss., Institute of Mathematics, Faculty of Science,Novi Sad, 1989.

[KuMo1] Kuzmanov S., Mogin P. "The independent extensions of Database Schemes", 31. Yugoslav Conference of ETAN, Bled, 1987.

[Me] Mendelzon A. O. "Database States and their Tableaux", ACM TODS 9, 2, June 1984.

[MoKu1] Mogin P., Kuzmanov S. "Formal Aspects of the Database Independency", 31. Yugoslav Conference of ETAN, Bled, 1987.

[Sc] Sciore E. "Adding Attributes to Improve Database Schemes", Proc. SIGACT - SIGMOD PODS, 1983.

[Sh] Shoenfield J.R. "Mathematical Logic", Addison - Wesley Publ. Comp., 1987.

[Ull] Ullman J. D. "Principles of Database Systems", Comp. Sci. Press, Inc., Rockville, Md., 1982.

AVTOMATSKO UČENJE VODENJA DINAMIČNIH SISTEMOV

INFORMATICA 4/90

Keywords: dynamic systems control, inverted pendulum, machine learning.

Tanja Urbančič
Institut Jožef Stefan, Ljubljana

BOXES, AHC in CART so algoritmi, ki se lahko naučijo vodenja dinamičnega sistema. Učenje poteka s poskušanjem, med katerim program svoje odločitve ocenjuje in izpopolnjuje na podlagi informacij o odzivu vodenega sistema. Podajamo opis omenjenih algoritmov, njihove značilnosti in primerjavo med njimi. Primerjava temelji na rezultatih poskusov, v katerih so se programi naučili voditi inverzno nihalo.

MACHINE LEARNING OF DYNAMIC SYSTEMS CONTROL

BOXES, AHC and CART are algorithms which learn to control dynamic systems. During experimentation algorithms evaluate and improve their decisions based on the information about the reactions of the controlled system. In the paper the algorithms are described, their characteristics are given and compared. The comparison is based on the results of experiments in learning to control inverted pendulum.

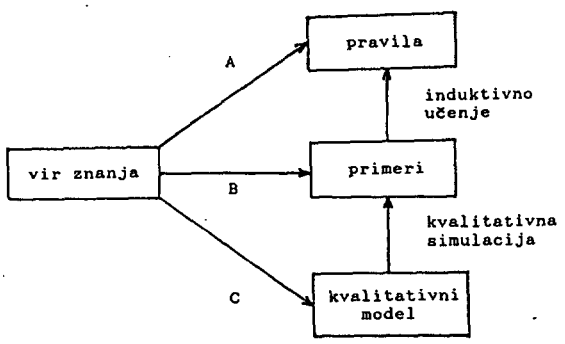
1. UVOD

Naloga vodenja je doseči željeno vedenje dinamičnega sistema z določenim vplivanjem nanj. Kljub nespornim uspehom klasične teorije vodenja je naloga za nekatere sisteme v njenem okviru zelo težka ali celo nerešljiva. Težave lahko nastopijo iz različnih razlogov, kot na primer:

- Ne poznamo zakonov, po katerih se vede sistem, ki ga želimo voditi.
- Kljub poznavanju zakonov vedenja nastopijo težave, ker je npr. sistem izrazito nelinearen, ker se njegove karakteristike s časom spreminjajo ali ker je prekompleksen.
- Sprotno zagotavljanje tako natančnih vrednosti, kot jih zahtevajo klasične metode, je predrago, predolgotrajno ali sploh nemogoče.
- Potrebno je zagotoviti hitro in jedrnato razlago vodenja.

Tovrstne probleme skušamo reševati s kvalitativnim vodenjem sistemov, pri katerem natančna numerična reprezentacija sistema ni potrebna. Vodenje poteka po pravilih, ki predpisujejo, kakšno ukrepanje je potrebno, če se sistem nahaja v določenem kvalitativnem stanju.

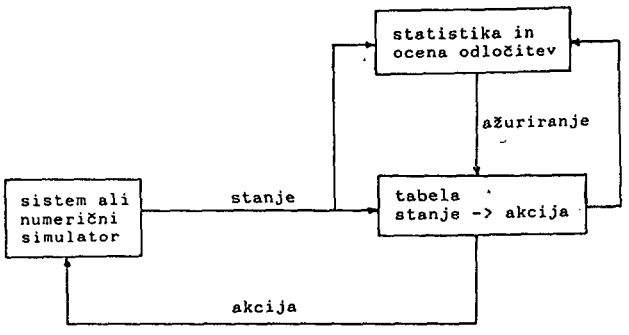
Znano je, da je formulacija ustrezne množice pravil (A na sliki 1) tudi za strokovnjaka težka naloga. Tega problema se lotevajo z razvojem metod za avtomatsko sintezo znanja (Michie & Bratko 1986). Ena od metod za avtomatsko sintezo znanja temelji na dejstvu, da strokovnjaki večinoma lažje posredujejo svoje znanje v obliki množice dobro izbranih primerov rešenih problemov (B na sliki 1). Iz množice primerov lahko z induktivnim sklepanjem rekonstruiramo ekspertov "know-how". Pristop je možno realizirati z metodami avtomatskega induktivnega učenja (Michalski et al. 1983, 1986; Kononenko 1985), ki so že dale odlične rezultate. Lahko jih uporabimo za izgradnjo baze znanja na podlagi poskusov, izvedenih bodisi z numeričnim simulatorjem (če obstaja) ali pa v realnem okolju. Težave se lahko pojavijo pri zagotavljanju neprotislovnosti in popolnosti baze znanja.



Slika 1: Različne metode zajemanja znanja

Lahko se zgodi, da niti s pomočjo eksperta niti s poskusi ne zajamemo vseh potrebnih primerov. V takem primeru se lahko obnese uporaba kvalitativnega modela (Bobrow & Hayes 1984; Weld & de Kleer 1990), kot kaže slika 1. Z izčrpno simulacijo dobimo vse možne primere vedenja sistema (seveda le na stopnji natančnosti modela), iz njih pa kakor prej generiramo operativna odločitvena pravila. Ta pristop je bil predstavljen in uspešno preizkušen na področju diagnostike v projektu KARDIO (Bratko in sod. 1988).

Na področju avtomatske sinteze vodenja je bil zaenkrat poudarek predvsem na generiranju primerov uspešnega vodenja s pomočjo statističnega učenja. Temelji na zamisli, da se lahko program sam nauči voditi sistem, ne da bi vnaprej poznal njegovo dinamiko ali notranjo strukturo. Učenje poteka s poskušanjem, med katerim program svoje odločitve ocenjuje in izpopolnjuje na podlagi opazovanja odziva vodenega sistema (slika 2). Za razliko od induktivnega avtomatskega učenja (Michalski in sod. 1983, 1986; Kononenko 1985), ki poteka na simboličnem nivoju in je blizu človekovemu načinu sklepanja, je to učenje numeričnega, statističnega značaja.



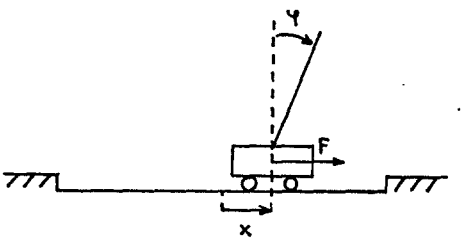
Slika 2: Učenje vodenja

Za tako vrsto učenja najdemo v literaturi tudi naziv učenje črne škatle (Forsyth & Rada 1986). V članku bomo obravnavali tri programe, ki se lahko na ta način naučijo voditi dinamični sistem: BOXES (Michie & Chambers 1968), AHC (Selfridge in sod. 1985) in CART (Connel & Utgoff 1987). Po opisu algoritmov se bomo posvetili njihovim značilnostim in jih primerjali med seboj. Primerjava bo temeljila na rezultatih poskusov, v katerih so se programi naučili voditi inverzno nihalo.

2. TESTNI PROBLEM: VODENJE INVERZNEGA NIHALA

Za testni problem smo izbrali enega od standardnih referenčnih problemov področja. To je problem vodenja vozička in palice, ki sta speta v inverzno nihalo. Pojavlja se v učbenikih in številnih člankih s področja matematične teorije vodenja. Pogosto je uporabljen kot preizkusni problem za alternativne pristope, saj ga je lahko razumeti in razmeroma objektivno oceniti rezultate. Kljub temu pa njegova rešitev ni preprosta.

Čeprav gre na prvi pogled zgolj za šolski problem, ima rešitev tudi uporabno vrednost. Zelo podobni problemi se pojavljajo npr. pri vodenju rakete ob vzletu in pri vodenju umetnih satelitov (Sammut & Michie 1989).



Slika 3: Voziček in palica, speta v inverzno nihalo

Na ravnem tiru omejene dolžine imamo voziček, na katerem je pritrjena homogena palica tako, da je vrtljiva okoli vodoravne osi, pravokotne na tirnice. Na voziček delujemo s silo F konstantne velikosti, ki ji v enakomernih časovnih razmikih lahko spremenimo predznak. Sistem je treba voditi tako, da palica ne pade in da voziček ves čas ostane na tirnicah (glej sliko 3).

Stanje sistema je določeno z vrednostmi naslednjih spremenljivk:

- lega vozička (x),
- hitrost vozička (v),
- naklon palice (fi),
- kotna hitrost palice (fiv).

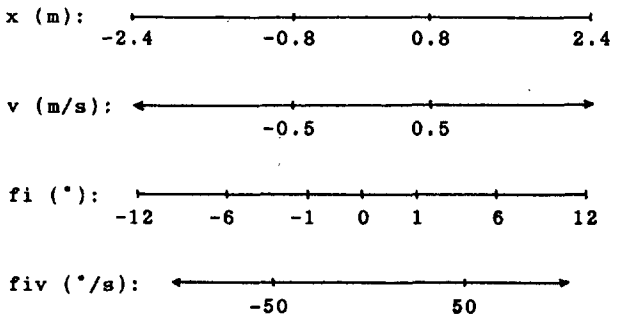
Dovoljeno območje za vsako spremenljivko je razdeljeno na manjše število disjunktnih intervalov, ki predstavljajo kvalitativne vrednosti spremenljivke. Čas je merjen diskretno, s korakom Δt. Ob vsakem koraku dobimo informacijo o trenutnem kvalitativnem stanju sistema. Če je sistem znotraj dovoljenega območja, dobimo kvalitativne vrednosti spremenljivk x, v, fi, fiv. Če pa je vsaj ena izmed spremenljivk prekoračila mejo dovoljenega območja, dobimo signal o neuspehu.

Namesto dejanske naprave uporabljamo numerični simulator, ki izračuna pospešek vozička in kotni pospešek palice iz ustreznih gibalnih enačb (Cannon 1967), vrednosti ostalih spremenljivk pa določi z Eulerjevo aproksimacijo. V vseh poskusih, ki jih navajamo, so bile vrednosti parametrov sistema in določitve mej za kvalitativne vrednosti spremenljivk, za dovoljeno območje in za začetna stanja enake. Podajamo jih na sliki 4.

Parametri sistema:

masa vozička	M	=	1 kg
masa palice	m	=	0.1 kg
dolžina palice	L	=	1 m
težni pospešek	g	=	9.8 m/s^2
velikost sile na voziček	F	=	10 N
časovni korak	Δt	=	0.02s

Razdelitev na intervale:



Omejitve za začetno stanje:

$$-0.1\ m < x < 0.1\ m$$
$$-6^\circ < fi < 6^\circ$$
$$v = 0\ m/s$$
$$fiv = 0^\circ/s$$

Dovoljeno območje: $-2.4\ m < x < 2.4\ m$, $-12^\circ < fi < 12^\circ$

Slika 4: Standardni parametri sistema voziček-palica in najpogostejša izbira intervalov za vrednosti spremenljivk

3. ALGORITEM BOXES

Michie in Chambers sta v svojem zdaj že klasičnem članku (Michie & Chambers 1968) predstavila algoritem BOXES in pokazala, da se program z njim lahko nauči voditi inverzno nihalo.

Sistem vozička in palice, ki sta speta v inverzno nihalo, ima štiridimenzionalen neskončen prostor stanj. Michie in Chambers sta ga zaradi lažje obvladljivosti razbila na končno število disjunktnih podprostorov, ki jim pravita "škatle" (od tod tudi ime algoritma). To sta storila z razdelitvijo zaloge vrednosti vsake spremenljivke sistema na intervale. Vrednost posamezne spremenljivke tako ni realno število, pač pa eden od vnaprej določenih intervalov. Trenutno kvalitativno stanje sistema je določeno s trenutnimi kvalitativnimi vrednostmi (intervali) spremenljivk sistema. Čas je merjen diskretno, s korakom Δt. Ob vsakem koraku dobimo informacijo o tem, v katerem izmed kvalitativnih stanj se nahaja sistem, ali pa signal, da je vsaj ena od spremenljivk prekoračila meje dovoljenega območja.

Učenje poteka tako, da program izvaja zaporedje poskusov. Vsak poskus se začne z naključnim stanjem znotraj določenega območja in traja, dokler ena od spremenljivk ne prekorači meje dovoljenega območja. Program vzdržuje tabelo "stanje -> akcija" in se med posameznim poskusom v istem stanju zmeraj odloči za isto akcijo. Za vsako stanje zbira informacijo o tem, koliko potez je sledilo v primeru, ko je izbral eno ali drugo akcijo. Shranjuje tudi podatek, kolikokrat je bila pri tem stanju v prejšnjih poskusih uporabljena ena oziroma druga akcija. Ob koncu vsakega poskusa program na podlagi zbranih informacij in ocenitvene funkcije ažurira tabelo "stanje -> akcija". Tako se v naslednjem poskusu pri nekaterih stanjih odloča za drugačno akcijo kot v prejšnjem. Učenje se konča, ko pride do poskusa, ki traja vnaprej določeno število korakov.

Oglejmo si podrobneje, kako se program odloči za akcijo v posameznem kvalitativnem stanju. Za vsako stanje beleži vrednosti količin:

LL (left life):
utežena vsota preživetij po levih akcijah v tem stanju (preživetje je število potez od vstopa v to stanje do signala o neuspehu),

RL (right life):
enako za akcijo "desno",

LU (left usage):
utežena vsota števila levih akcij v tem stanju med prejšnjimi poskusi,

RU (right usage):
enako za akcijo "desno",

$T_1, T_2, \ldots, T_N$:
časi, ob katerih je sistem v trenutnem poskusu vstopil v to stanje (merjeno s koraki).

Akcija "levo" pomeni silo z negativnim, akcija "desno" pa silo s pozitivnim predznakom. Po končanem poskusu program spremeni vrednosti za stanja, v katerih je bila uporabljena akcija "levo", po naslednjih pravilih (z X' označujemo prejšnjo vrednost X):

$$LL = LL' \cdot DK + \sum_{i=1}^{N} (T_F - T_i)$$

$$LU = LU' \cdot DK + N$$

$$RL = RL' \cdot DK$$

$$RU = RU' \cdot DK$$

Analogno program spremeni vrednosti za stanja, v katerih je uporabil akcijo "desno". Pri tem je:

N : število vstopov v to stanje med zadnjim poskusom
DK : utež trenutne izkušnje v odnosu na prejšnje, DK < 1
(Michie je izbral DK = 0.99)
T_F : čas, ob katerem je prišlo do neuspeha, $T_F \geq T_i$, i = 1, .., N

Za celoten sistem program računa GL ("global life") in GU ("global usage"), ki ju po vsakem poskusu spremeni:

$$GL = GL' \cdot DK + T_F$$

$$GU = GU' \cdot DK + 1$$

Po končanem poskusu za vsako stanje izračuna vrednosti

$$V_L = \frac{LL + K \cdot (GL/GU)}{LU + K} \quad \text{in}$$

$$V_R = \frac{RL + K \cdot (GL/GU)}{RU + K}$$

Parameter K (Michie je izbral K = 20) utežuje globalno izkušnjo v odnosu na lokalno. V naslednjem poskusu se bo program v stanjih, za katera je $V_L > V_R$, odločal za akcijo "levo", v ostalih pa za akcijo "desno".

4. UČENJE VODENJA Z UMETNO NEVRONSKO MREŽO

Za učenje vodenja lahko uporabimo tudi umetne nevronske mreže (Kononenko 1989; Kovačič 1989). Te so sestavljene iz nevronov, ki komunicirajo preko vezi. Vsak nevron generira svoj izhod na podlagi različno uteženih vhodov, uteži pa se med učenjem spreminjajo.

Oglejmo si pristop učenja z nevronsko mrežo, ki so ga predlagali Barto in sodelavci (Barto in sod. 1983) in ki je kasneje postal znan kot AHC (Adaptive Heuristic Control) algoritem. Tudi oni so za testno domeno izbrali vodenje inverznega nihala. Prostor stanj so kot pri pristopu BOXES razdelili na disjunktna območja. Izbrali so razdelitev na 162 območij, ki jo prikazuje slika 4.

Zaradi lažjega razumevanja si oglejmo najprej osnovno različico nevronske mreže (Kovačič 1989), ki jo prikazuje slika 5. Vsebuje en sam nevron, katerega naloga je izbiranje akcije. Nevron ima 163 vhodov. Na enem vhodu je signal (r) o neuspehu sistema: r(t) postane -1, ko ena od merjenih količin prestopi meje dovoljenega območja, sicer pa ima vrednost 0. Preostalih 162 vhodov posreduje informacijo o stanju sistema kot binarni vektor

$$S(t) = (s_1(t), s_2(t), \ldots, s_{162}(t))$$

Če se sistem ob času t nahaja v i-tem stanju, je $s_i(t) = 1$, vse ostale komponente vektorja S(t) pa imajo vrednost 0. Čas je merjen diskretno, s korakom $\Delta t = 0.02$ sekunde. Vsakemu od 162 vhodov, ki opisujejo stanje sistema, je pridružena utež $w_i(t)$. Uteži so realna števila, od katerih je odvisna izbira akcije y(t):

$$y(t) = f\left(\sum_{i=1}^{162} w_i(t)\, s_i(t) \right)$$

$$f(x) = \begin{cases} 1 & \text{(akcija "desno"), če } x \geq 0 \\ -1 & \text{(akcija "levo"), če } x < 0 \end{cases}$$

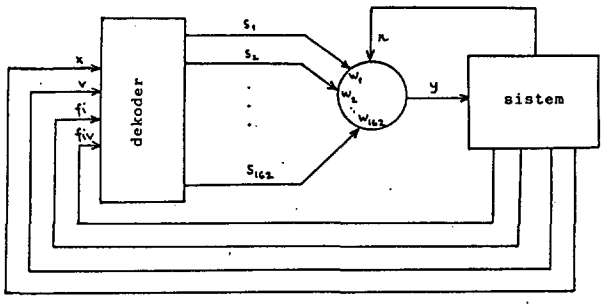
Učenje je pravzaprav spreminjanje uteži $w_i(t)$. Le-to poteka po naslednjih formulah:

$$w_i(t+1) = w_i(t) + \text{alfa}\ r(t)\ e_i(t)$$

$$e_i(t+1) = \text{delta}\ e_i(t) + (1 - \text{delta})\ y(t)\ s_i(t)$$

Pri tem je:

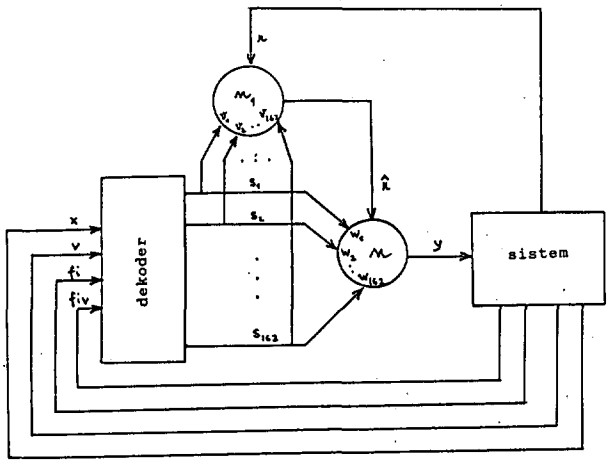
$e_i(t)$: časovna sled i-tega stanja
alfa : hitrost spreminjanja uteži, 0 < alfa (pri Bartu alfa = 1000)
delta : hitrost spreminjanja časovne sledi $e_i(t)$, $0 \leq \text{delta} < 1$ (delta = 0.9)



Slika 5: Osnovna različica nevronske mreže za vodenje

Dokler ni signala o neuspehu, je r(t) = 0, zato se takrat uteži ne spreminjajo. Negativna vrednost r(t) sporoči neuspeh sistema in povzroči spremembo uteži. Velikost spremembe posameznih uteži je odvisna od časovnih sledi. Časovna sled za posamezno stanje se ob vstopu v to stanje močno spremeni, potem pa postopoma izzveneva. Posledica tega je, da se najbolj spremenijo uteži na vhodih, ki ustrezajo stanjem tik pred padcem, pri ostalih pa je sprememba zelo majhna.

Učenje je precej oteženo, ker ne vemo, kako dobra je posamezna odločitev. Edina razpoložljiva informacija je le signal, ki sporoči padec palice oziroma iztirjenje vozička. Do odločilne napake v vodenju pa je seveda lahko prišlo že precej prej. Zato je v Bartovem pristopu prej opisani osnovni različici mreže dodan še en nevron (n1 na sliki 6), ki se uči napovedovati neuspeh.



Slika 6: Nevronska mreža z nevronom za napovedovanje neuspeha

Nevron n1 ima 163 vhodov, s katerih dobiva vrednosti $s_1(t)$, ..., $s_{162}(t)$ in r(t) z enakim pomenom kot prej. Vsem vhodom razen zadnjega so pridružene uteži v_1, ..., v_{162}. Uteži se spreminjajo po naslednjih formulah:

$$p(t) = \sum_{i=1}^{162} v_i(t)\, s_i(t)$$

$$v_i(t+1) = v_i(t) + \text{beta}\,(r(t) + \text{gama}\, p(t) - p(t-1))\, a_i(t)$$

$$a_i(t+1) = \text{lambda}\, a_i(t) + (1 - \text{lambda})\, s_i(t)$$

Pri tem je:

p(t) : $p(t) = v_i(t)$: napoved euspeha, če je sistem v i-tem stanju

beta : hitrost spreminjanja uteži, beta > 0 (pri Bartu beta = 0.5)

gama : faktor znižanja: povzroči zniževanje napovedi neuspeha, kadar ni signala o neuspehu; $0 < \text{gama} \leq 1$ (pri Bartu gama = 0.95)

$a_i(t)$: časovna sled i-tega stanja, ki ni neposredno odvisna od akcije (za razliko od sledi $e_i(t)$, ki je odvisna od y(t))

lambda : hitrost spreminjanja časovne sledi $a_i(t)$; $0 \leq \text{lambda} < 1$ (pri Bartu lambda = 0.8)

Uteži se spremenijo, kadar p(t-1) slabo aproksimira r(t) + gama p(t). Povečajo se, kadar se sistem premakne v stanje z višjo napovedjo neuspeha. Sprememba uteži je večja pri vhodih, ki ustrezajo stanjem, v katerih smo bili tik pred tem; pri ostalih je manjša, ker se časovna sled $a_i(t)$ približuje vrednosti 0, če sistem dalj časa ne vstopi v i-to stanje.

Izhod nevrona n1 v času t označimo z $\hat{r}(t)$, izračunamo pa takole:

$$\hat{r}(t) = r(t) + \text{gama}\, p(t) - p(t-1)$$

Dokler je sistem znotraj dovoljenega območja, je r(t) = 0. V tem primeru je $\hat{r}(t)$ razlika med trenutno napovedjo neuspeha, pomanjšano za faktor gama, in predhodno napovedjo neuspeha. Ko sistem prekorači meje dovoljenega območja, postane

$$r(t) = -1$$
$$p(t) = 0$$

zato je v tem primeru $\hat{r}(t)$ enako razliki med signalom neuspeha in napovedjo neuspeha. Izhod nevrona n1 je vhod v nevron, ki izbira akcijo. Ta se sedaj ne uči na osnovi informacije o tem, ali je prišlo do neuspeha ali ne (r(t)), ampak namesto tega upošteva oceno $\hat{r}(t)$, ki mu jo posreduje nevron n1.

5. ALGORITEM CART

Algoritem CART (Connell & Utgoff 1987) ne uporablja intervalskih vrednosti spremenljivk, ampak številske. V primerjavi z že obravnavanima pristopoma je zahteva po numeričnih vrednostih slabost, zato pa odpade vnaprejšnje določanje meja intervalov. Algoritem opredeljujejo: inicializacija, odločanje za akcijo, ocenjevanje posameznih stanj in učenje.

Sistem je na začetku postavljen v stanje, ki zagotavlja, da je vzdrževanje sistema v dovoljenem območju možno. Voziček je blizu izhodišča, palica skoraj navpična, hitrost vozička in kotna hitrost palice pa imata vrednost nič.

Odločanje za akcijo v določenem stanju poteka tako, da sistem po izbrani akciji preide v "boljše" stanje. Na vsakem koraku se odloča, ali bo ponovljena zadnja izvršena akcija ali pa jo je treba spremeniti. Težava je v tem, da program zaradi nepoznavanja dinamike sistema ne more predvideti, kaj se bo zgodilo v primeru uporabe ene ali druge akcije. Zato spremlja gibanje sistema v faznem prostoru, to je abstraktnem prostoru, razpetem na spremenljivke sistema (x, v, fi, fiv). Opazuje, kako se je po uporabi določene akcije v predhodnem stanju spremenil kot med vektorjem, ki nakazuje trenutno smer gibanja sistema in vektorjem, ki kaže v smer maksimalnega naraščanja ocene stanja. Zmanjšanje tega kota kaže, da je šlo za zaželjen premik, zato se ista akcija v trenutnem stanju ponovi. V nasprotnem primeru se odloči za nasprotno akcijo.

Za vsako stanje lahko program poda oceno, ki je število med -1 (za nezaželjena stanja) in 1 (za zelo zaželjena). Za nekatera stanja določi ocene po vnaprej določenem pravilu, dobljenim s poskušanjem, za ostala pa jih računa z interpolacijo. Ocenjuje takole:

- Začetno stanje privzame kot zaželjeno in mu pripiše oceno 1.

- Stanje, iz katerega je sistem neposredno prešel v stanje izven dovoljenega območja, označi kot nezaželjeno in mu določi oceno -1.

- Zaželjeno je tudi stanje, iz katerega je sistem prešel v stanje, v katerem so absolutne vrednosti vsaj treh spremenljivk zmanjšane in po katerem je sistem preživel še vsaj 50 korakov. Po vsakem poskusu, ki traja vsaj 100 korakov, program poišče vsa stanja s temi lastnostmi in jim pripiše oceno 1.

- Za ostala stanja računa ocene z interpolacijsko funkcijo, ki upošteva razdaljo stanja od zaželjenih in nezaželjenih stanj in je utežena tako, da na oceno bolj vplivajo bližnja stanja:

Za točko

$$x = (x_1, x_2, \ldots, x_m)$$

dobimo oceno z interpolacijsko formulo

$$f(x) = \frac{\sum_{i=1}^{n} w_i \, f(x_i)}{\sum_{i=1}^{n} w_i}$$

pri čemer i teče po vseh stanjih, za katera oceno že poznamo, utež w_i pa je

$$w_i = \prod_{\substack{j=1, \\ j \neq i}}^{n} (d_j)^p \quad , \qquad p > 0$$

in

$$d_j = \sqrt{(x_1 - x_{1,j})^2 + \ldots + (x_m - x_{m,j})^2}$$

Connell in Utgoff sta izbrala p = 2.

Cilj učenja je izboljšanje ocen za stanja. Učni primeri so stanja z oceno 1 oziroma -1.

Ko pridemo do novega učnega primera, dobi interpolacijska formula nov člen in ocene stanj se spremenijo. Zato program v naslednjem poskusu običajno pregleda drugo območje prostora stanj, kar zagotavlja majhno, a informativno množico učnih primerov.

6. ZNAČILNOSTI UČENJA BREZ UPOŠTEVANJA ZNANJA O SISTEMU

Testni problem za vse tri algoritme je bilo vodenje inverznega nihala s parametri, podanimi na sliki 4. Značilnosti posameznih algoritmov smo obravnavali na podlagi člankov njihovih avtorjev, Sammutovega primerjalnega članka (1988), za BOXES in AHC pa tudi na podlagi ponovljenih eksperimentov (Urbančič 1990).

6.1. Hitrost učenja

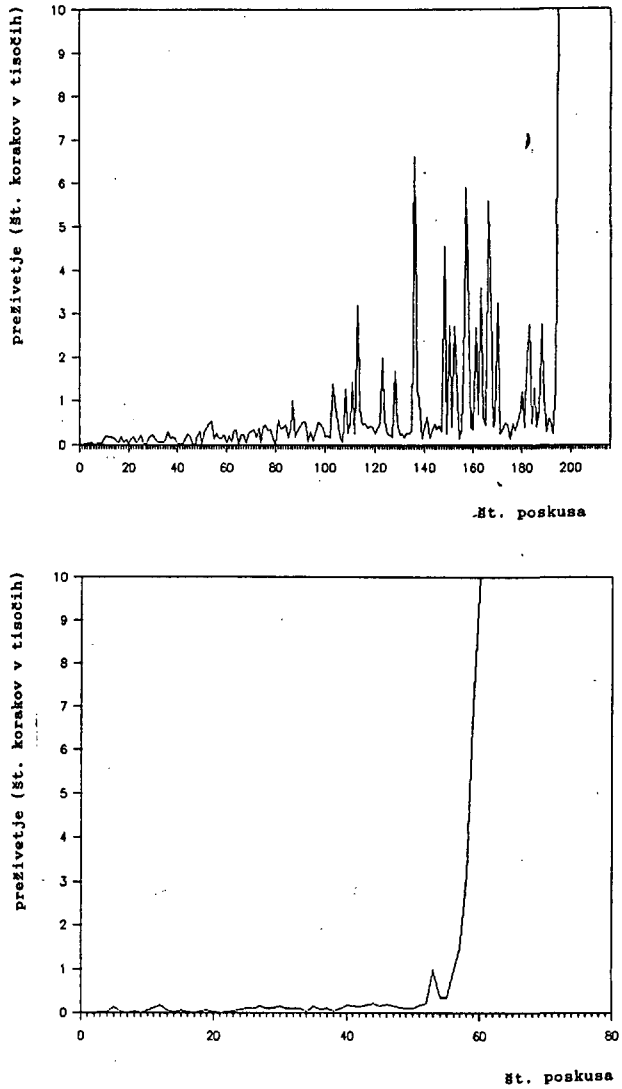
Po primerjalnem članku (Sammut 1988) je algoritem BOXES potreboval v povprečju 225 poskusov, preden je prišlo do poskusa, v katerem je sistem preživel 10000 korakov. Za isti cilj je algoritem AHC potreboval v povprečju 90 poskusov, algoritem CART pa 13 poskusov. Ta podatek pa seveda še ne pove vsega o hitrosti učenja. Velike razlike so namreč v tem, koliko časa porabi posamezen algoritem za izvedbo posameznega poskusa. To je odvisno od hitrosti sprejemanja odločitev med poskusom. Pri algoritmu BOXES poteka odločitev za akcijo med poskusom izredno hitro, saj je treba samo pogledati v tabelo (O(1) algoritem). Pri algoritmu AHC je treba računati odločitve za vsako stanje po vsaki potezi (O(n), pri čemer je n število stanj). Algoritem CART pa je v tem pogledu očitno še veliko zahtevnejši. Dejansko je potekalo posamezno učenje z algoritmom BOXES in AHC od nekaj minut do treh ur (oboje na PC/AT kompatibilnem računalniku), z algoritmom CART pa (po Sammutu) kar nekaj ur na računalniku VAX 11/750.

6.2. Potek učenja

Pri algoritmu BOXES ostane med enim poskusom, to je v času od začetnega stanja do signala o neuspehu, tabela "stanje -> akcija" nespremenjena: med poskusom se sistem v istem stanju zmeraj odloči za isto akcijo. Šele po signalu o neuspehu na podlagi ocenitvenih funkcij spremeni posamezne odločitve.

Pri algoritmu AHC učenje ni vodeno le z dejanskim neuspehom, ampak tudi z ocenami za neuspeh, ki jih računa ob vsakem koraku poskusa. Zato se odločitev za akcijo v določenem stanju lahko spremeni tudi med poskusom. Mreža prepozna premik iz stanja z večjo oceno za neuspeh v stanje z manjšo oceno za neuspeh kot ugoden, obraten premik pa kot nezaželen. Tako se nauči izogibanja nevarnim območjem, česar algoritem BOXES ni sposoben. Tudi CART je zasnovan tako, da se izogiba že odkritim nezaželjenim stanjem.

Za učenje z algoritmom BOXES je značilno izrazito nihanje dolžine poskusov med učenjem (slika 7.a). Uspešnemu poskusu poogosto sledi poskus z veliko krajšim preživetjem, zatem pa se izvajanje znova izboljša. Pri nevronski mreži tega pojava nismo opazili: preživetje sistema je med učenjem dolgo zelo skromno in rahlo niha, nato pa pride do bliskovitega izboljšanja (slika 7.b).



Slika 7: Primer poteka učenja z algoritmom BOXES (zgoraj) in z nevronsko mrežo (spodaj)

6.3. Zanesljivost naučenega vodenja

Cilj učenja vodenja pri večini avtorjev ni splošno pravilo za vodenje sistema, ampak uspešno vodenje v posameznem primeru. Kriterij uspeha je določeno število doseženih korakov, v primeru vodenja inverznega nihala ponavadi 10000. Ali lahko rečemo, da se je sistem po takem uspešnem poskusu dejansko naučil voditi sistem? Zanesljivost dobljenega pravila (tabele "stanje -> akcija") lahko preverimo tako, da ga ponovno preizkusimo pri naključno izbranih začetnih stanjih z dovoljenega območja. Za sistem BOXES smo s poskusi ugotovili, da je zanesljivost razmeroma majhna. Preizkusili smo 15 "naučenih" tabel, vsako 20-krat. Povprečno preživetje je bilo približno 5000 korakov in le v 15% poskusov je bilo ponovno zadoščeno kriteriju uspeha.

Poskusi niso pokazali nobene povezave med hitrostjo učenja in zanesljivostjo naučenih pravil.

Pri nevronski mreži obstaja povezava med hitrostjo učenja in območjem zanesljivosti. Če želimo dobiti bolj zanesljiva pravila, je potrebno izvajati učenje na več začetnih stanjih, kar podaljša čas učenja. Do pravil za eno samo začetno stanje pa lahko mreža pride hitreje. Uporabnik ima torej izbiro glede na omejitve pri posameznem problemu. Razen tega ima naučena nevronska mreža v utežeh nevrona za napovedovanje neuspeha dovolj informacij za razmeroma dobro določitev zanesljivosti pri različnih začetnih stanjih.

6.4. Adaptivnost

Za sisteme, katerih delovanje je povezano z zunanjim svetom, je zelo pomembna adaptivnost, to je sposobnost prilagajanja novim razmeram. Te lahko nastopijo zaradi predvidenih ali nepredvidenih sprememb v okolici in lahko bistveno spremenijo vedenje sistema, ki ga želimo voditi. Metode za učenje vodenja, ki ne temeljijo na poznavanju dinamike in točnih vrednosti parametrov sistema, imajo v tem pogledu veliko prednost pred klasičnimi metodami. Za prilagoditev novim razmeram namreč ni potrebna vnaprejšnja identifikacija sprememb vodenega sistema.

Sammut je algoritme preizkusil na problemu vodenja inverznega nihala s silo, ki je v eni smeri velika 10N, v drugi pa 5N (Sammut 1988). BOXES se je naučil vodenja v 837 poskusih, AHC v 4216 poskusih, CART pa ni uspel rešiti problema. Število potrebnih poskusov se je torej v primerjavi s problemom, kjer je sila v obe smeri enako velika, pri sistemu BOXES pomnožilo le z enomestnim faktorjem. Lahko rečemo, da je BOXES v tem pogledu precej bolj robusten od AHC, pri katerem se je število potrebnih poskusov pomnožilo s 50.

Selfridge s sodelavci (Selfridge in sod. 1985) poroča o poskusih, ki potrjujejo visoko stopnjo adaptivnosti nevronske mreže. Sistem s standardnimi parametri (slika 4) se je mreža naučila voditi 10000 korakov v približno 70 poskusih, nato pa je obvladala naslednje spremembe:

- spremembo v velikosti sile od +10N in -10N na +12N in -8N;
- spremembo v masi palice z 0.1kg na 1kg v povprečno 20 dodatnih poskusih;
- spremembo v masi in dolžini palice na dve tretjini prvotne vrednosti v povprečno 6 dodatnih poskusih;
- spremembo v dolžini tračnic s 4.8m postopoma na 4m, 3m in 2m v povprečno 4+9+5=18 dodatnih poskusih.

Po navedbah Connella in Utgoffa (1987) je vse naštete spremembe obvladal tudi CART in to v manj kot 18 poskusih.

6.5. Diskretizacija zalog vrednosti

BOXES in originalni Bartov pristop z nevronsko mrežo potrebujeta vnaprejšnjo razdelitev zalog vrednosti za spremenljivke sistema na intervale. Pri slabo izbranih mejah intervalov je učenje počasno, lahko pa se zgodi, da se sistem sploh ne nauči vodenja. Določitev potrebnega števila intervalov in njihovih mej za posamezne količine v splošnem ni lahka naloga. Kovačič je Bartov pristop dopolnil tako, da se nevronska mreža sama nauči izbrati primerne meje intervalov (Kovačič 1989). V primeru, da so bile vse meje sprva postavljene v točko 0, je mreža potrebovala za učenje 36 poskusov. V primeru, ko so bile meje sprva na skrajnem robu dovoljenih vrednosti, pa je bilo število poskusov 10-krat večje: 366.

6.6. Parametri za algoritem učenja

Obravnavani algoritmi učenja uporabljajo številske parametre, za katere ni jasno, kako jih je treba določiti, da bo učenje uspešno. Pri algoritmu BOXES je tak parameter K. Pri algoritmu AHC jih je več: alfa, beta, gama, delta in lambda. Sammut poroča (Sammut 1988), da je BOXES z isto vrednostjo parametra K obvladal različne sisteme, pri AHC pa se je izbira vrednosti parametrov izkazala za bolj problemsko odvisno. Program CART zahteva en parameter tega tipa v formuli za interpolacijo. Posebno težavo pa predstavlja določitev pravila za izbiro zaželjenih oziroma nezaželjenih stanj. V njem nastopajo še trije numerični parametri, ki sta jih Connell in Utgoff sicer določila eksperimentalno, vendar domnevata, da bi jih lahko dobila tudi avtomatsko, na primer kot mnogokratnike dolžine preživetja naključno vodenega sistema.

6.7. Razlaga

Eden glavnih problemov pri učenju vodenja je interpretacija rezultatov. Do odločitev za akcije prihaja na podlagi netrivialnih ocenitvenih funkcij, zato jih je težko pojasniti brez določenega matematičnega znanja in izkušenj. Razen tega je dobljeno pravilo vodenja, predstavljeno s tabelo "stanje -> akcija", pri večjem številu stanj nepregledno in neprimerno za nazorno razlago.

Pomagamo si lahko s sistemi za avtomatsko induktivno učenje, ki rezultate učenja predstavijo v drugačni obliki. Tak sistem dobljeno tabelo "stanje -> akcija" uporabi kot učne primere; vsako pravilo, ki stanju priredi akcijo, predstavlja en učni primer. Rezultat je z induktivnim sklepanjem izpeljano pravilo. Predstavljeno je lahko kot odločitveno drevo, kot npr. pri sistemih Assistant Professional (Cestnik in sod. 1987) in C4 (Quinlan 1986), ali pa kot množica produkcijskih pravil, kot npr. pri sistemih AQ15 (Michalski in sod. 1986) in Gynesis (Gams 1988). Lahko pa se zgodi, da je tudi tako predstavljeno pravilo vodenja razmeroma nepregledno. Sliko lahko meglijo stanja, ki jim je v tabeli prirejena slabo izbrana akcija, ki pa ni vplivala na rezultat učenja - bodisi da sistem sploh ni prišel v to stanje ali pa so sosednja stanja učinek slabe akcije popravila. Rezultate lahko izboljšamo, če upoštevamo le stanja, v katerih se je sistem zares znašel, in če ob učenju tabele uporabimo dovolj različnih začetnih stanj.

7. ZAKLJUČEK

Povzemimo najprej skupne značilnosti obravnavanih pristopov za učenje vodenja:

- Za učenje vodenja sistema ne potrebujejo znanja o njegovi dinamiki. Če bi ga imeli in ga znali upoštevati, bi bili seveda bolj učinkoviti. Vendar pa tako znanje ni vedno na razpolago.

- Do določene mere so adaptivni in robustni, kar je zelo pomembno za vsak program, ki deluje v interakciji z realnim sistemom.

- Za preverjanje smiselnosti delovanja programa je potrebno, da program zna generirati razumljivo razlago za svoje odločitve. V ta namen so potrebni dodatni mehanizmi, npr. induktivno učenje.

- Za učenje potrebujejo numerične parametre, ki pomembno vplivajo na učenje in jih moramo uganiti.

- Učenje poteka z eksperimentiranjem. Seveda ne moremo zmeraj poljubno dolgo in poljubnokrat izvajati poskusov, bodisi ker bi bilo to predrago, prenevarno ali sploh nemogoče. Zato težimo za čimhitrejšim učenjem. Učenje lahko pospešimo z opustitvijo predpostavke, da o vodenem sistemu ne vemo ničesar, saj v večini realnih primerov to vendarle ne drži.

V tabeli 1 podajamo še primerjavo značilnosti posameznih pristopov.

	BOXES	nevr. mreže	CART
število potrebnih poskusov	veliko	srednje	majhno
hitrost sprejemanja odločitve	zelo hitro	hitro	počasi
zahteva vnaprejšnjo diskretizacijo	da	ne	ne
zahteva num. vrednosti spremenljivk	ne	ne	da
število parametrov učenja	1	5	4

Tabela 1: Značilnosti posameznih pristopov k učenju vodenja

Poskusi so pokazali (Urbančič 1990), da lahko z vključitvijo skrajno preprostega delnega pravila o problemskem področju v algoritem BOXES pospešimo učenje za faktor 10, hkrati pa povečamo zanesljivost naučenih tabel s 15 na 50%. Nadaljnje raziskave gredo v smeri sinteze pravil vodenja s pomočjo kvalitativnega modela sistema (Bratko 1989; Urbančič 1990). Z generiranjem pravil vodenja s pomočjo kvalitativnega modela namreč lahko zagotovimo določeno stopnjo vodenja sistema brez poskušanja. Ker pa je vsak model le približek realnega sistema, se nam zdi obetavna povezava obeh pristopov, modeliranja in učenja.

ZAHVALA

Zahvaljujem se Ivanu Bratku in Claudu Sammutu za številne zanimive pogovore, ki so vplivali na vsebino tega članka. Hvala tudi Matevžu Kovačiču za pomoč pri sežnanjanju z nevronskimi mrežami ter Igorju Kononenku za skrben pregled članka in tehtne pripombe.

LITERATURA

Barto, A. G., Sutton, R. S., Anderson, C. W. (1983) Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, No.5, 834-846.

Bobrow, D. G., Hayes, P. J. (eds.) (1984). Artificial Inteligence, Special Volume on Qualitative Reasoning about Physical Systems, Vol. 24.

Bratko, I. (1989) Pole balancing - a study in qualitative reasoning about control. ISSEK Workshop, Udine.

Bratko, I., Mozetič, I., Lavrač, N. (1988) Automatic Synthesis and Compression of Cardiological Knowledge. Machine Intelligence 11, Michie, D., Richards, J. (eds.), Oxford University Press.

Cannon, R. (1967) Dynamics of physical systems. McGraw-Hill, Inc.

Cestnik, B., Kononenko, I., Bratko, I. (1987) ASSISTANT 86: A knowledge elicitation tool for sophisticated users. Bratko, I., Lavrač, N. (eds.), Progress in machine learning. Wilmslow: Sigma Press.

Connell, M. E., Utgoff, P. E. (1987) Learning to Control a Dynamic Physical System. Proceedings of the 6th National Conference on Artificial Intelligence. Morgan Kaufmann, 456-459.

Forsyth, R., Rada, R. (1986) Machine Learning: Applications in expert systems and information retrieval. Ellis Horwood.

Gams, M. (1988) Principi poenostavljanja v sistemih za avtomatsko učenje. Doktorska disertacija, Fakulteta za elektrotehniko in računalništvo, Ljubljana.

Kononenko, I. (1985) Strukturno avtomatsko učenje. Informatica, št.3/85, 44-55.

Kononenko, I. (1989) Nevronske mreže. Informatica, št. 2/89, 56-71.

Kovačič, M. (1989) Balansiranje palice z nevronsko mrežo. Zbornik konference ETAN 89, Novi Sad, junij 1989.

Michalski, R. S., Carbonell, J. G., Mitchell, T. M. (eds.) (1983, Vol.1; 1986, Vol.2) Machine Learning: An Artificial Intelligence Approach. Palo Alto: Tioga Publishing Company.

Michalski, R. S., Mozetič, I., Hong, J., Lavrač, N. (1986) The multipurpose incremental learning system AQ15 and its testing application to three medical domains. Proc. National Conference on Artificial Intelligence, AAAI-86, 1041-1045, Philadelphia, PA: Morgan Kaufmann.

Michie, D. & Chambers, R. A. (1968) BOXES: an experiment in adaptive control. Dale, E., Michie, D. (eds.), Machine Intelligence 2, 137-152, Edinburgh University Press.

Michie, D., Bratko, I. (1986) Expert Systems: Automating Knowledge Acquisition, Addison Wesley.

Quinlan, R. (1986) Induction of Decision Trees. Machine Learning 1, 81-106, Kluwer Academic Publishers.

Sammut, C. (1988). Experimental Results from an Evaluation of Algorithms that Learn to Control Dynamic Systems. Proceedings of 5th International Conference on Machine Learning, Ann Arbor, Michigan. Morgan Kaufmann.

Selfridge, O. G., Sutton, R. S., Barto, A. G. (1985) Training and Tracking in Robotics. Proceedings of the 9th International Conference on Artificial Intelligence. Morgan Kaufmann, Vol.1, 670-672.

Urbančič, T. (1990) Avtomatska sinteza znanja za kvalitativno vodenje dinamičnega sistema. Magistrsko delo, Fakulteta za elektrotehniko in računalništvo, Ljubljana.

Weld, D. S., de Kleer, J. (eds.) (1990) Readings in Qualitative Reasoning about Physical Systems. Morgan Kaufmann.

Marjan Bradeško
Ljubo Pipan
Fakulteta za elektrotehniko in računalništvo,
Tržaška 25, Ljubljana

Keywords: Identification, chipcards, optical cards, compression, decompression, ciphering, deciphering.

POVZETEK - Uporaba slike - fotografije v identifikacijske namene je bila doslej omejena na klasične slike v dokumentih. V članku prikazujemo možnost, ki jo ponuja kombinacija optične in čipkartice. Opisan je eden od možnih postopkov pri identifikaciji s sliko, zapisano na taki kartici.

ABSTRACT - The usage of pictures - photographies for identification purposes was bound to classical photographies in documents up till now. In the article the combination of optical and chipcard is presented and one of possible procedures at identification by picture , stored on such a card, is described.

1. Uvod

Dokumenti, ki so namenjeni identifikaciji uporabnika na osnovi slike - fotografije, imajo eno od velikih pomanjkljivosti - na izgubljenem ali ukradenem dokumentu je možno sliko zamenjati in dokument uporabljati naprej. Če ni strogih dodatnih kontrol, so možne hude zlorabe. Tipičen, najbolj drastičen primer so izgubljeni potni listi, pa tudi drugi podobni dokumenti.

Če bi hoteli zagotoviti večjo varnost in preprečiti take ponaredbe, bi morala biti za uporabnika slika nevidna in naj je tudi ne bi bilo možno spreminjati. To je možno le, če slika ni klasična, pač pa shranjena v elektronski obliki v ustreznem pomnilniku in po možnosti še v šifrirani obliki. Ob preverjanju se taka slika dešifrira in prikaže na nekem prikazovalniku.

Pogoj za učinkovito uporabo takega načina preverjanja identitete uporabnika je velika kvaliteta slike - visoka ločljivost in veliko število barv, ki so na voljo za sliko. Na tem mestu pa trčimo na veliko oviro - za zapis slike rabimo namreč velik pomnilnik, ki ga je za prenosljivo obliko težko implementirati. Doslej so se v identifikacijskih postopkih v glavnem uporabljale kartice z magnetnim zapisom, čipkartice in optične kartice, katerih prednosti in pomanjkljivosti opisujemo v naslednjem poglavju.

2. Prednosti in slabosti identifikacijskih kartic

Večina kartic je v standardizirani obliki velikosti kreditnih kartic (54 x 85.6 x 0.76 mm). Kartice z magnetnim zapisom so še veliko v uporabi, vendar ne zagotavljajo niti posebne varnosti niti ne omogočajo zapisa velike količine podatkov. Čipkartice so po inteligenci oziroma procesorski moči, ki jim jo daje vgrajen mikroprocesor, v vrhu. Zaradi procesorskih zmogljivosti spadajo med 'aktivne' kartice. Tudi pomnilnika premorejo nekaj več (do 32 K EEPROM-a, nekaj ROM-a in manjšo količino RAM-a), vendar za shranjevanje visoko kvalitetnih slik še vedno premalo. Velikost in kapaciteta vgrajenih čipov sta omejena zaradi dovoljenega upogibanja in torzije po standardih ISO DIS 7816. Optične kartice ponujajo veliko količino pomnilnika (npr. Canon OC - 2 Mbyte), nimajo pa nobene procesorske moči in sodijo tako kot kartice z magnetnim zapisom med t.i. 'pasivne' kartice.

Če združimo naša razmišljanja, ugotovimo, da bi za prej opisano aplikacijo izredno ustrezala kombinacija optične in čipkartice - tako možnost bomo tudi podrobneje analizirali, kartici pa bomo rekli opto-čipkartica.

3. Struktura kartice za identifikacijo s sliko

Pravzaprav gre za dve polovici - pomanjšani sliki originalnih kartic. Oboje je združeno v kartico običajne velikosti po standardih ISO. Elektronski del kartice predstavlja mikroprocesor, ki na čipu vsebuje še nekaj ROM-a (za vgrajene programirane postopke), EEPROM-a in morda RAM-a. V primeru, da gre za sodobnejši tip čipkartice brez fizičnih kontaktov (CCD oziroma C2 Card - Connectionless Chip Card), je dodan še čip CCI - Contactless Chip Interface. Velikost čipa (čipov) je odvisna od zahtev po procesorski moči naše kartice oziroma zahtev aplikacije. Položaj kontaktov je opredeljen s standardom ISO, v pripravi pa je tudi standard za C2-kartice.

Drugi del kartice je površina, namenjena optičnemu zapisu podatkov z laserjem. Digitalni papir kot optična površina ustreza tudi zahtevam po upogibanju in torziji kartice, specificiranih v standardih. Površino lahko večamo oziroma manjšamo glede na zahteve po kapaciteti pomnilnika in glede na velikost obstoječega procesorskega dela na kartici. Vzemimo, da je s tako površino pokrita polovica kartice. Na voljo imamo približno 1 MByte pomnilnika tipa WORM (Write Once Read Many times).

Ob vsem tem je seveda treba omeniti tudi napravo, ki take kartice bere oziroma nanje piše. Pri opisani aplikaciji je zlasti pomembno branje, saj se le-to izvaja velikokrat in mora biti hitro. Naprava mora vsebovati laserski del z ustreznimi lečami za branje oziroma zapisovanje optičnih podatkov in ustrezni del za komunikacijo z

vgrajenimi čipi - bodisi preko direktne povezave s kontakti na kartici ali z induktivno/kapacitivno povezavo v primeru C2-kartic (Contactless). Taka naprava (glede na podobne, že obstoječe) je relativno draga, vendar je njena cena v nekaterih 'občutljivih' aplikacijah vsekakor opravičljiva. Naprava mora vsebovati tudi dovolj procesorske moči (npr. specializirane čipe - signalne procesorje ali ASIC-vezja) za dekodiranje prebrane slike z optične površine. Slika mora biti namreč komprimirana, saj je količina pomnilnika še vedno majhna za sliko, ki mora ustrezati kvaliteti barvne fotografije.

4. Uporaba opto-čipkartice pri identifikaciji s sliko

Na centralnem mestu, kjer izdajajo kartice, se z laserjem na optično površino zapiše digitalizirana slika - fotografija uporabnika. Slika se pred zapisovanjem ustrezno predobdela. Najprej se komprimira z eno od metod za komprimiranje. Glede na dosedanja dogajanja v teleinformatiki in razvoju ISDN kaže, da bo prevladala metoda DCT (Discrete Cosine Transform), saj je po standardih CCITT predvidena kot transformacija pri prenosu gibljive slike. Pri komprimiranju so doseženi dobri rezultati, potrebna pa je precejšnja procesorska moč. Tako kompresija kot dekompresija se izvajata izven kartice - dekompresija v bralno/pisalni napravi pri identifikaciji.

Slika je na kartici iz varnostnih razlogov lahko shranjena še v šifrirani obliki. Ob izdaji kartice se v zaščiten del pomnilnika kartice (del EEPROM-a) vpišejo podatki o lastniku kartice (ID_Data). Na centralnem mestu se na podlagi tajnega algoritma f iz uporabnikovih podatkov (priimek, ime, rojstni datum, naslov...) izračuna enolična identifikacijska številka kartice (ID_Num). Uporabnik poleg tega izbere svojo osebno identifikacijsko številko (PIN - Personal Identification Number), ki se skupaj z ID_Num zapiše v zaščiten del pomnilnika kartice. Omenjena kombinacija se lahko uporabi tudi kot ključ pri šifriranju komprimirane slike.

Tudi šifriranje in dešifriranje morata potekati izven kartice. Poznani so sicer učinkoviti algoritmi za uporabo v sami kartici (npr. Fiat-Shamir ali FEAL), vendar jih v naši aplikaciji ne moremo uporabiti. Težava je namreč v počasnem prenosu podatkov med bralno/pisalno napravo in kartico, ki je danes v večini primerov 9600 bit/s. Edina izjema je branje z optične površine, kjer so danes aktualne hitrosti okoli 100 Kbit/s. Branje 1 Mbyte podatkov iz optične površine tako kljub vsemu še vedno traja približno 80 s (npr. Canon OC), zato je toliko bolj razumljiva naša zahteva po učinkoviti kompresiji podatkov. Pri takih hitrostih je povsem nesmiselno že prebrane podatke ponovno pošiljati kartici v dešifriranje. Tako je šifrirni/dešifrirni algoritem implementiran kar v bralno/pisalni napravi sami. Ključ za dešifriranje prebere bralno/pisalna naprava iz EEPROM-a kartice. Po dešifriranju se izvede še dekompresija, nakar se slika prikaže.

Tu ne bomo opisovali dodatnih kontrol, ki se izvedejo na centralnem mestu pred branjem slike s kartice. Branje je namreč dovoljeno šele po uspešni primerjavi f(ID_Data) - podatki ID_Data se preberejo s kartice - z ID_Num, ki je shranjena na centralnem mestu. Spodaj sta opisana postopka kompresije/dekompresije in šifriranja/dešifriranja podatkov - slike, ki jo zapišemo na optično površino kartice.

Postopek pri izdaji kartice:

1. Uporabnikovi osebni podatki *ID_Data* se zapišejo v zaščiten del pomnilnika kartice
2. Izračuna se $ID\_Num = f(ID\_Data)$ in se shrani na centralnem mestu; algoritem *f* uporabniku ni poznan
3. Uporabnik izbere *PIN*, ki se skupaj z *ID_Num* zapiše v zaščiten del pomnilnika kartice, ključ za šifriranje postane $K = PIN + ID\_Num$, *PIN* se na centralnem mestu ne hrani
4. Uporabnikova slika - fotografija se digitalizira
5. Digitalizirana slika *DPict* se komprimira z eno od transformacijskih metod *T* (npr. DCT)
   $Cpr\_Pict = T(D\_Pict)$
6. Komprimirana slika *Cpr_Pict* se šifrira z algoritmom *C* ob uporabi ključa *K*
   $Cph\_Pict = C_K(Cpr\_Pict)$
7. Šifrirana slika *Cph_Pict* se zapiše na optično površino kartice

Postopek pri branju kartice ob identifikaciji:

1. Uporabnik vstavi kartico v bralno/pisalno napravo, izvrši se lokalno preverjanje *PIN* - uporabnik ga vnese preko posebne tipkovnice, dodane bralno/pisalni napravi ali pri zelo kvalitetnih tipih kartic (kot je npr. čipkartica Toshiba Smart Card) kar preko tipkovnice na sami kartici
2. Iz zaščitenega dela pomnilnika se preberejo identifikacijski podatki *ID_Data*, na centralnem mestu se izračuna *f(ID_Data)* in primerja s shranjenim *ID_Num*
3. Shranjena (šifrirana in komprimirana) slika se prebere z optične površine kartice v delovni pomnilnik bralno/pisalne naprave
4. Prebere se ključ za šifriranje *K*
5. Izvrši se dešifriranje slike na osnovi dešifrirnega algoritma *D* ob uporabi ključa *K*
   $Cpr\_Pict = D_K(Cph\_Pict)$
6. Izvrši se dekompresija slike z uporabo inverzne transformacije T^{-1} (npr. IDCT - Inverse DCT)
   $D\_Pict = T^{-1}(Cpr\_Pict)$
7. Digitalizirana slika *D_Pict* se prikaže na monitorju, lahko pa se dodatno primerja z morebitno shranjeno sliko na centralnem mestu

5. Zaključek

Z opisano rešitvijo zadovoljivo rešimo problem prenašanja velikih količin podatkov na majhnem prenosnem mediju (v velikosti kreditne kartice) za uporabo v identifikacijske in druge varnostne namene. Kartico, ki nam to omogoča, smo poimenovali opto-čipkartica.

Bistvena prednost, ki jo poleg že doslej znanih prednosti čipkartic in optičnih kartic prinaša njuna kombinacija, je seveda v veliki količini pomnilnika (tipa WORM) in hkratni procesorski moči kartice. Vgrajen mikroprocesor namreč omogoča sočasno uporabo že doslej znanih postopkov, uporabljenih v čipkarticah. Taka kartica je uporabna povsod tam, kjer se za identifikacijo uporablja večja količina podatkov. Prikazani primer s sliko (neke vrste elektronski potni list) ni edina možna uporaba. Z naraščanjem teleinformatskih storitev, ISDN-a in hišnih sistemov se odpirajo nove in nove možnosti.

Omeniti moramo vsekakor tudi določene pomanjkljivosti prikazane rešitve. Prva težava je v relativno počasnem prenosu podatkov med kartico in bralno/pisalno napravo (kljub 'hitremu' branju

100 Kbit/s z optične površine), s čimer je seveda dolgotrajen celoten identifikacijski postopek. Druga slabost pa je visoka cena bralno/pisalne naprave zaradi precejšnje kompleksnosti - naprava mora namreč pokrivati tako komunikacijo s čipi kakor tudi branje z optične površine. Pričakovati je, da bosta oba omenjena problema s hitrim razvojem kmalu zadovoljivo rešena in bo opto-čipkartica lahko široko uporabljana.

Literatura:

/1/ M.Bradeško, L.Pipan, Čipkartica - plastični mikroračunalnik, referat, Mipro 90

/2/ Ming L. Liou, Visual Telephony as an ISDN Application, IEEE Comm. Magazine, februar 1990

/3/ Stefano Cazzani, Carte intelligenti, Elettronica oggi, november 1989

/4/ Dokumentacija proizvajalcev čipkartic ADE, mbp GmbH, KryptoKom, NTT, Toshiba ipd.

/5/ Dokumentacija proizvajalca optičnih kartic Canon

MINIMALNE RAZDALJE MED GEOMETRIJSKIMI OBJEKTI

INFORMATICA 4/90

Štefan Žerdin
Niko Guid
Borut Žalik
Tehniška fakulteta Maribor,
Smetanova 17, 62000 Maribor

Keywords: computer graphics, minimum distance between geometric objects, composed Hermit's interpolation.

Povzetek: V članku predstavimo določevanje minimalnih razdalj med dvema točkama, točko in krivuljo, točko in ploskvijo, dvema krivuljama, krivuljo in ploskvijo ter dvema ploskvama. Za krivulje in ploskve uporabimo parametrično kubično Hermitovo predstavitev. V nadaljevanju opišemo metode za določanje tistih odsekov oz. tistih krp ploskve, ki vsebujejo točko, ki je najbližje sosednjemu geometrijskemu objektu in tako močno zmanjšamo čas izračuna.

Minimum distances between geometric objects:
Abstract: Determination of the minimum distance between two points, the point and the curve, the point and the surface, two curves, the curve and the surface and two surfaces are presented in this paper. The cubic parametric Hermite interpolation for curve and surface representation is used. Then a method for determination of those curve segments end those surface patches which contains the points which are closest to the neighbour geometric object are explained. This is the reason why computation time is reduced.

1 UVOD

Med geometrijskimi objekti obstajajo različne relacije. Za skupino objektov se lahko npr. vprašamo, kateri objekt je najbližji oz. kateri je najbolj oddaljen od dane točke, kateri objekti so delno ali v celoti zakriti z ostalimi objekti, kateri objekt je največji oz. najmanjši v skupini, kateri objekti so v notranjosti določenih objektov itd. Te relacijske lastnosti so zelo pomembne v geometrijskem modeliranju, pogosto pa morajo biti rešene tudi v računalniški grafiki in sistemih CAD/CAM. Osnovo teh relacijskih lastnosti pa predstavljata dva problema. To sta problem presečišč in problem minimalne razdalje, ki ga bomo opisali v članku.

V računalniški grafiki krivulje in ploskve zaradi mnogih prednosti opisujemo v parametrični obliki [GUID88]. Najpopularnejše so Hermitove krivulje in ploskve, krivulje in ploskve B-zlepkov, Bézierove krivulje in ploskve, neuniformne racionalne krivulje in ploskve B-zlepkov ipd. Program, ki smo ga izdelali, uporablja le parametrične kubične Hermitove krivulje in ploskve. Napisan je v programskem jeziku pascal, grafično podprt s standardom GKS, teče pa pod sistemom VMS.

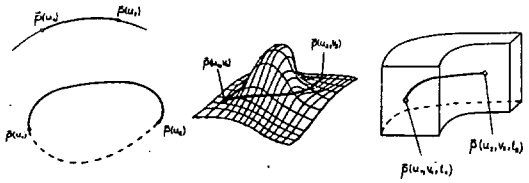
2 NAJMANJŠA RAZDALJA MED DVEMA TOČKAMA

Izračun minimalne razdalje med dvema točkama je hkrati enostaven in zapleten. Če računamo minimalno razdaljo med dvema poljubnima točkama P_1 in P_2 v prostoru, je izračun seveda preprost. Pojem minimalna razdalja je v tem primeru le formalen, saj med dvema točkama obstaja le ena razdalja.

Izračun postane bolj zapleten, če se točki nahajata na krivulji, ploskvi ali pa v telesu in iščemo najkrajšo pot med njima [MORT85]. Če se točki nahajata na krivulji in če moramo potovati po krivulji, preide izračun minimalne razdalje med točkama v izračun dolžine loka med

njima (slika 1a). V primeru, ko je krivulja nesklenjena, je izračun s tem končan. Ko pa je krivulja sklenjena, moramo izračunati dolžino obeh lokov in izbrati krajšega. Za izračun dolžine loka krivulje obstajajo direktne metode.

Iskanje minimalne razdalje med dvema točkama na ploskvi, ko potujemo po ploskvi, ali v telesu, ko se gibljemo znotraj telesa, zahteva v splošnem iskanje prostorske krivulje skozi podani točki (slika 1b,c). Direktnih metod za reševanje ni, zato uporabljamo iterativne metode. Rešitev lahko predstavlja ena ali več prostorskih krivulj, ki določajo minimalno razdaljo oz. najkrajšo pot med točkama.

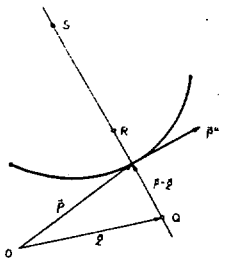


a) b) c)

Slika 1 Minimalna razdalja med točkama na geometrijskem objektu

3 NAJMANJŠA RAZDALJA MED TOČKO IN KRIVULJO

Minimalna oddaljenost točke Q od krivulje $\vec{p}$ ($\vec{p} = \vec{p}(u)$) je določena z vektorjem ($\vec{p} - \vec{q}$) od točke do krivulje, ki je pravokoten na tangentni vektor $\vec{p}^u$.



Slika 2 Minimalna razdalja med točko in krivuljo

Minimalna razdalja med točko Q in krivuljo $\vec{p}$ je določena z enačbo:

$$d_{min} = |\vec{p} - \vec{q}|. \qquad (1)$$

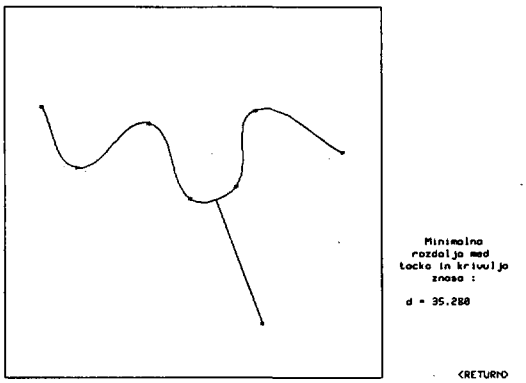
Točka na krivulji $\vec{p}$, ki je najbližje dani točki Q s krajevnim vektorjem $\vec{q}$, zadošča pogoju [MORT85]:

$$(\vec{p} - \vec{q})\,\vec{p}^u = 0. \qquad (2)$$

Pogoj minimuma (en. 2)je le potrebni pogoj, zato moramo kontrolirati obliko krivulje v točki rešitve. Na sliki 2 vidimo, da predstavljata razdalji točke Q in R do krivulje $\vec{p}$ minimalno razdaljo, razdalja točke S do krivulje $\vec{p}$ pa maksimalno razdaljo točke do krivulje.

Pogoj minimuma je lahko brez rešitve, lahko pa je rešitev tudi več. Če je krivulja $\vec{p}$ v točki rešitve lokalno konveksna glede na točko Q, predstavlja dobljena rešitev minimalno razdaljo točke Q do krivulje $\vec{p}$. V primeru, ko krivulja v točki rešitve ni lokalno konveksna glede na točko Q, oz. če pogoj minimuma nima rešitve, določimo razdalji točke Q do krajišč krivulje $\vec{p}$. Med izračunanimi razdaljami izberemo najmanjšo in ta predstavlja minimalno razdaljo točke Q do krivulje $\vec{p}$.

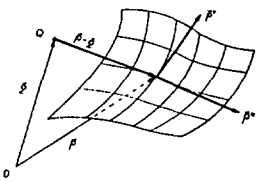
Primer določitve minimalne razdalje med točko in krivuljo z razvitim programom je razviden iz slike 3. V programu smo za reševanje nelinearne enačbe uporabili Newtonovo metodo [BOHT78].



Slika 3 Najmanjša razdalja med točko in krivuljo določena s programom

4 NAJMANJŠA RAZDALJA MED TOČKO IN PLOSKVIJO

Podobno kot med točko in krivuljo, je tudi minimalna razdalja med točko Q in ploskvijo $\vec{p}$ ($\vec{p} = \vec{p}(u,v)$) določena z vektorjem ($\vec{p} - \vec{q}$) od točke do ploskve, ki je hkrati pravokoten na oba tangentna vektorja ploskve $\vec{p}^u$ in $\vec{p}^v$ (slika 4).



Slika 4 Minimalna razdalja med točko in ploskvijo

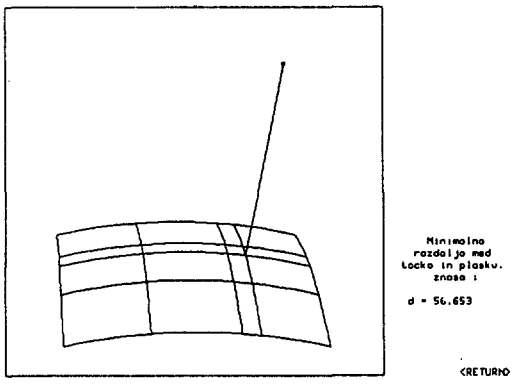
Velja enačba 1 pri pogoju:

$$(\vec{p} - \vec{q}) \times (\vec{p}^u \times \vec{p}^v) = \vec{0}. \qquad (3)$$

Pri numeričnem reševanju sistema nelinearnih enačb, ki ga dobimo iz pogoja minimuma, lahko nastopijo podobni primeri kot pri določevanju minimalne razdalje med točko

in krivuljo. Če je ploskev $\vec{p}$ v točki rešitve lokalno konveksna glede na točko Q, predstavlja določena razdalja minimalno razdaljo točke Q do ploskve $\vec{p}$. Če pa ploskev v točki rešitve ni lokalno konveksna glede na točko Q, oz. če sistem nelinearnih enačb nima rešitev, določimo minimalne razdalje točke Q do robnih krivulj ploskve. Izmed izračunanih rešitev izberemo najmanjšo in ta predstavlja minimalno razdaljo točke Q do ploskve $\vec{p}$.

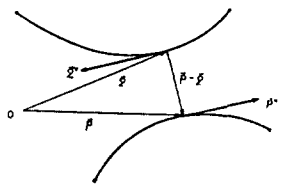
Primer določitve najmanjše razdalje med točko in ploskvijo z razvitim programom vidimo na sliki 5. V programu smo za reševanje sistemov nelinearnih enačb uporabili Newtonovo metodo ([BOHT78], [WAIT79]).



Slika 5 Minimalna razdalja med točko in ploskvijo določena s programom

5 NAJMANJŠA RAZDALJA MED DVEMA KRIVULJAMA

Minimalno razdaljo med krivuljama $\vec{p}$ ($\vec{p} = \vec{p}(u)$) in $\vec{q}$ ($\vec{q} = \vec{q}(v)$) določa vektor $(\vec{p} - \vec{q})$, ki mora biti hkrati pravokoten na tangentna vektorja obeh krivulj $\vec{p}^u$ in $\vec{q}^v$ (slika 6).



Slika 6 Minimalna razdalja med dvema krivuljama

Minimalna razdalja je določena z enačbo 1 pri pogoju:

$$(\vec{p} - \vec{q})\ \vec{p}^u = 0$$

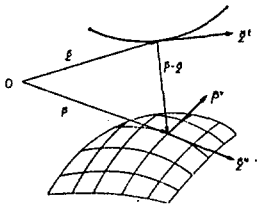
in

$$(\vec{p} - \vec{q})\ \vec{q}^v = 0. \qquad (4)$$

Pogoja minimuma morata biti izpolnjena hkrati, zato imamo spet opravka z reševanjem sistema nelinearnih enačb. Če sta obe krivulji v točkah rešitve konveksni, predstavlja dobljena rešitev minimalno razdaljo med krivuljama $\vec{p}$ in $\vec{q}$. V primeru, ko ena ali obe krivulji v točki rešitve ni lokalno konveksna, oz. ko sistem nelinearnih enačb nima rešitev, določimo minimalne razdalje krajišč krivulje $\vec{p}$ do krivulje $\vec{q}$ in krajišč krivulje $\vec{q}$ do krivulje $\vec{p}$. Izmed določenih razdalj izberemo minimalno in ta predstavlja minimalno razdaljo krivulje $\vec{p}$ do krivulje $\vec{q}$.

6 NAJMANJŠA RAZDALJA MED KRIVULJO IN PLOSKVIJO

Kakor v prejšnjih primerih, je tudi razdalja med krivuljo $\vec{q}$ ($\vec{q} = \vec{q}(t)$) in ploskvijo $\vec{p}$ ($\vec{p} = \vec{p}(u,v)$) določena z vektorjem $(\vec{p} - \vec{q})$, ki mora biti hkrati pravokoten na tangentni vektor krivulje $\vec{q}^t$ in tangentna vektorja $\vec{p}^u$ in $\vec{p}^v$ ploskve $\vec{p}$ (slika 7).



Slika 7 Minimalna razdalja med krivuljo in ploskvijo

Minimalna razdalja je določena z enačbo 1 pri pogoju:

$$(\vec{p} - \vec{q}) \times (\vec{p}^u \times \vec{p}^v) = \vec{0}$$

in

$$(\vec{p} - \vec{q})\ \vec{q}^t = 0. \qquad (5)$$

Da bi določili minimalno razdaljo med krivuljo $\vec{q}$ in ploskvijo $\vec{p}$, moramo poiskati vrednosti parametrov u, v in t, ki izpolnjujejo pogoj minimuma, oz. rešiti sistem nelinearnih enačb.

Če sta krivulja $\vec{q}$ in ploskev $\vec{p}$ v točkah rešitve lokalno konveksni, predstavlja dobljena rešitev minimalno razdaljo krivulje $\vec{q}$ do ploskve $\vec{p}$. Ko pa krivulja $\vec{q}$ oz. ploskev $\vec{p}$ v točki rešitve ni lokalno konveksna, oz. ko sistem nelinearnih enačb nima rešitev, poiščemo minimalne razdalje krivulje $\vec{q}$ do robnih krivulj ploskve $\vec{p}$. Najmanjša med njimi določa minimalno razdaljo krivulje do ploskve.

7 NAJMANJŠA RAZDALJA MED DVEMA PLOSKVAMA

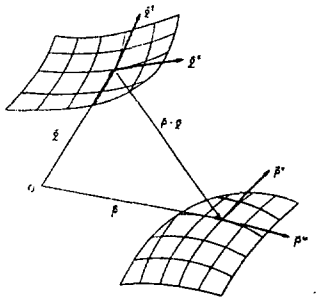
Najbolj kompleksen primer določevanja minimalne razdalje je določevanje minimalne razdalje med ploskvama $\vec{p}$ ($\vec{p} = \vec{p}(u,v)$) in $\vec{q}$ ($\vec{q} = \vec{q}(s,t)$). Le-ta je določena z vektorjem $(\vec{p} - \vec{q})$, ki mora biti hkrati pravokoten na tangentna vektorja $\vec{p}^u$ in $\vec{p}^v$ ploskve $\vec{p}$ ter tangentna vektorja $\vec{q}^s$ in $\vec{q}^t$ ploskve $\vec{q}$ (slika 8).

Minimalna razdalja je določena z enačbo 1 pri pogoju:

$$(\vec{p} - \vec{q}) \times (\vec{p}^u \times \vec{p}^v) = \vec{0}$$

in

$$(\vec{p} - \vec{q}) \times (\vec{q}^{s} \times \vec{q}^{t}) = \vec{0}. \qquad (6)$$

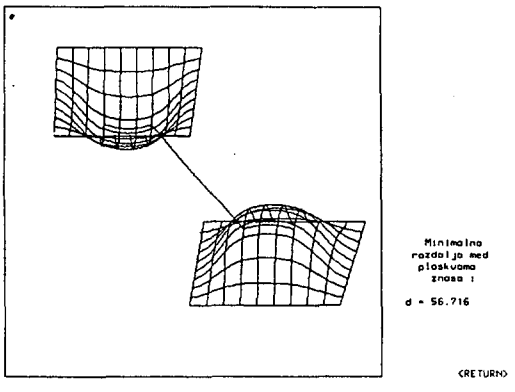


Slika 8 Minimalna razdalja med dvema ploskvama

Da bi določili minimalno razdaljo med ploskvama $\vec{p}$ in $\vec{q}$, moramo torej rešiti sistem nelinearnih enačb, ki ga dobimo iz zgornjega pogoja.

Če sta obe ploskvi v točkah rešitve konveksni, predstavlja dobljena rešitev minimalno razdaljo ploskve $\vec{p}$ do ploskve $\vec{q}$. Če pa katera ploskev (ali pa obe hkrati) v točki rešitve ni lokalno konveksna, oz. če sistem nelinearnih enačb nima rešitve, določimo minimalne razdalje robnih krivulj prve ploskve do druge in obratno. Najmanjša med njimi bo določala minimalno razdaljo med dvema ploskvama.

Določitev najmanjše razdalje med dvema ploskvama z razvitim programom je razvidna iz slike 9.



Slika 9 Najmanjša razdalja med dvema ploskvama določena s programom

8 OPIS UPORABLJENE PREDSTAVITVE GEOMETRIJSKIH OBJEKTOV

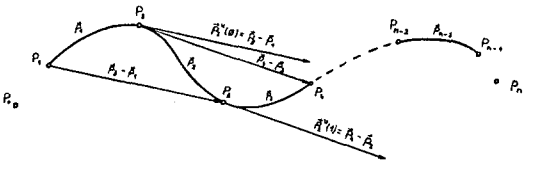
V programu, ki smo ga razvili za določevanje minimalnih razdalj med geometrijskimi objekti, smo uporabili sestavljeno parametrično kubično Hermitovo interpolacijo, ki smo jo nekoliko priredili.

i-ti odsek sestavljene parametrične kubične Hermitove krivulje je določen z enačbo 7:

$$\vec{p}_i(u) = (2u^3 - 3u^2 + 1)\,\vec{p}_i(0) + (-2u^3 + 3u^2)\,\vec{p}_i(1) + (u^3 - 2u^2 + u)\,\vec{p}_i^{u}(0) + (u^3 - u^2)\,\vec{p}_i^{u}(1), \quad i \in [1, n-2]. \qquad (7)$$

Ker je podajanje tangentnih vektorjev za vsak odsek zelo neugodno, lahko uporabimo za določitev le-teh Coons-Ahujin postopek, ki nam zagotavlja v stičiščih odsekov zveznost C^2 [GUID88]. Ta postopek pa zahteva invertiranje matrike, katere red je odvisen od števila interpolacijskih točk.

Zato smo uporabili drugo metodo, pri kateri tangentne vektorje v stičiščih odsekov izračunamo iz sosednjih interpolacijskih točk, kar pa nam zagotavlja samo zveznost reda C^1. Pogoj za zveznost C^1 je enakost tangentnih vektorjev dveh sosednjih odsekov v točki P_i. Interpolacijo krivulje po tej metodi kaže slika 10.



Slika 10 Interpolacijska krivulja

Tako velja:

$$\vec{p}_i^{u}(0) = \vec{p}_{i+1} - \vec{p}_{i-1}$$

$$\vec{p}_i^{u}(1) = \vec{p}_{i+2} - \vec{p}_i, \qquad i \in [1, n-2]. \qquad (8)$$

Vendar pa tako določeni tangentni vektorji niso direktno uporabni. Velja namreč, da je s smerjo tangentnega vektorja, oz. smerjo vektorja odvoda, določena smer gibanja točke po krivulji, velikost tangentnega vektorja pa predstavlja hitrost, s katero se točka giblje na začetku oz. koncu odseka [GUID88]. Če je ta hitrost prevelika, lahko krivulja močno zaniha, interpolacija pa je zelo slaba. Zato normirana tangentna vektorja pomnožimo z razdaljo med začetno in končno točko odseka. Enačba za i-ti odsek krivulje dobi tako naslednjo obliko [ŽERD90]:

$$\vec{p}_i(u) = (2u^3 - 3u^2 + 1)\,\vec{p}_i + (-2u^3 + 3u^2)\,\vec{p}_{i+1} + (u^3 - 2u^2 + u)\,\frac{(\vec{p}_{i+1} - \vec{p}_{i-1})}{|\vec{p}_{i+1} - \vec{p}_{i-1}|}\,d_i + (u^3 - u^2)\,\frac{(\vec{p}_{i+2} - \vec{p}_i)}{|\vec{p}_{i+2} - \vec{p}_i|}\,d_i, \qquad (9)$$

kjer je d_i razdalja med začetno in končno točko odseka.

Podobno kot enačbo za interpolacijo krivulje, smo izpeljali tudi enačbo za interpolacijo ploskve. Tudi v tem primeru smo za tangentne vektorje uporabili enak postopek kot pri krivuljah. Tangentne vektorje robnih

krivulj krpe, ki so določeni z interpolacijskimi točkami, smo najprej normirali, nato pa smo jih pomnožili z razdaljo med začetno in končno točko robne krivulje krpe. Sukalne vektorje, ki vplivajo le na notranjost krpe, smo postavili na 0. Posledica tega je, da je površina krpe bolj ravna v okolici kontrolnih točk [YAMA88]. Končna oblika enačbe krpe ploskve je torej naslednja:

$$\vec{p}_{ij}(u,v) = [\, F_{1u}\vec{p}_{i,j} + F_{2u}\vec{p}_{i+1,j} + F_{3u}\vec{k}^{u}_{00} + F_{4u}\vec{k}^{u}_{10}\,]\, F_{1v} +$$
$$[\, F_{1u}\vec{p}_{i,j+1} + F_{2u}\vec{p}_{i+1,j+1} + F_{3u}\vec{k}^{u}_{01} + F_{4u}\vec{k}^{u}_{11}\,]\, F_{2v} +$$
$$[\, F_{1u}\vec{k}^{v}_{00} + F_{2u}\vec{k}^{v}_{10}\,]\, F_{3v} + [\, F_{1u}\vec{k}^{v}_{01} + F_{2u}\vec{k}^{v}_{11}\,]\, F_{4v}, \qquad (10)$$

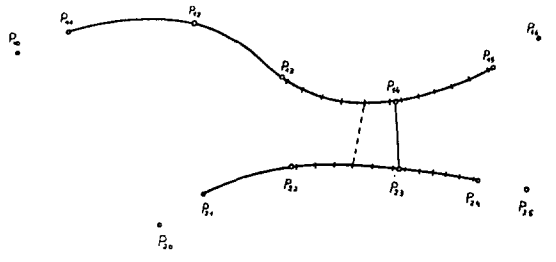
kjer je:

$$\vec{k}^{u}_{00} = \frac{\vec{p}_{i+1,j} - \vec{p}_{i-1,j}}{|\vec{p}_{i+1,j} - \vec{p}_{i-1,j}|}\, |\vec{p}_{i,j} - \vec{p}_{i+1,j}|, \qquad (11)$$

F_{1u}, F_{2u}, F_{3u}, F_{4u} so Fergusonove povezovalne funkcije v smeri parametra u, F_{1v}, F_{2v}, F_{3v}, F_{4v} pa Fergusonove povezovalne funkcije v smeri parametra v.

Analogno kot vektor $\vec{k}_{00}$, so določeni tudi ostali vektorji $\vec{k}$. Predstavljajo torej tangentni vektor v smeri parametra u oz. v v enem oglišču, ki je normiran in pomnožen z razdaljo med ogliščema krpe, ki določata robno krivuljo, na kateri se omenjeni vektor nahaja. Med sosednjimi krpami ploskve imamo zveznost reda C^1.

Zaradi mnogih prednosti v računalniški grafiki sestavimo krivulje iz odsekov, ploskve pa iz krp. Zato moramo za določitev minimalne razdalje med objektoma določiti razdalje med vsemi kombinacijami odsekov oz. krp na obeh objektih, kar pa lahko dovolj hitro storimo le, ko jih je dovolj malo. Ugodno bi bilo pred samim izračunom minimalne razdalje oceniti, katera odseka oz. krpi sta najbližji in s tem izločiti veliko nepotrebnih računanj. Primer ocenitve najbližjih odsekov dveh interpolacijskih krivulj je razviden iz slike 11:



Slika 11 Določitev najbližjih odsekov krivulj

Najprej poiščemo interpolacijski točki, ki sta najbližji. V našem primeru sta to točki P_{14} in P_{23}. Zdaj določimo razdalje med "karakterističnimi" točkami odsekov, ki se nahajajo levo in desno od točk P_{14} in P_{23}. Nato med določenimi razdaljami poiščemo najmanjšo in ker vemo, na katerih odsekih se "karakteristični" točki nahajata, smo tako ocenili, katera odseka sta najbližja. Zdaj bomo minimalno razdaljo določali le med tema dvema odsekoma, kar zelo zmanjša čas, potreben za določitev minimalne razdalje med dvema krivuljama.

Včasih pa se zgodi, da najbližji interpolacijski točki ne določata najbližjih odsekov. Zato shranimo več parov interpolacijskih točk (odvisno od števila interpolacijskih točk krivulj), ki so najbližje. Nato poiščemo minimalno razdaljo med "karakterističnimi" točkami odsekov, ki jih shranjene interpolacijske točke določajo. Tako dobimo pravilno oceno o odsekih krivulj, ki sta najbližja.

9 ZAKLJUČEK

Čas računanja, potreben za določitev minimalne razdalje med geometrijskima objektoma, je odvisen od izbranih objektov. Najmanjši je v primeru določevanja minimalne razdalje med točko in krivuljo, največji pa pri določevanju minimalne razdalje med dvema ploskvama.

Če npr. določamo minimalno razdaljo med dvema ploskvama, bomo v najbolj neugodnem primeru morali določiti minimalno razdaljo med ploskvama, minimalne razdalje vseh robnih krivulj prve ploskve do druge in obratno, razdalje končnih točk robnih krivulj prve ploskve do druge ploskve in obratno, razdalje med robnimi krivuljami obeh ploskev, razdalje oglišč ene krpe do robnih krivulj druge krpe in razdalje med oglišči krpe. Vidimo, da smo morali v tem primeru določiti minimalne razdalje med vsemi kombinacijami geometrijskih objektov, nekatere celo večkrat.

Število kontrolnih točk geometrijskega objekta vpliva na čas, potreben za določitev minimalne razdalje zato, ker moramo najprej oceniti, katera odseka oz. krpi sta najbližja.

Problem določevanja minimalne razdalje med geometrijskimi objekti teoretično ni zahteven, pri numeričnem določanju minimalne razdalje pa lahko nastopijo težave zaradi časa, ki je potreben za reševanje. Uspešnost in uporabnost algoritmov za določevanje minimalnih razdalj med geometrijskimi objekti je predvsem odvisna od uspešnosti algoritmov za izločanje nepotrebnih računanj in konvergence numeričnih metod za reševanje sistemov nelinearnih enačb.

LITERATURA

BOHT78 Bohte, Z., "Numerične metode", Društvo matematikov, fizikov in astronomov SRS, Ljubljana, 1978

GUID88 Guid, N., "Računalniška grafika", Tehniška fakulteta Maribor, Maribor, 1988

MORT85 Mortenson, M. E., "Geometric Modeling", John Wiley & Sons, Ltd., New York, 1985

WAIT79 Wait, R., "The numerical solution of algebraic equations", John Wiley & Sons, Ltd., Chichester, New York, Brisbane, Toronto, 1979

YAMA88 Yamaguchi, F., "Curves and surfaces in computer aided geometric design", Springer-Verlag, Heidelberg, 1988

ŽERD90 Žerdin, Š., "Minimalne razdalje med geometrijskimi objekti", diplomsko delo, Tehniška fakulteta Maribor, Maribor, 1990

REAL-TIME EXECUTIVES FOR EMBEDDED MICROPROCESSOR APPLICATIONS

INFORMATICA 4/90

Barbara Koroušić*
Jim E. Cooling
Loughborough University of Technology,
Loughborough, UK
Peter Kolbezen
*»Jožef Stefan« Institute,
Ljubljana

Keywords: real-time executive, microprocessor, application.

Real-time embedded systems are controlled by microcomputers. They are almost invisible for the users, but have a great and powerful role in the system. Therefore, the operating system support must be provided to achieve reliable, predictable and fast behavior of the computer and of the whole system.

In this paper, we introduce the basic theoretical aspects, which should be understood by the user and programmer of embedded applications. Emphasis is given on the scheduling policies, which are most frequently used for 'soft' real-time embedded systems. Some of the scheduling approaches which could be used for 'hard' real-time systems are also reviewed.

Namenski operacijski sistemi za delo v realnem času - Članek opisuje namenske sisteme za delo v realnem času. Opisane so glavne funkcije jedra operacijskega sistema, s poudarkom na različnih principih razvrščanja opravil.

1 Real-Time Embedded Systems

Embedded systems are dedicated systems which are under the control of one or more microcomputers, that are connected directly to the physical equipment. 'Embedded' microcomputers run a specific real-time application, which is typically not reprogrammable by the user. The user even need not know that computers exist inside the embedded system [Rea86]. Typical examples of embedded real-time systems are monitoring and control systems, industrial robots, aircraft avionics, and the like [Sto87].

An 'embedded' application executes several tasks on data which are obtained from the sensors of the embedded microcomputers and outputs results to the environment via output devices such as actuators, displays, etc.

A task is a routine (a sequence of instructions) which is a logically separate unit of an embedded application. It can suspend itself, and when it is reactivated it proceeds with its execution from the point where it was previously suspended. Because several tasks can be active at the same time, we call such tasks processes [Bro90].

Application consists of periodic and aperiodic tasks. Periodic (also called synchronous) tasks accept and process data repeatedly, in predefined and regular intervals, while aperiodic (asynchronous or event-driven) tasks are activated at random times.

In real-time applications, the duration of each execution of a periodic or aperiodic task is limited with its deadline. Deadline is a time period, in which a task has to finish with its execution. In some applications, deadlines of periodic tasks can be assumed to be the same as their periods.

Real-time systems are distinguished into two classes (soft' and 'hard'). 'Soft' real-time systems execute applications, in which tasks have to meet their deadlines. If some event prevents meeting of some deadline, the functioning of a systems will be seriously degraded, but no catastrophe will cause. Meanwhile, in 'hard' real-time systems, missing of 'safety' critical task's deadline is unacceptable [BS88b].

While signals from the environment are often generated in parallel, tasks must operate concurrently to respond within specified and finite intervals. Tasks must be ready all the time to respond to the external events. In real-time systems no external event may be neglected, because a task that handles it is not active at that moment. Most current real-time microcomputer systems have limited resources, including the number and speed of the processors. Therefore, the concurrency of the processes can not be achieved by allocating each process to one processor and execute them in parallel. The scheduling techniques are used to allocate processes to the limited resources (e.g. processors) so that their deadlines are met.

Different hardware architectures are used for embedded systems. Uniprocessor systems have single processors that execute all the tasks of the applications. The scheduling techniques are used to achieve fair and timely access of all tasks to the processor. Multiprocessor systems are distributed systems of more processors, that execute more tasks in parallel. The scheduling techniques have to be used when the number of processes that have to be executed concurrently exceeds the number of available resources.

2 Embedded Real-Time Operating Systems

An embedded application consists of many tasks that are logically separate units. It can be implemented by using interrupt driven programs (routines). Each interrupt routine can be regarded as a separate task. The programmer is responsible for all aspects of task handling. He (or she) must be experienced in programming and needs detailed knowledge of machine hardware to achieve correct, safe and timely operation of an embedded application [Lei88].

Current microcomputers have powerful computation capabilities, speed of operation and memory capacity. Thus OS (Operating System) support is a feasible solution for the development and implementation of embedded applications.

'Embedded' operating systems must support efficient and reliable operations. The most important aspects of embedded real-time operating systems are[SG90]:

- predictability of performance,
- high degree of schedulability,
- stability under transient overload.

The most important aspect of the real-time operating systems is not extremely fast response to the external events. It is more important to give the logically correct response in predictable time interval. The scheduling techniques have to provide the high degree of schedulability. They must enable the meeting of deadlines to as many as possible tasks. It does not mean that the processors' utilization should be as high as possible. The processor utilization

is the fraction of processor time spent in the execution of the task set [LL73]. The real-time operating systems must provide the safety behavior of the system. In the case of the transient overload, the scheduler have to decide which task is not so important for the application and can be suspended for some time.

2.1 Real-Time Executive

The central part of an embedded real-time operating system software is the executive. It is primarily responsible for the management and control of computer resources. User processes call system resources via the executive. System resources can be of hardware (e.g. processors, memory, I/O devices) or software (e.g. system tasks) type. Mutual exclusion prevents damage or corruption to any shared resources. The executive controls scheduling of the processes and mutual exclusion. It also provides facilities to allow different processes to synchronize and communicate between one another.

Actually the executive calls on facilities provided by the kernel. It behaves as the manager, and the kernel as the executor [Coo90].

In real-time systems, two factors are particularly important in evaluating the performance of an executive [Tes84]:

- interrupt latency (the task's reaction time [Coo90]),
- context switching time (how long it takes to stop one task and start another).

Interrupt latency and context switching time is typically in the range 20-300 microseconds [Lei88].

Executives may be provided on ROM (Read Only Memory Chip) designed to be added to the developer's microcomputer boards [Tes84]. The executive's code is written to be position-independent. It is able to operate properly regardless of the address locations the ROM chip occupies in the system. Other executives are provided on diskette to allow code modifications during development. The resulting programs may operate at specific memory locations within a given computer system.

2.1.1 Mutual Exclusion

When two or more user or system processes execute concurrently, the accesses to the shared resources need to be mutually exclusive. Otherwise the processes may not execute in complete safety. Usually, accesses to the processors are controlled by the scheduler.

When a process is accessing a shared resource, it is said to be in a critical region [Lei88]. When a process enters the critical region, it must be carried through to completion by one process only. Before a process accesses a shared resource, it must check whether any other process is in the critical region for the same resource. If the resource is free, the process may enter the region, but must indicate that the resource is in use. When it finishes, it must signal that a resource is free again.

The indication that the resource is in use or it is free can be implemented by using common variables (flags), semaphores or monitors.

a) Flags

Dekker's algorithm is an application of flag signalling. Each process that accesses a shared resource has its own variable (a flag) to indicate whether it uses the resource or not. These variables are common to all processes, but only its owner (a process) can set or clear it. The algorithm uses one more common variable ('turn') to signal which process has priority to access the shared resource (**Figure 1**).

```
PROGRAM DekkersAlg;

CONST: numOfProcesses = 2;
VAR
  flag: ARRAY [1..numOfProcesses] OF BOOLEAN;
  turn: BOOLEAN;

PROCEDURE Process (processNum: 1..numOfProcesses;
                   exitCondition: BOOLEAN);
  VAR secondProcessNum: 1..numOfProcesses;
  BEGIN
    secondProcessNum:= (processNum MOD 2) + 1;
    REPEAT
      flag[processNum]:= TRUE;
      WHILE flag[secondProcessNum] DO
        IF turn = secondProcessNum THEN
          flag[processNum]:= FALSE;
          WHILE turn = secondProcessNum DO
          END;
          flag[processNum]:= TRUE;
        ENDIF;
      END;
      UseResource;
      turn = secondProcessNum;
      flag[processNum]:= FALSE;
      DoRest;
    UNTIL exitCondition;
  END Process;

BEGIN
  FOR i:= 1 TO numOfProcesses DO
    flag[i]:= FALSE
  END;
  turn:= 1;
  ExecuteConcurrentlyAllProcesses;
END DekkersAlg;
```

Figure 1: Dekker's algorithm

The testing and setting of the common variables must be uninterruptible. Otherwise, the problem may appear when two processes are testing the flags simultaneously, and there is not enough time to write the proper state into variables. In such situation, deadlock may happen. A process is deadlocked whenever it is permanently denied a resource it needs to proceed [Lei88].

Unfortunately, the Dekker's algorithm is not very useful for real-time systems. It can be used for mutual exclusion of common resources between two or three processes. If there are more than three processes, it takes too much time to check all the common variables.

b) Semaphores

Mutual exclusion can be implemented by using semaphores [Lei88]. Each resource has its own semaphore to indicate the occupation of the resource. Semaphores are common to all processes that access to the common resources. If a process needs to access to a resource, it has to wait on semaphore. It is suspended, if a resource is not free. When a process finishes with its occupation of a resource, it has to release a semaphore and the process that waits for it.

A semaphore is an integer variable, which can be of a nonnegative value. Three basic operations may change the value of a semaphore s:

1. Initialization of the semaphore s.

2. Waiting on the semaphore *s*:

```
PROCEDURE P(s);
(* P was originally used by Dijkstra
   ('passeren', means 'to pass') *)
  BEGIN
    IF s > 0 THEN
      s:= s - 1
    ELSE
      (* Suspends process which executed P(s)
*)
      SuspendProcess;
  END P(s);
```

3. Signalling with the semaphore *s*:

```
PROCEDURE V(s);
(* V was originally used by Dijkstra
   ('vrygeven', means 'to release' ) *)
  VAR process: tProcess;
  BEGIN
    (* Checks if a process has been suspended
by
       a previous P(s) *)
    IF process = SuspendedProcess THEN
      Wake(process);
    ELSE
      s = s + 1
    ENDIF
  END V(s);
```

A semaphore which can only take the values 0 and 1 is called a binary semaphore. In the example (**Figure 2**) a binary semaphore is used to implement mutual exclusion between two processes. A general or counting semaphore is more general, and is usually used for counting.

```
PROGRAM MutualExclusionByUsingSemaphores;

  CONST numOfProcesses = 2;
  VAR s: 1..numOfProcesses;   (* Each process has
its own semaphore *)

  PROCEDURE Process (processNum: 1..numOfProcesses;
exitCondition: BOOLEAN);
    BEGIN
      REPEAT
        P(s);
        UseResource;
        V(s);
        DoRest;
      UNTIL exitCondition;
    END Process;

BEGIN
  s:= 1;
  ExecuteConcurrentlyAllProcesses
END MutualExclusionByUsingSemaphores;
```

Figure 2: Mutual exclusion using a semaphore

A programmer still has to confront the details of making the procedures V(s) and P(s) uninterruptedly.

Monitors

Mutual exclusion provided by using semaphores can not be guaranteed if a programmer makes some mistake (e.g. after V(s) forgets to call P(s)). Therefore, a process (a monitor) which cares about the operations that are needed on the resource is very useful tool in the building of real-time operating systems [Lei88]. If a programmer wants to access the resource, the needed operation will be done by the monitor. The monitor handles the semaphores, and the details are hidden from the programmer.

A task that has to use a shared resource, needs 'a key' to access the resource. The key is managed by the monitor. A task can not access directly to the resource. It has to send the request to the monitor. By having one key, only its owner can access the resource at any one time. When it finishes with accessing the resource, it sends the key to the monitor.

If the monitor receives the request for a key, and it does not possess it, it generates the signal of 'wait' type. When a key is released, the monitor activates the task, which has been suspended, because a key was not available. Tasks that wait for a key are ordered in a priority queue.

2.1.2 Interprocess Communication

The interprocess communication must be provided by the real-time executive. The processes that are executing concurrently on a single processor or on distributed processors have to communicate and synchronize their activities. Three different types of interprocess communication are used in the real-time executives(task synchronization, data transfer and task synchronization with data transfer).

a) Task Synchronisation

In some embedded applications, processes need to synchronize their activities in regard of events that include time related factors. Synchronization can be achieved by using signals [Coo90].

Three types of signals ('send', 'wait' and 'check') are usually used for task synchronization. They are implemented by using procedures, whose parameters are signal names. The implementation of signals is similar as for semaphores.

A task that wants to synchronize its action with another tasks, transmits a signal of type 'send'. If a task needs for its execution a signal from another task, it has to send a 'wait' signal. A task which generates a 'send' signal will be suspended if no task waits for it. The same situation happens when a process waits for a signal which has not been sent by another process. The process which has been suspended continues with its execution after the signal transmission (in the first case) or reception (in another case). A signal of 'check' type can be sent to verify if some process is sending a signal of type 'send'. If a signal has not been sent, a process that waits for this signal can decide, in itself, whether it will suspend or continue with its execution.

The executive is responsible for task synchronization. A process that wants to send a signal to some another process actually sends it to the executive.

b) Data transfer

Real-time executive has the facility of data transfer between tasks. A process can send data periodically or at random times. Usually, two data storage mechanisms are used, 'pools' and 'channels' [Coo90].

'Pools' are used for storing data common to a number of processes. A read-write memory (RAM) can be used to implement the pool. Because a pool is a shared resource, mutual exclusion must be provided to control its usage.

A 'channel' is used for the communication between two processes. It is realized as a pipe, and the insertion and extraction from it

can proceed asynchronously. The data (or pointers to data) from the channel are read by using a FIFO (First In First Out) mechanism.

The pipe can be implemented by using a queue data structure or a circular buffer. A queue is limited only by the available memory space. Its size is not fixed, but the large size causes long delay between the data storing and reading.

A circular buffer has a fixed size which is defined at its creation time. Pointers are used to define the reading and writing position in the buffer.

c) Data Transfer with Task Synchronization

On some occasions, processes have to use data which are associated with events. For this the 'mailbox' mechanism provides the required synchronization facility and data storage area [Coo90]. It uses signals for synchronization, and storage for data. A mailbox can be used by many processes.

When a process wants to send some data to another one, it sends a signal to the executive. The process that has sent the signal to the executive is suspended until another process reads the data from the mailbox. Actually, the receiving process sends the signal of 'wait' type to the executive.

Mailbox keeps the pointer to the data. Therefore, the size of the mailbox does not depend on its content.

d) Reliability Aspects

Real-time executives are responsible for safe and reliable process interactions. Interprocess dependencies may lead to deadlock, which can produce a fault in the embedded system. Signals of 'check' type can be used to avoid deadlock.

Real-time systems must be fault-tolerant. Fault tolerance can be achieved by error recognition ('watch-dog' mechanisms) and error recovery [CHT88]. Error recovery must be coordinated between all current active processes. The executive must provide it by interprocess communication within the critical time period. All critical tasks have to meet their deadlines irrespective of the fault that have happened in the system.

2.1.3 Scheduling

Tasks need hardware and software resources in order to execute. We can consider the scheduling problem as the allocation of the execution of tasks to the resources. Each resource has a local queue, whose items are members of a set of currently active processes. Scheduling is maintaining these queues, which are sorted according to a specified key [LA90].

A real-time task scheduler manages the access to the resources so that the tasks meet their timing requirements. The execution of tasks must be met or the system will be considered to have failed.

A scheduling policy is an approach which is suitable for scheduling a specific type of service (processing, I/O service, communication service).

The factors which should influence the scheduling policy for a specific application include [KR89]:

- reliability,
- timing constraints of tasks,
- criticality of tasks to the application,
- precedence and exclude relations between tasks,
- resource constraints.

A scheduling manager is a system task in which the management of the policy is dynamically updated. It can be also implemented on the application level. A run-time scheduler is an application of current policy, and is a part of a real-time kernel.

The calculation of schedules can be done off-line using complete knowledge about the task's characteristics and timing constraints (**static scheduling**) or can be planed at run-time (**dynamic scheduling**).

An optimal schedule of active tasks is one which ensures that task lateness (failure to meet deadlines) is minimized. This assumes that the set of tasks has been correctly organized in the first place. If all active tasks meet their deadlines, a set is feasible scheduled [XP90].

An overview of some preemptive scheduling policies is given in this section. Preemptive scheduling policies allow interrupting the execution of the running task, and swopping it with some another task from the queue of ready tasks. The implementation of a nonpreemptive optimal scheduling is NP-hard [CC89]. It means, that none of nonpreemptive optimal scheduling algorithms have polynomial time complexity (solution) [HU79].

a) Pre-emptive Round Robin Scheduling

The pre-emptive scheduling policy makes sure that no one task can use the resource for too long. Tasks which are ready to execute are ordered in a queue ('ready' list) of a resource. The pre-emptive round robin scheduler selects the first task from a queue and executes it till the end of a time slice period. Then it saves a task that has been executed on the end of a ready list. The execution is repeated until the tasks are finished. New 'ready-to-execute' tasks are added to the end of the ready list. The policy uses *FIFO (First In First Out)* strategy.

The policy is also called 'time slicing', because each process has the same time slice for its execution. When a time-slice period expires, an interrupt is created by the system clock. A task is returned to the tail of the ready list (queue), and the contents of the program counter and processor registers are saved (context switching).

A time slice is proportional to the period of the clock tick. The period of the clock tick have an effect on the response time of the system [Lei88]. If it is too short then context switching time is large.

The worst-case delay in servicing a task is nT, where n is the number of tasks in the ready list and T is the period of the tick. New tasks are connected to the end of the ready list, and have the worst-case execution time, irrespectively of their criticalness (priorities).

b) Pre-emptive Priority Scheduling

In this policy, the executive runs the 'ready' task with the highest priority at the end of each time slice [Lei88]. The highest-priority task is guaranteed to have a response time no longer than the tick. Low-priority tasks can remain unserviced for long periods of time.

The real-time kernel VRTX32 from Ready Systems [Rea86], [Cam88] supports preemptive priority scheduling. The real-time embedded system *VXcel* [Cam88] is under the control of a UNIX host processor and many VXchip processors that are connected via VME bus. The VRTX32 kernel presents the target environment of the VXcel system.

c) Rate Monotonic Priority Scheduling

The rate monotonic scheduling policy is an approach based on priority-driven preemption [LA90]. The priority scheme can be static or dynamic. A priority scheme is static if a priority of a task is assigned only once, while a dynamic scheme allows changing the priority with time. A task's priority reflects its required frequency.

The rate-monotonic scheduling algorithms, given on a set of periodic tasks, assign higher priorities to tasks with higher frequencies. It was proved that these algorithms can feasible schedule any set of size n with processor utilization u [Sha87]:

$$u = \sum_{i=1}^{n} c_i f_i \leq n(2^{\frac{1}{n}} - 1)$$

where c_i is the computation time of the task P_i, and f_i is its frequency. In other words, while the utilization u lies below this bound, the set of n periodic tasks will be feasible scheduled. u is the least upper bound, and is defined with the infimum of the utilization factors corresponding to the rate monotonic priority assignment over all possible request periods and run-times for the task [LL73].

The rate monotonic approach is typically able to schedule periodic task sets with utilizations of 85% to 90% [Sha87]. The worst-case bound is 0.693. It can be higher if some more information about the task set is known. The rate monotonic scheduling approach is optimum in the sense that no other fixed priority scheme can schedule a task set which cannot be scheduled by the rate monotonic priority assignment [LL73].

A non-critical task with a high frequency may postpone a critical task with a low frequency (*priority inversion*). Therefore it is not guaranteed that all the active tasks will meet their deadlines. An alternate solution is to adjust the priorities of the most frequent and the least frequent tasks (*period transformation*) [Lei88].

Some important results of rate monotonic scheduling theory is presented by Lui Sha and Goodenough [SG90].

Deadline Scheduling

The earliest deadline scheduling policy is a variable-priority approach which is given on a set of periodic and aperiodic tasks. Higher priority is assigned to the task whose deadline is closer to current time t. This scheme requires dynamic changes in priorities. The deadline of each task's execution is revised with time [Lei88].

The least slack scheduling policy is also called *earliest deadline as late as possible*. It selects the task with the the least slack to assign the lowest priority. The slack of task P_i at time t is:

$$slack(P_i, t) = max(d_i - c_i - t, 0)$$

where d_i is the deadline of task P_i, c_i its computation time, and t current (run) time.

Earliest deadline policies order all ready tasks by increasing deadline and execute them as soon as or as late as possible (the least slack scheduling). After the execution of the task its priority falls to some low priority, and then gradually rise again as time approaches to its deadline.

The algorithms that implement deadline scheduling assign priorities on the basis of the time still available for each task to meet its next deadline. Different mathematical functions may be available for assignment of the task priorities. Deadline scheduling can be used to achieve up to full 100% processors utilization [KR89].

The on-line dynamics acquired by deadline scheduling are reflected in the cost of sorting and rescheduling of tasks.

The real-time operating system Portos [HH89] supports deadline scheduling policy. When a task becomes 'ready', a check of feasible task set schedulability is performed. If its result is negative, faster alternatives are invoked and some tasks may be terminated. Early detection of the possibility of processing a task set in time is provided.

Other similar solutions of deadline scheduling were implemented in different real-time executives [MAC*87], [SZ89], [Wen88], [XP90], [McE88], [BS88a].

e) The Integration of Deadline and Criticalness

In hard real-time applications, no safety critical task may miss its deadline. Otherwise, unacceptable consequences may result. Deadline and priority scheduling order tasks to different priority levels. Priorities are functions of tasks' deadlines (deadline scheduling) or criticalness (priority scheduling). These two requirements sometimes conflict with each other. Non-critical tasks with short deadlines can exclude some safety critical task from the ready list (queue of active tasks). Therefore, a scheduling policy for hard real-time systems must integrate both task requirements, that is, its deadline and its criticalness.

Two scheduling algorithms for distributed hard real-time systems, which are implementations of such scheduling policy, were introduced and analyzed by Biyabani and Stankovic [BS88b].

The assumption is that tasks arrive dynamically and independently. Each task is defined with its worst-case computation time, level of importance, and deadline. These attributes are static, and are known at the time of arrival of the task. All tasks are aperiodic, and have no precedence constraints (they are independent). Both algorithms provide the guarantee that the higher criticalness tasks will meet their deadlines even if that implies the withdrawal of other (lower criticalness) tasks from the 'ready' list. The algorithms differ only in how they remove less critical ones from the 'ready' list in the case of transient overload. In the first algorithm, tasks are removed one at a time in strict order from low to high criticalness. In the second algorithm, the task with the largest deadline is selected to be first removed.

Because, the dynamic rescheduling of a task set can be very time consuming heuristic approaches are possible in the algorithms.

The performance analysis were done by simulating the behavior of the algorithms. The results showed that under the range of system conditions the algorithms outperformed all the baseline algorithms [BS88b].

f) Hard-Real Time Scheduling

Real-time applications are usually very complex. Therefore, a high degree of schedulability must be provided by the executive. The combination of different scheduling policies may improve the performance of the system. It was proved by Chetto and Chetto [CC89] that an algorithm for localization and duration of idle time intervals exists for periodic tasks scheduled on a monoprocessor system. An algorithm defines in advance when an idle time of the processor will begin and how long it will take. The remaining processor time can be used to service non-critical tasks. Such approach requires sophisticated scheduling mechanism, and schedulability analysis.

g) Scheduling in Distributed Multiprocessors' Scheduling

Distributed multiprocessor real-time embedded systems may achieve higher degree of schedulability than single-processor systems. However, even in such systems, and despite the best planning, there is always the possibility of a transient overload occuring. To handle such a case, load sharing (balancing) schemes which migrate tasks between the nodes of a distributed system have been considered. Load balancing is a central strategy. If a task can not be guaranteed to meet its deadline at a certain node, remote nodes are asked to bid for the right to run this task based on their surplus time. The information collecting process itself introduces communication delays [KR89], [Sha87]. Such schemes are generally not applicable for the real-time systems [Hal89]. They only hold for computing tasks. The real-time control applications are typically I/O oriented and the hard wiring of the peripherals to the nodes makes load sharing impossible.

h) Scheduling Analysis and Monitoring

In analyzing the scheduling algorithms, different performance metrics can be adopted. The metric used in dynamic real-time scheduling is to determine the percentage of tasks which meet their deadlines (Weighted Guarantee Ratio) [BS88b]. The performance analysis is done by simulating the behavior of the algorithms.

More complex analysis can be done by using the Scheduler 1-2-3, which has been developing for the Arts distributed real-time kernel [Tok90]. It uses analysis methods to determine whether a feasible schedule exists for a given task set and under what conditions deadlines can not be met. Scheduling 1-2-3 can be used as the close-form analyzer (used with the target) or simulator.

The real-time executive should provide the facility of observing the dynamics of task behavior by monitoring objects such as tasks, queues, times, etc [HH89]. This facility is necessary to verify the execution of application tasks, in the sense of correctness and timeliness. 'Arm' is the example of the advanced real-time monitor, which monitors and debugs real-time tasks [Tok90]. It can visualize the scheduling decisions and tasks' behavior. Besides it can analyzes the number of tasks, the number of met and missed deadlines, the number of scheduling events, and the CPU utilization. This tool has been developing to be used with the Arts real-time kernel for distributed real-time applications. The monitoring can be done directly on the target embedded system, or it can be run with Scheduler 1-2-3 (Arts schedule analyzer) simulator.

Conclusions

In this paper, we introduced the real-time executives for embedded systems. Some theoretical background needed to understand real-time embedded systems and concurrent programming were presented. We have reviewed some implementations of mutual exclusion approaches and real-time scheduling policies, that are most frequently used. Most of them are based on some assumptions, which are usually not very useful in 'hard' real-time embedded systems.

References

[Bro90] Andrej Brodnik. *Coroutines, Processes, and Parallelism.* Jožef Stefan Institute, Jamova 39, Ljubljana, Yugoslavia, 1990.

[BS88a] T.P. Baker and A. Shaw. The cyclic executive model and ada. In *Proceedings of the Real-Time Systems Symposium*, pages 120–129, December 1988.

[BS88b] S.R. Biyabani and J.A. Stankovic. The integration of deadline and criticalness in hard real-time scheduling. In *Proceedings of the Real-Time Systems Symposium*, pages 152–160, December 1988.

[Cam88] Peter Cameron. Real-time, multiprocessing and unix - the radstone mix. *Microsystem Design*, October 1988.

[CC89] H. Chetto and M. Chetto. Some results of the earliest deadline scheduling algorithm. *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, October 1989.

[CHT88] G.F. Carpenter, D.J. Holding, and A.M. Tyrrell. Analysis and protection of interprocess communications in real-time systems. *Journal of the Institution of Electronic and Radio Engineers*, Vol. 58, No. 4:173–180, June 1988.

[Coo90] Jim E. Cooling. *Software Design for Real-Time Systems.* Chapman and Hall, 1990.

[Hal89] Wolfgang A. Halang. New approaches for distributed industrial process control systems aimed to cope with strict time constraints. In *Proceedings of the EUROMICRO Workshop on Real-Time*, pages 130–136, June 1989.

[HH89] W.A. Halang and R. Henn. On-time:a high level real time language environment based on a portable operating system featuring, feasible and fault tolerant scheduling. In *Proceedings of the 2nd International Sofware Engineering for Real Time Systems*, September 1989.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison Wesley, 1979.

[KR89] D. Kalinsky and J. Ready. Real-time operating system kernel technology in the 21st century. September 1989.

[LA90] S.-T. Levi and Ashok K. Agrawala. *Real Time System Design.* McGraw-Hill, Inc., first edition, 1990.

[Lei88] A.W. Leigh. *Real Time Software for Small Systems.* Sigma Press, first edition, 1988.

[LL73] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, Vol. 20(1):46–61, 1973.

[MAC*87] A.K. Mok, P. Amerasinghe, M. Chen, S. Sutanthavibul, and K. Tanisirivat. Synthesis of a real-time message processing system with data-driven timing constraints. In *Proceedings of the Real-Time Systems Symposium*, December 1987.

[McE88] Michelle C. McElvany. Guaranteeing deadlines in maft. In *Proceedings of the Real-Time Systems Symposium*, pages 130–139, December 1988.

[Rea86] James F. Ready. Vrtx: a real-time operating system for embedded microprocessor applications. *IEEE Micro*, 8–17, August 1986.

[SG90] L. Sha and J.B. Goodenough. Real-time scheduling theory and ada. *COMPUTER*, April 1990.

[Sha87] Lui Sha. Ada tasking in real-time applications. November 1987.

[Sta88] John A. Stankovic. A serious problem for next-generation systems. *COMPUTER*, 1988.

[Sto87] Alexander D. Stoyenko. A schedulability analyzer for real-time euclid. In *Proceedings of the Real-Time Systems Symposium*, pages 218–227, December 1987.

[SZ89] R.F. Stone and F.S.M. Zedan. Designing time critical systems with tact. In *Proceedings of the EUROMICRO Workshop on Real-Time*, pages 74–82, May 1989.

[Tes84] Leland Teschler. Evaluating real-time software. *Machine Design*, June 1984.

[Tok90] Hideyuki Tokuda. Arts real-time scheduler analyzer/debugger. May 1990.

[Wen88] James W. Wendorf. Implementation and evaluation of a time-driven scheduling processor. In *Proceedings of Real-Time Systems Symposium*, pages 172–180, December 1988.

[XP90] J. Xu and D.L. Parnas. Scheduling processes with release times, deadlines, precedence and exclusion relations. *IEEE Transactions on Software Engineering*, Vol 16, No 3, March 1990.

NAČRTOVANJE UPORABNIŠKEGA VMESNIKA

Matjaž Debevc
Rajko Svečko
Dali Đonlagić
Tehniška fakulteta, Maribor

Keywords: man-computer interaction, user interface, user interface design.

POVZETEK

V prispevku so opisani pogoji in nekatere rešitve uspešne komunikacije med človekom in računalnikom. To področje računalniške tehnologije pridobiva vedno večjo veljavo, saj v mnogočem olajša uporabnikovo delo in poveča zadovoljstvo pri delu. Uspešno načrtovanje komunikacije zahteva poznavanje lastnosti uporabniškega vmesnika, uporabnikov, različnih tehnik dialoga ter navodil za izdelavo učinkovitega uporabniškega vmesnika. Na podlagi izkušenj pri raziskavi različnih tehnik dialoga je prikazan razvoj splošnega, univerzalnega uporabniškega vmesnika.

ABSTRACT

In the paper the conditions as well as some solutions of an effective computer-human interaction are given. In order to develop an effective design the properties of the users, user's interface, different technics of dialogs and instructions for making an effective user's interface should be known. Experiences in research of different technics of dialogs are the base to the instructions for the developement of general user's interface.

1. UVOD

Zaradi hitrega razvoja zmogljive računalniške opreme se vedno bolj pojavljajo zahteve po učinkovitem, hitrem, enostavnem in prijaznem uporabniškem vmesniku. Visoka tehnologija sedaj tudi vedno bolj omogoča tak sistem človek-stroj, ki je bliže človeku kot glavnemu udeležencu v tem sistemu. S tem prihajajo človeški faktorji vedno bolj do izraza pri načrtovanju uporabniškega vmesnika. Zaradi kompleksnosti in včasih zaradi zahtevnosti je programiranje učinkovitega uporabniškega vmesnika obsežen in dolgotrajen proces. Pogoje in navodila za izdelavo marsikje tudi premalo upoštevajo, vendar je vedno več področij, kjer se uveljavlja prizadevanje človeku čimbolj olajšati delo.

Dobri uporabniški vmesniki bi morali dati uporabniku čim večjo možnost samostojnega izražanja svojih ciljev in nalog. Uporabnik mora ves čas imeti občutek, da lahko svobodno dela in uporablja program. Imeti mora tudi občutek, da lahko vodenje programa prilagodi svojemu znanju in sposobnosti in ne obratno, da se mora prilagoditi programu.

Za načrtovanje tistih delov programa, ki skrbijo za pravilno komunikacijo med človekom in računalnikom, je potrebno veliko časa in vloženega truda. Ti deli običajno zavzamejo tudi 60-80% celotne programske kode. Vendar se vloženi napor in trud bogato obrestujeta s tem, da je uporabnik pri delu s tem programom zadovoljen, takšni programi so tudi finančno uspešnejši.

Če želimo uspešno načrtovati uporabniški vmesnik, moramo najprej spoznati pojem uporabniškega vmesnika in tipe uporabnikov, ki bodo uporabljali program. Nato na podlagi raziskave tehnik za dialog izberemo čim bolj univerzalen tip, kot je prikazan v primeru splošnega uporabniškega vmesnika. Najbolje je, da razvijemo takšen uporabniški vmesnik, ki se lahko prilagodi čimvečjemu krogu uporabnikov.

2. UPORABNIŠKI VMESNIK

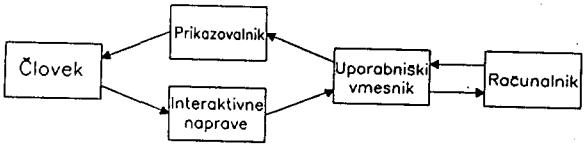
Uporabniški vmesnik je tip vmesnika človek-stroj. Seveda pri tem predstavlja drugačno problematiko, kakor običajni vmesnik za strojno opremo. Pri strojnem vmesniku mora načrtovalec misliti predvsem na razpoložljivo strojno opremo, pri uporabniškem pa na program.

V primeru, da gre za strojno opremo, je vmesnik enostavno definirati, saj gre za napravo, ki vsebuje samo strojne elemente in je izključno element strojne opreme.

V računalniških sistemih uporabniškega vmesnika ni tako enostavno definirati. Vključuje seveda strojno opremo, vendar je programska oprema pomemben del celega sistema. Brez programske opreme sistem ne more delovati. To, kar vidi uporabnik na zaslonu, je skupek funkcij strojne in programske opreme.

Učinkovit uporabniški vmesnik mora izpolnjevati naslednje pogoje:
- **enostavnost,**
- **jasnost,**
- **spoznavnost,**
- **popolnost,**
- **urejenost,**
- **zanesljivost.**

Slika 1 prikazuje lego uporabniškega vmesnika v komunikaciji med človekom in računalnikom.



Slika 1: Komunikacija človek - računalnik

3. TIPI UPORABNIKOV

Uporabniki so si med seboj zelo različni. Potrebno je poglobljeno razmišljanje o različnih potrebah in sposobnostih različnih skupin uporabnikov. Pri analizi uporabnikov je močno vpletena tudi psihološka teorija spoznavanja. V zadnjih letih je ta veja psihologije raziskovala informacijske procese znotraj človeka in njegovo zaznavanje. Poiskala je povezave med človeškim umom in računalniškim procesom. Pri tem najdemo presenetljive vzporednice. Najbolj pomembno je, da analiziramo ustrezen model uporabnika in ga vključimo v načrtovanje programa.

Eden od prvih problemov, do katerih pride načrtovalec, je analiza uporabnikov. Vprašanja, ki si jih pri tem zastavlja, so lahko naslednja:
- **česa je uporabnik sposoben?**
- **koliko učenja potrebuje?**
- **ali ima znanje programiranja in poznavanja algoritmov?**
- **ali uporablja terminal pogosto ali občasno?**

Po odgovoru na vprašanja je potrebna še opredelitev uporabnikov glede na zmožnosti in spretnosti:
- **strokovnjaki za računalništvo,**
- **strokovnjaki brez pravega znanja računalništva,**
- **vešči uporabniki,**
- **neizkušeni uporabniki.**

Sedaj, ko poznamo večino sposobnosti in naloge uporabnikov, se lahko odločamo, kakšno obliko in tehnike bomo uporabili za dialog med človekom in računalnikom. V glavnem trenutno poznamo dve različni tehniki dialoga. To so uporabniški vmesniki z ukazno vrstico in z ukaznim menujem. Na podlagi pridobljenih izkušenj, pri analizi teh dveh načinov, bomo prikazali primer izdelave uspešnega in prijaznega uporabniškega vmesnika.

4. UPORABNIŠKI VMESNIK Z UKAZNO VRSTICO

V tem primeru govorimo tudi o dialogu z iniciativo uporabnika. Uporabnik najprej vpiše besedo, kodo ali kakšen drug ukaz. Pri tem računalnik takoj izvede akcijo, ki ustreza vnosu. Program vsebuje svoj lasten programski jezik, ki se ga mora uporabnik temeljito naučiti. Vrstica, kamor uporabnik vpisuje ukaze, običajno ne vsebuje nikakršnih dodatnih informacij. Dialog z iniciativo uporabnika ima malo različnih oblik. Glavna razlika je v tipu informacije, ki jo vpiše in se interpretira v računalniku. Ti tipi so:
- **besede** (predstavljajo imena funkcij, ki se naj izvedejo),
- **črke, okrajšave,**
- **logični in matematični izrazi,**
- **akcijske kode** (kratke kombinacije različnih znakov),
- **ukazni jeziki** (formalni jezik za vodenje, podobno kot pri programiranju).

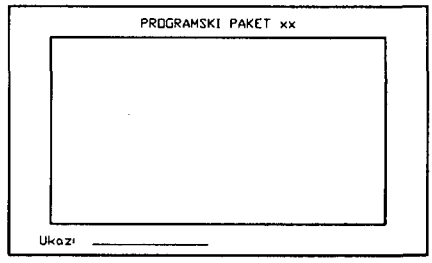
Značilnost teh programov je, da po začetnem, uvodnem spoznavanju in učenju dela s programom, lahko nudijo zelo hitro delo in s tem mnogo hitrejše reševanje problema, še posebej tedaj, kadar gre za rutinsko delo.

Prednosti te vrste programov torej so:
- visoka prilagodljivost glede na hitrost dela,
- splošna uporabnost vseh funkcij v vsakem trenutku,
- pri rutinskem delu skrajša čas reševanja problema.

Vendar je problem pri uporabi teh programov v rednem in vztrajnem delu, saj je v nasprotnem primeru potrebno posebno osveževanje spomina s pomočjo priročnika ali tečaja.

Slabosti so torej:
- dolga uvodna, spoznavna faza,
- velik začetni napor,
- samo redno delo je učinkovito.

Slika 2 kaže primer uporabniškega vmesnika z ukazno vrstico.



Slika 2: Uporabniški vmesnik z ukazno vrstico

5. UPORABNIŠKI VMESNIK Z UKAZNIMI MENUJI

Pri teh vmesnikih zasledimo dialog z iniciativo računalnika. To so programi, ki pokažejo menu in čakajo na odgovor. Imajo lasten nabor ukazov, enako kot vmesniki z ukazno vrstico. Računalnik prikaže te ukaze v obliki menuja in podmenujev. Po izbiri odgovarjajoče točke v menuju se izvede njemu ustrezen podprogram ali modul. Pri tej vrsti dialoga so najbolj pogoste naslednje tehnike izbire:
- **vtipkanje imena**
  Želeni ukaz vtipkamo s pomočjo tipkovnice. Menu uporabimo samo kot pomnilno in obveščevalno sredstvo.
- **vtipkanje labele**
  Želeni ukaz vtipkamo s pomočjo labele, ki je lahko numerična, črkovna ali kombinacija obeh.
- **simulacija odbire z značko**
  Značka ("cursor") se postavi preko želenega elementa. Nato pritisnemo na določeno tipko (običajno <Return>). Simulacijo lahko opravimo s tipkami za premik značke ali z miško. Za miško je značilen premik značke po zaslonu glede na premik miške z roko. Prva skrajno leva tipka je namenjena potrditvi, kakor tipka <Return>. Skrajno desna ima lahko enak pomen kot <ESC>. Ob pritisku obeh tipk hkrati se običajno simulira tipka <F1> (Help).
- **izbira ikon - grafičnih simbolov**
  Z uporabo grafičnih simbolov je izbira ukaza na manjšem področju zaslona enostavnejša in hitrejša, kakor izbira med množico besed.
- **funkcijska tipka**
  Ko pritisnemo na funkcijsko tipko (običajno F1-F20), se izvede ukaz prirejen tej tipki.
- **razpoznavanje govora**
  Uporabnik izgovarja ime ukaza. Ta metoda je posebej primerna za tiste, ki niso vajeni uporabljati tipkovnice in za invalide. Je pa tudi najbolj naravna možnost izbiranja ukazov. Današnja tehnologija že omogoča zaznavo posameznih ukazov s tem, da je potreben predčasen trening, oziroma navajanje računalnika na posamezne besede, ki jih izgovarja uporabnik. Problem, ki še ni dokončno rešen, je v zaznavi tekočega govora.

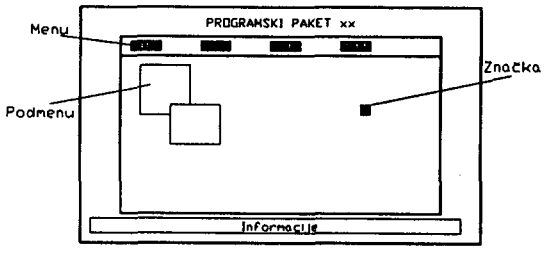
Pri uporabniških vmesnikih z ukaznimi menuji se na zaslonu najprej prikaže uporabniško polje z menuji in informacijami. Sledeč uporabnikovi zahtevi, prikaže program podmenuje in dodatne informacije ter ob potrditvi izvede želen ukaz.

Prednosti, ki jih prinašajo takšni programi so:
- uporabniško polje z menuji in informacijami,
- kratka, uvodna spoznavna faza,
- majhen začetni napor,
- možno je tudi samo občasno delo.

Po drugi strani nudijo vmesniki z ukaznimi menuji vedno samo določen potek izvajanja ukazov, ki je odvisen od zgradbe programa.

Slabosti so zato naslednje:
- slabša prilagodljivost glede na hitrost tipkanja,
- težje je krajšanje postopka,
- težje je razširiti množico funkcij.

Slika 3 kaže primer slike na zaslonu uporabniškega vmesnika z ukaznimi menuji.



Slika 3: Uporabniški vmesnik z ukaznimi menuji

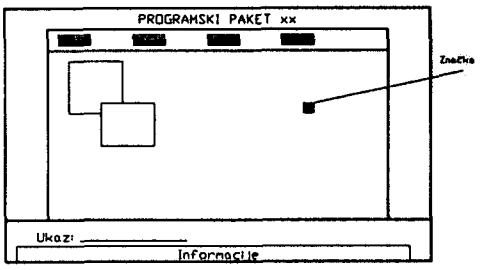
6. SPLOŠEN UPORABNIŠKI VMESNIK

Ena izmed zelo pomembnih zahtev vsakega programskega paketa je uporabniška prijaznost. Uporabnik, ki želi obvladati kak kompleksen programski paket, nima časa, da bi se nekaj dni ali celo tednov moral učiti dela s programom. Zato je potrebno pravilno načrtovanje in razvijanje uporabniškega polja na zaslonu. To polje mora biti tako zasnovano, da ga lahko s pridom uporabljajo tako začetniki kot izkušeni uporabniki. Začetnik mora imeti na voljo dovolj začetnih informacij, ki mu pomagajo reševati problem, medtem ko mora imeti izkušen uporabnik možnost še bolj pospešiti svoje delo.

Vse omenjene izkušnje in zahteve se upoštevajo pri načrtovanju in razvijanju splošnega uporabniškega vmesnika. Pri tem se poskuša olajšati delo in reševanje problema ne glede na tip uporabnika. To se doseže z združevanjem lastnosti uporabniških vmesnikov z ukazno vrstico in ukaznim menujem. Ukazi obeh so v tem primeru identični. Pri tem se združijo dobre lastnosti in poskušajo zmanjšati slabe. Slika 4 kaže sliko na zaslonu s takšnim uporabniškim vmesnikom.

S tem se omogoči začetnikom lažje in preglednejše delo s pomočjo menujev, medtem ko že izkušeni uporabniki posežejo po ukaznem načinu. Možna je seveda tudi kombinacija obeh. Pri razvijanju se držimo navodil za izdelavo uspešnega vmesnika.

Ta navodila so:
- **razumljivost ukazov**
  Dobro delo z menuji zahteva enostavne in razumljive menujske točke. Na primer, namesto besede "MENTOČ" uporabimo stavek "Menjava točke".
- **jezik**
  Ukazi naj bodo v jeziku naroda, kamor program prodajamo.

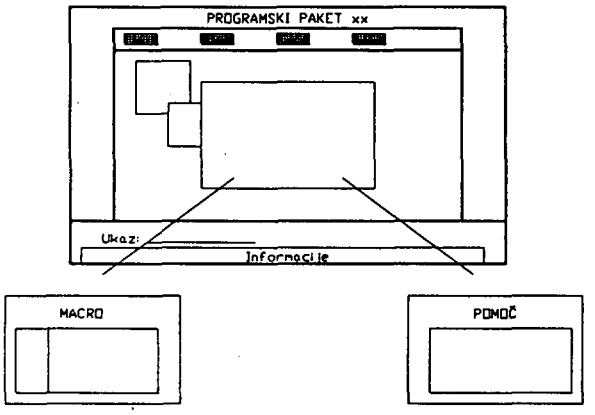


Slika 4: Hibridni uporabniški vmesnik

- **preglednost besed**
  Besede pišemo tako, da začetnice pišemo z velikimi črkami, ostalo pa z malimi. S tem povečamo preglednost in berljivost ukazov in s tem posredno tudi hitrost branja. Primer, namesto "NALAGANJE", pišemo "Nalaganje".
- **informativnost**
  Poleg razumljivih, nedvoumnih besed omogočimo dodaten izpis kratkih informacij o trenutnem ukazu v spodnjem delu zaslona. Ta izpis naj bo omogočen, kadar se premikamo s pomočjo tipkovnice. Pri uporabi miške pa izpisi niso več potrebni.
- **naslavljanje menujev**
  Čim večje je število menujev, bolj pomembno je to navodilo. Naslovi naj bodo kratki in jedrnati.
- **omejevanje števila ukazov v menuju**
  Primerno število je nekje v intervalu 7 ± 2 (Millerjevo magično število).
- **razvrščanje ukazov**
  V odvisnosti od njihove pogostosti uporabe jih razvrstimo na različne načine. Običajno postavimo najbolj pogosto uporabljane ukaze na vrh menuja. Pri daljših menujih ali izpisih, če jih ne moremo razbiti, je bolje, da omogočimo razvrstitev, kakršno si uporabnik želi (po abecedi, po številčnem vrstnem redu, po kronologiji...). Ukaze lahko tudi razvrstimo po pomenu, tako da skupaj združimo podobne kategorije ukazov.
- **izbira s črkami**
  Omogočiti moramo izbiro samo z ustrezno črko ukaza, kadar izbiramo po menuju. To omogočimo brez potrditve s tipko <Return>. Pri ukazu, na primer, "Vnos", naj uporabnik vtipka samo črko "V" za izvedbo tega ukaza. Če se več ukazov začne z isto črko, uporabnik vtipka veliko črko znotraj besede. Pri ukazu "naLaganje" se vtipka črka "L". Omogočena naj bo tudi drugačna barva ozadja, kjer je trenutni ukaz.
- **zapisovanje ukazov**
  Ukazi so lahko zapisani vertikalno ali horizontalno. Bolj primerna je vertikalna organizacija ukazov, saj potrebuje človeško oko manj časa za iskanje ukazov po vertikali kakor po horizontali.
- **vpisovanje ukazov**
  Pri vpisu daljših ukazov v ukazno vrstico naj bo omogočen tudi krajši vpis. Namesto ukaza "Računanje" se lahko vtipka beseda "Rač".
- **število oken**
  Število oken, ki so hkrati na zaslonu, je odvisno od specifičnosti posamezne naloge. Največje, še primerno število pri vodenju programa je med 3 in 5 okni. V takšnem primeru je uporabnik še sposoben dojeti informacijo in ne prihaja do nasičenosti zaslona.
- **globina hierarhičnosti**
  Naj bo v 3 do največ 4 nivojih. Pri več nivojih postane delo utrudljivo in nepregledno. Uporabnik mora uporabiti preveč funkcij, da lahko opravi svoje delo.

To so glavna navodila, po katerih se ravnamo pri razvoju poljubnega programskega paketa. Kljub temu, da se omogoči hibridno vodenje programa, se odločimo še za dva koraka. Omogočiti je potrebno še dodatno krajšanje postopka in posameznih ukazov s pomočjo pisanja ukazov makro in nato še tekočo pomoč ("on-line-help"), ki se je

v zadnjem času zelo močno razširila (slika 5).



Slika 5: Ukazi makro in tekoča pomoč

Ukazi makro se v zadnjem času tudi zelo pogosto pojavljajo v večjih, dražjih programskih paketih. Je zelo koristna tehnika, ki omogoča, da se omeji samo na določene naloge. Ta tehnika omogoča, da se lahko z eno samo tipko ali ukazom opravi postopek, ki zahteva mnogo tipkanja. Makroji se morajo enostavno definirati, popravljati, in za nadaljnjo rabo shraniti po potrebi v datoteko. Pri tem se uporabi poseben makro urejevalnik, kamor se vpisujejo in pregledujejo imena makro ukazov. Vpisujejo se lahko tudi samo črke ali okrajšave.

Ukazi makro:
- so enostavni za definiranje,
- vodijo k skrajšanju dela s programom,
- obsegajo vse funkcije programskega paketa.

S pomočjo ukazov makro se lahko tudi zmanjša število napak pri delu, saj se lahko daljši postopek zamenja z enim samim ukazom. S pomočjo menujev, ukazov in ukazov makro smo dosegli zelo visoko stopnjo prilagodljivosti programskega paketa, ki vodi k uspešnemu pripomočku v merilni tehniki. Višji nivo makro ukazov so okrajšave več zaporednih ukazov, ki se pri delu lahko večkrat ponavljajo. Na ta način zelo zmanjšamo čas reševanja problemov in povečamo učinkovitost programskega paketa. Z enim samim ukazom lahko izvajamo, na primer, več zaporednih meritev, ki vsebujejo vedno enako množico ukazov, le rezultati so lahko vsakokrat drugačni.

Vendar tudi pri vseh teh olajšavah še lahko nastanejo problemi, vprašanja in nejasnosti pri delu s programom. Iz tega razloga se omogoči še posebno tekočo pomoč na zaslonu. To je majhen priročnik z izpisi na zaslon, ki nudi tudi možnost za šolanje uporabnika. To je zelo pomemben pripomoček med delom s programom. Ko smo v določen delu programa in imamo težave, si lahko pomagamo s tipko F1, ki nam prikaže informacije o funkcijah, pri katerih smo se ustavili. Poleg dodatnih informacijah o trenutni funkciji se lahko pomikamo v tem načinu po različnih notranjih menujih za popolnejšo informacijo. Poleg tega naj nudi tekoča pomoč tudi različne primere, ki so potrebni za rešitev določenega problema. Tekoča pomoč naj bo enostavna za uporabo in pregledna.

7. SKLEP

V svetu se vedno pogosteje postavlja vprašanje po visoki sposobnosti komunikacije človeka z računalnikom. Vedno hitrejši razvoj računalnikov minimizira dimenzije strojne opreme in maksimizira hitrost obdelave. Na ta način se lahko sedaj uresničujejo želje, ki še pred nekaj leti niso bile uresničljive. Danes je zelo velik napredek v razvoju programskih paketov z učinkovitim uporabniškim vmesnikom. Predvsem od komunikacije med človekom in računalnikom je odvisno, kako prijetno in učinkovito je delo s programom. Ta del tudi običajno zahteva večji del programske kode, kakor samo jedro programa. Vedno bolj pogosti so uporabniški vmesniki, ki vsebujejo menuje, nasvete, pri katerih uporabnik sporoča svoje odločitve z izbiro ene od možnosti ali pa s kazanjem (miška). Enostaven in prilagodljiv uporabniški vmesnik je zato najboljše orodje za daljšo uporabo programskega paketa.

V članku smo odgovorili na vprašanje, po katerih pravilih se moramo ravnati pri pisanju uporabniških vmesnikov. Vse te metode in rešitve, ki smo jih našteli v primeru splošnega uporabniškega vmesnika, so značilnosti kompleksnega in zaokroženega uporabniškega vmesnika. Na ta način je uporabniški vmesnik enostaven in zelo prilagodljiv glede na zahteve uporabnikov. V primeru splošnega uporabniškega vmesnika lahko program uporabljajo hkrati začetniki, izkušeni uporabniki in strokovnjaki. Pri tem tudi nudi visoko stopnjo gotovosti in uspešno, kreativno delo pri meritvah brez motenj v delovanju programa.

Če vse to upoštevamo pri razvoju programskega paketa, bo mnogo lažje reševati probleme, v našem primeru problem učinkovitega uporabniškega vmesnika.

LITERATURA

Simpson, H., Design of User-friendly programs for small computers, McGraw-Hill, New York, 1985.

Simpson, H., Programming the IBM PC User Interface, McGraw-Hill, New York,1986.

Maier, H., A. Piotrowski, Messen, Steuern, Regeln mit IBM-kompatiblen PCs, Interest Verlag, Kissing, 1990.

Johannsen, G., Kassel, Neue Entwicklungen bei Mensch-Maschine-Systemen, Automatisierungstechnik, št. 10, 1987, str. 385-395.

Bullinger, Software Ergonomie, Expert Verlag, Sindelfinger, 1986.

Weiskamp, K., L. Heiny, N. Shammas, Power Graphics using Turbo Pascal, John Wiley & Sons, New York, 1989.

SODOBNI PROGRAMSKI JEZIKI V SISTEMIH REALNEGA ČASA

INFORMATICA 4/90

Keywords: real time computer systems, programming languages, high level languages, reliability, safety.

Giovanni Godena
Vladimir Jovan
Inštitut J. Stefan, Ljubljana

Članek obravnava značilnosti programskih jezikov namenjenih uporabi v sistemih realnega časa. Zaradi značilnih zahtev sistemov realnega časa ter vedno zahtevnejših aplikacij, se pri tovrstnih programskih jezikih posveča vedno večja pozornost načinu obravnavanja podatkovnih tipov, modularnosti jezika, možnosti nizkonivojskega programiranja, varnem obravnavanju napak in mehanizmom multiprogramiranja, kar vse vpliva na zahtevano visoko stopnjo pravilnosti, zanesljivosti in varnosti delovanja sistemov realnega časa.

MODERN REAL-TIME PROGRAMMING LANGUAGES. This paper discusses some important features of real-time programming languages. Due to specific requirements of real-time systems, as well as expanding applications complexity, more and more attention is devoted to data types, modularity, low-level programming, exception handling and multiprogramming facilities that help achieving the required high degree of correctness, reliability and safety of real-time systems.

1. UVOD

Eno največjih rasti uporabe digitalnih računalnikov smo v zadnjih letih zaznali na področju sistemov realnega časa. Aplikacije postajajo vedno večje in kompleksnejše, kot posledica tega pa se pojavlja vedno večja potreba po programski opremi, ki deluje pravilno, zanesljivo in varno, ter je dokončana v okviru časovnih in finančnih omejitev. Pri doseganju tega cilja so nedvomno ključnega pomena pravilne specifikacije, ustrezna metoda načrtovanja ter dobra razvojna orodja. Navedene pojme navadno obravnavamo ločeno od programskih jezikov, vendar se moramo zavedati, da je izbira ustreznega programskega jezika bistvenega pomena za izdelavo kvalitetne programske opreme.

2. ZNAČILNOSTI SISTEMOV REALNEGA ČASA

Pojem "delovanje v realnem času" označuje aktivnost sistema, ki se mora na zunanje dogodke odzvati v vnaprej določenem končnem času. Po tej definiciji bi lahko tudi vsak poslovni računalniški sistem obravnavali kot sistem v realnem času, saj se pričakuje odziv v nekaj sekundah. Po drugi strani pa ne moremo reči, da tak sistem deluje nepravilno, če se namesto po dveh, odzove po štirih sekundah, kar pa gotovo ne velja za prave sisteme realnega časa, pri katerih je pravilen rezultat, ki zamuja, lahko enako slab kot napačen rezultat v pravem času. Taki sistemi so navadno nameščeni v okolje, kjer lahko časovna zakasnitev povzroči zelo hude posledice (materialne ali tudi človeške).
Sistemi realnega časa morajo sočasno izvajati različna asinhrona in sinhrona opravila, od vzorčenja, ugotavljanja trenutnega stanja do izvajanja ukrepov vodenja. Pri izpolnjevanju vseh teh zahtev seveda ni vseeno, kakšen programski jezik uporabimo. Če naj jezik podpira učinkovito načrtovanje, razvoj in vzdrževanje programske opreme za delo v realnem času, mora poleg vseh lastnosti splošnih programskih jezikov, posebej zadoščati nekaterim kriterijem značilnim za sisteme realnega časa.

3. KRITERIJI OCENJEVANJA PROGRAMSKIH JEZIKOV V REALNEM ČASU

Kot glavni kriteriji pri ocenjevanju programskih jezikov v realnem času se v zadnjih letih vedno pogosteje navajajo (v zaporedju po njihovi pomembnosti) zanesljivost, čitljivost, fleksibilnost, enostavnost, prenosljivost in učinkovitost /1/.

- *zanesljivost* pomeni možnost odkrivanja napak pri prevajanju ali pri izvajanju. Ker je sisteme, delujoče v realnem času pogosto težko testirati, je ta značilnost jezika verjetno najvažnejša. K zanesljivosti jezika največ prispeva način obravnavanja podatkovnih struktur v smislu omejevanja obsega vrednosti in dovoljenih operacij (strogo obravnavanje tipov, omejevanje vidljivosti podatkov), poleg tega pa tudi strukturiranost in modularnost jezika.

- *čitljivost* je odvisna od izbire ključnih besed jezika, strukturiranosti in modularnosti jezika. Posledice dobre čitljivosti programa so zniževanje stroškov dokumentiranja in vzdrževanja, lažje odkrivanje napak in s tem večja zanesljivost.

- *fleksibilnost* pomeni možnost izvajanja vseh potrebnih operacij, brez uporabe zbirnega jezika. V sistemih realnega časa je ta značilnost posebej važna, ker v teh sistemih srečujemo veliko nestandardnih perifernih enot. Seveda je fleksibilnost v nasprotju s zanesljivostjo.

- *enostavnost* jezika omogoča lažje učenje in uporabo, kar dosežemo ne toliko z izogibanjem kompleksnim konstruktom, kot z izogibanjem raznim nelogičnim omejitvam v jeziku.

- *prenosljivost* pomeni neodvisnost programske od materialne opreme, kar omogoča prenos programov z enega na drug računalnik. V sistemih realnega časa je prenosljivost težko doseči, je pa tudi manj pomembna kot pri nekaterih drugih računalniških aplikacijah

- *učinkovitost* izraža značilnost jezika, da je dobljena namenska koda kompaktna in hitra za izvajanje. Jezik mora zato vsebovati take konstrukte, ki omogočajo učinkovito implementacijo. Zaradi zagotavljanja odzivnih časov sistema, se je potrebno izogibati mehanizmom, katerih čas izvajanja je nepredvidljiv. V preteklosti je učinkovitost jezika obravnavana kot najvažnejša, kar je šlo na račun zanesljivosti, fleksibilnosti in enostavnosti jezika. Danes, ko je materialna oprema vedno zmogljivejša, stroški razvoja in vzdrževanja programske opreme pa vedno višji, pomembnost učinkovitosti programske opreme pada, vse bolj pa se skuša doseči časovno deterministično obnašanje sistema. To pomeni ne čim hitrejše, ampak pravočasno izvajanje funkcij.

4. ZAHTEVE JEZIKOV ZA UPORABO V SISTEMIH REALNEGA ČASA

Za zanesljivost jezika je gotovo najpomembnejši način obravnavanja podatkovnih tipov. Podatkovni tip določa nabor vrednosti in dovoljene operacije nad podatkom.
Tako kot moramo določiti podatke, moramo določiti tudi akcije, ki jih program izvaja nad temi podatki. Znane so prednosti strukturiranega programiranja; zato mora jezik vsebovati konstrukte, ki omogočajo, ali celo silijo k strukturiranem programiranju.
Sistemi v realnem času so pogosto veliki in kompleksni, pri njihovem razvoju sodelujejo večje ekipe, pogosto so potrebne spremembe. Vse to pogojuje zahtevo po možnosti modularnega programiranja, kar dosežemo s pomočjo raznih bločnih struktur in mehanizmov abstrakcije, ki omogočajo ločeno prevajanje in omejeno vidljivost podatkov.
Ker so sistemi v realnem času pogosto izvedeni kot množica sočasnih dejavnosti, obstaja potreba po vključevanju mehanizmov multiprogramiranja, ki so sicer elementi operacijskega sistema, v programski jezik.
Zaradi raznih nestandardnih vhodno-izhodnih enot s katerimi mora sistem v realnem času komunicirati, kot tudi zaradi potrebe po sistemskem programiranju, mora jezik omogočati nizkonivojsko programiranje. Pri nizkonivojskem programiranju prihaja do opuščanja pravil konsistentnosti tipov, kar zmanjšuje zanesljivost. Zato mora biti nizkonivojsko programiranje strogo ločeno od ostalega dela jezika.
Če naj bo sistem v realnem času zanesljiv in varen, mora programski jezik vsebovati tudi konstrukte, ki omogočajo varno obravnavanje napak.
V zadnjem času se pri najzahtevnejših t.i. *hard real-time* sistemih vedno bolj kažejo pomanjkljivosti sedanjega nedeterminističnega obravanja časa, tako v obstoječih operacijskih sistemih, kot tudi v programskih jezikih. Časovno obnašanje se verificira z "ad-hoc" tehnikami (testiranje), ali z dragimi simulacijami, ki pa tudi niso popolne. Že manjša sprememba v sistemu zahteva ponovitev celega postopka. Zato so v zadnjem času prisotni poskusi eksplicitnega vpeljevanja časovne dimenzije v operacijske sisteme in programske jezike ter razvoja eksaktnih metod vnaprejšnjega verificiranja tj. dokazovanja pravilnega časovnega obnašanja sistema v vseh

okoliščinah. Konstrukti teh novih (sinhronih) programskih jezikov naj bi omogočali izražanje časovnih zahtev /8/.

5. PODATKOVNE STRUKTURE

Visokonivojski programski jezik podaja abstraktni model, ki na problemsko orientiran način definira podatke in nad njimi dovoljene operacije. To pomeni, da vsak podatek pripada določenemu tipu, ki določa nabor vrednosti, ki jih podatek lahko zavzame, kot tudi nabor operacij, ki so za ta tip dovoljene.
Vsak programski jezik pozna nekaj osnovnih tipov, kot so npr. **integer** (cela števila), **real** (realna števila), **boolean** (logične spremenljivke), **char** (črkovne spremenljivke). Poleg tega naj bi jezik poznal izpeljane tipe, ki omogočajo ločevanje spremenljivk, ki so istega osnovnega tipa, logično pa so različne. Za doseganje višjega nivoja abstrakcije so bistveni sestavljeni tipi, kot so zapisi (**record**), polja (**array**), naštevalni in delni tipi.
Tipi podatkovnih struktur so lahko podani šibko (npr. *C*) ali močno (npr. *Modula-2*, *Ada*) /1/. Jezik ima močno podane tipe, če velja:

- vsak objekt pripada določenemu tipu
- vsak tip določa nabor vrednosti in operacij
- pri vsaki prireditveni operaciji mora tip prirejene vrednosti biti enak tipu objekta, ki se mu prireja vrednost
- vsak operator nad podatkom mora pripadati naboru operatorjev prirejenih tipu tega podatka.

Razlikujemo strukturno in imensko enakost tipov. Če jezik zahteva strukturno enakost (*Modula-2*), lahko medsebojno prirejamo spremenljivke različnih izpeljanih tipov, ki pripadajo istemu osnovnemu tipu, če pa je zahtevana imenska enakost (*Ada*), lahko prirejamo samo spremenljivke istega tipa. V tem primeru je jezik varnejši pred napakami. To pa gre na račun fleksibilnosti, in zato mora jezik omogočati redefiniranje operatorjev. Primer:

```
TYPE VSOTA = INTEGER; INDEX = INTEGER;
 . . .
VAR  x:VSOTA; i:INDEX;
 . . .
x:=x+i;
```

Če jezik zahteva imensko enakost tipov, bo prevajalnik tukaj javil napako.

6. MEHANIZMI ABSTRAKCIJE

Programsko opremo vedno skušamo načrtovati s pomočjo postopka zaporednega razčlenjevanja. To velja za načrtovanje podatkovnih, kot tudi programskih struktur. Pri podatkovnih strukturah nam to omogočajo sestavljeni tipi, pri programskih strukturah pa večnivojsko gnezdenje procedur.
Vzporednost postopnega razčlenjevanja podatkovnih in programskih struktur pomeni poskus specificiranja programa od najvišjega nivoja abstrakcije navzdol. Take podatke in operacije združimo v programske enote, ki okolju dovoljujejo dostop do podatkov samo preko teh operacij. Tako programsko enoto imenujemo *modul* /1/,/2/.
Modul sestoji iz dveh delov: specifikacijskega in privatnega ali implementacijskega. Objekti navedeni v specifikacijskem delu so dostopni zunanjem okolju, vse podrobnosti implementacije pa so skrite v privatnem delu. Na ta način je preprečen vsak nedovoljen dostop do podatkov. Primer:

```
MODULE stack;
  (* definicijski del modula *)
  DEFINE PROCEDURE push(in x:item);
         PROCEDURE pop(out x:item);
  (* implementacijski del modula *)
  PRIVATE
    TYPE STKIDX = INTEGER RANGE 0..100;
    VAR s:ARRAY[1..100] OF ITEM;
        sp:STKIDX;
    ...  (* implementacija procedur
            push in pop *)
END MODULE stack;
```

Zunanji uporabnik ne more npr. vpisati podatka direktno v polje s ali pa spremeniti vrednost kazalca sp; edini dovoljeni operaciji sta push in pop.

Dobre lastnosti modulov so naslednje:

1. Modul zbira logično povezane objekte v enoto v skladu z abstraktnim konceptom, ki izvira iz procesa načrtovanja.
2. Vmesne specifikacije (definicijski del) točno določajo funkcijo, ki jo modul opravlja za okolje. To omogoča, da privatni (implementacijski) del ne zamegljuje delovanje modula z za okolje nebistvenimi detajli.
3. Modul omogoča lažje testiranje in odpravljanje napak. Ko je modul enkrat stestiran, omejeni dostop do njega onemogoča okolju da povzroči napako znotraj modula. To pomeni, da sistem gradimo od spodaj navzgor z dodajanjem posameznih modulov. Ko ugotovimo napako, bo ta možna samo v zadnjem dodanem modulu.

4. Modul podpira timsko programiranje in graditev zelo velikih sistemov. Ko imamo enkrat izdelane vmesne specifikacije, se posamezne module lahko razvija in testira neodvisno od ostalih.
5. Modul olajšuje vzdrževanje programa, saj bistveno povečuje njegovo čitljivost.
6. Modul ne povečuje kode in ne časa izvajanja, ker je to samo informacija prevajalniku.

Tip modul je bil prvič vpeljan pri jeziku Modula, pozna pa ga tudi *Ada*, kjer se imenuje **package**.

7. NIZKONIVOJSKO PROGRAMIRANJE

Računalniški sistem za delovanje v realnem času se od drugih računalniških sistemov najbolj razlikuje po načinu povezave z zunanjim okoljem. Tak sistem mora sinhronizirati svoje delovanje s zunanjimi dogodki ter komunicirati z različnimi nestandardnimi enotami. Zato pri takih sistemih ne moremo imeti standardnih knjižnic, ki bi nam omogočale komunikacijo na visokem nivoju z zunanjimi enotami, kot je to primer pri večini ostalih računalniških sistemov in splošnih operacijskih sistemih. Zaradi tega pri sistemih za delo v realnem času navadno ne najdemo več običajne strukture materialna oprema + operacijski sistem + aplikacijski program, ampak pogosto samo materialna oprema + aplikacijska programska oprema z komponentami operacijskega sistema.
Iz tega sledi zahteva, da mora programski jezik vsebovati konstrukte, ki bodo omogočali programiranje na nizkem nivoju.
Večina programskih jezikov slabo podpira nizkonivojsko programiranje. V glavnem omogočajo vstavljanje kode v zbirnem jeziku, kar je premalo.
Izjeme so *C*, *Ada* in *Modula-2*, ki je šla najdlje v to smer, saj je tudi zasnovana kot jezik za sistemsko programiranje. *Modula-2* pozna tipe **address**, **word** in **bitset**, absolutne spremenljivke, omogoča naslavljanje vhodno-izhodnih registrov, preslikavo uporabniško definiranih tipov na registre (tip **bitset**), ter programiranje prekinitvenih podprogramov /5/.

8. OBRAVNAVANJE IZJEM

Računalniški sistem za vodenje v realnem času ne sme obstati, če med izvajanjem pride do kakšne izjemne situacije, ampak mora nadaljevati izvajanje.
Napako lahko odkrije:

1. Implementacijski nivo, to je materialna oprema ali koda, ki jo vrine prevajalnik (npr. deljenje z 0 ali prekoračitev polja). Te napake so hujše, saj lahko nastopijo kjerkoli, tudi sredi izraza.
2. Izrecno programirani testi stanja sistema (IF not pogoj THEN napaka).

Pri računalniških sistemih, ki niso namenjeni vodenju v realnem času, je ustrezen odgovor na napako tudi sporočilo o napaki in prekinitev programa, sistem za vodenje v realnem času pa mora biti zmožen odpraviti posledice napake in nadaljevati normalno obratovanje.
Zahteve, ki jim mora zadoščati mehanizem za obravnavanje napak, so naslednje:

- mehanizem mora biti enostaven
- količina dodatne kode mora biti čim manjša
- dodatna koda se sme izvajati samo v primeru napake
- vse napake mora obravnavati enako, ne glede na izvor in mehanizem odkrivanja napake.

Obstajajo različni pristopi. Pri večini jezikov se za ta namen uporablja **goto** stavek, kar pa ni dobra rešitev.
Najbolje je ta problem rešen v *Adi* z uvajanjem novega tipa **exception**, z operatorjem **raise**. Treba pa je povdariti, da je ta mehanizan zapleten in je eden od vzrokov kompleksnosti prevajalnikov jezika *Ada*.

9. PARALELNO PROGRAMIRANJE (MULTITASKING)

Multitasking pomeni sočasno (multiprocessing - več procesorjev), ali navidezno sočasno (multiprogramming - en procesor) izvajanje več programov. Pri večjih računalnikih je to že dolgo uporabljan način, ki je imel za posledico veliko boljšo izkoriščenost dragih računalniških zmogljivosti.
Pri sodobnih real-time aplikacijah in ob vedno cenejši in zmogljivejši materialni računalniški opremi pridejo v poštev drugi razlogi za multitasking. Glavni razlog je v lažjem reševanju določenih specifičnih problemov pri real-time aplikacijah:

- pri real-time aplikaciji računalnik streže posameznim asinhronim dogodkom, ki zahtevajo takojšen odziv. To se lahko doseže tudi v okolju, ki ni multitasking s pomočjo prekinitvenega mehanizma, je pa ta način manj fleksibilen
- real-time sistem mora istočasno opravljati več medsebojno različnih funkcij. Bolj naravno je, da N manjših programov izvaja po eno funkcijo, kot pa da en program (kompleksen) izvaja N funkcij
- delitev na več procesov izhaja tudi iz sodobnih metod načrtovanja, kjer sistem razgrajujemo na procese, kar ima za posledico lažji razvoj, dokumentiranje, dopolnjevanje, vzdrževanje ter timsko delo.

Ni treba posebej povdarjati, da je smiselno v programski jezik vpeljati konstrukte multitasking programiranja, ki omogočajo opisovanje množice med sabo sodelujočih procesov. Ti konstrukti naj bi omogočali specificiranje, kreiranje in prekinjanje procesov ter njihovo medsebojno komuniciranje.

9.1. Tradicionalni pristop k multitasking programiranju

Preden se posvetimo obravnavanju multitasking konstruktov v višjenivojskem programskem jeziku, si oglejmo tradicionalni pristop (brez multitasking konstruktov v jeziku), ki temelji na uporabi sekvenčnega višjenivojskega jezika in multitasking operacijskega sistema. Posamezne programe, ki tečejo na multitasking sistemu imenujemo procesi ali opravila. Proces je lahko *ready* (current, ready), ali *not-ready* (suspended, waiting, sleeping, receiving). Princip delovanja teh sistemov je tak, da poseben proces, ki je vezan na uro realnega časa (*scheduler*) po določenem algoritmu dodeljuje procesor posameznim procesom. Obstajajo različni algoritmi dodeljevanja (*time-slicing, round-robin, priority*), pri real-time sistemih pa je najpogostejše prioritetno izpraznjevalno dodeljevanje (*preemptive priority based scheduling*). Pri tem načinu se vedno izvaja proces z najvišjo prioriteto med *ready* procesi in to toliko časa, dokler drugi proces s še višjo prioriteto ne pride v stanje *ready* (zunanji dogodek ali iztek časovnega intervala). Takrat ta proces takoj zaseže procesor. Dodeljevanje procesorja (*context-switch*) se izvaja sinhrono v določenih časovnih presledkih. Proces, ki je bil prekinjen bo nadaljeval z izvajanjem, ko bo ponovno imel najvišjo prioriteto med *ready* procesi in to na mestu, kjer je bil prekinjen. Žal nam multitasking poleg naštetih prednosti prinaša tudi določene značilne probleme ki so posledica dejstva, da si procesi delijo nekatere skupne vire in da je proces lahko prekinjen v izvajanju v kateremkoli trenutku, tudi sredi izvajanja ukaza. Če te probleme ne obravnavamo pravilno, lahko povzročijo napake v delovanju , ali celo izpade sistema, ki jim zelo težko najdemo vzrok. To so problemi, ki nastopajo v zvezi z medsebojnim izključevanjem procesov ter njihovo medsebojno komunikacijo in sinhronizacijo /3/.
Kot primer bomo vzeli dva procesa, ki vpisujeta podatke v skupni vmesnik:

```
      P1                      P2
      .                       .
      .                       .
buffer[i]:=data;        buffer[i]:=data;
i:=i+1;                 i:=i+1;
      .                       .
      .                       .
```

Sedaj predpostavimo naslednji scenarij izvajanja in preklapljanja med procesi:

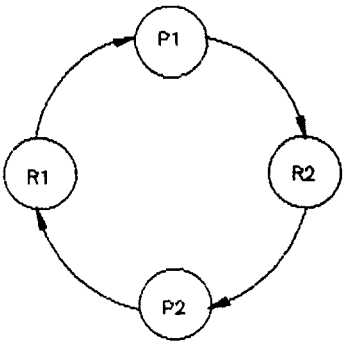
```
          .
          .
P1:  buffer[i]:=data;
P2:  buffer[i]:=data;
```

podatek, ki ga je vpisal P2 je "povozil" tistega, ki ga je vpisal P1.

```
P2:  i:=i+1;
P1:  i:=i+1;
          .
          .
```

Očitno je, da proces P1 ne bi smel biti prekinjen med izvajanjem ukazov vpisovanja v buffer in povečevanja spremenljivke i.
Tak niz ukazov imenujemo *kritični odsek*. Kritični odsek je torej tisti odsek programa, kjer se zahteva *medsebojno izključevanje* procesov pri dostopu do skupnih virov. Ti viri niso samo spremenljivke, ampak tudi diskovne enote, I/O enote, itd.
Ker je verjetnost, da bo proces P1 prekinjen ravno med izvajanjem ukazov znotraj kritičnega odseka zelo majhna, se nam lahko zgodi, da napake ne bomo odkrili tudi z večkratnim testiranjem ter da do napake pride med rednim obratovanjem sistema. Take napake se navadno pojavljajo v daljših, neenakomernih časovnih presledkih, kar zelo otežkoča njihovo odkrivanje.
Obstajajo različni načini (protokoli) za doseganje medsebojnega izključevanja. Pri vseh teh načinih se sicer znebimo nevarnosti kršitve medsebojnega izključevanja, ostanejo pa nekatere druge nevarnosti:

- pri zelo pogostem zaseganju določenega vira od strani več procesov z različnimi prioritetami, se lahko procesu z najnižjo prioriteto zgodi, da ostane blokiran za nedoločen čas, ali celo, da nikoli ne pride na vrsto
- če dovolimo možnost, da en proces hkrati zasega več virov, lahko pride do "smrtnega objema" (*deadlock*).



Zaprt graf dodeljevanja virov pomeni deadlock.

Bilo je več poskusov reševanja problema medsebojnega izključevanja. Nekateri temeljijo na uvajanju določenih kontrolnih spremenljivk, ki jih procesi testirajo in jim prirejajo vrednosti pred vstopom v kritični odsek (npr. Dekker-jev algoritem /3/). Ti algoritmi imajo pomanjkljivosti, ki izhajajo iz dejstva, da temeljijo na principu "*busy waiting*". Manj huda posledica tega je zapravljanje procesorjevega časa, kar v real-time sistemu načelno ni dopustno, hujša posledica pa je možnost *smrtnega objema*.
Zato vsak multitasking sistem vsebuje vsaj en mehanizem medsebojnega izključevanja, ki deluje na principu blokiranja dejavnosti. Najenostavnejši, pa vendar precej učinkovit tak mehanizem je *semafor*. To je mehanizem, ki ga je enostavno implementirati in ki je zadosti močan za reševanje večine problemov pri multitasking programiranju (medsebojno izključevanje procesov ter njihova medsebojna sinhronizacija in komunikacija).
Semafor je pozitivna celoštevilčna spremenljivka. Potem ko je semaforju prirejena začetna vrednost, sta nad njim dovoljeni le naslednji dve operaciji:

```
wait(s):    IF s>0 THEN s:=s-1
                   ELSE suspend;
signal(s):  IF kakšen proces blokiran na
               semaforju (klic wait)
            THEN ga zbudi ELSE s:=s+1;
```

Pri semaforju je bistveno to, da sta operaciji **wait** in **signal** neprekinljivi, s čimer je zagotovljeno medsebojno izključevanje procesov pri operacijah na semaforju.
Definicija operacije signal ne določa procesa ki bo zbujen, če je na semaforju blokiranih več procesov. To je stvar implementacije, lahko pa imamo FIFO princip, prioritetni princip, ali pa kak bolj zapleten algoritem, ki zagotavlja "fair" obravnavanje posameznih procesov.
Semafor, ki lahko zavzame poljubno pozitivno vrednost, imenujemo splošni semafor, tistega, ki lahko zavzame le vrednosti 0 in 1 pa binarni semafor.
Sedaj lahko zgoraj podani primer kršitve medsebojnega izključevanja rešimo tako:

```
      P1                      P2
       .                       .
       .                       .
WAIT(s);                WAIT(s);
buffer[i]:=data;        buffer[i]:=data;
i:=i+1;                 i:=i+1;
SIGNAL(s);              SIGNAL(s);
       .                       .
       .                       .
```

Na prvi pogled se zdi, da je problem medsebojnega izključevanja s pomočjo semaforjev zadovoljivo rešen, pa vendar temu ni tako. Semaforji so zelo nizkonivojski konstrukt, kar lahko ob njihovi nepravilni uporabi pripelje do hudih napak. Celo njihova pravilna uporaba praviloma pripelje do zamegljenih programov /1/,/2/. Možne napake pri uporabi semaforjev so:

- ni zagotovila, da bo proces, ki izvede **wait** izvedel tudi **signal**
- obstaja možnost, da pozabimo uporabiti semafor pri kritičnem odseku
- vir, ki ga semafor ščiti pred hkratnim dostopom več procesov, še vedno ni zaščiten pred nepravilno uporabo od strani procesa, ki ga je zasegel
- po izvorni definiciji ni predviden nedeterminizem pri proceduri **wait** (alternativna akcija če je s=0 ali čakanje na signal na kateremkoli od več semaforjev)
- nižjeprioritetni proces, ki izvede **wait** lahko za nedoločen čas ustavi izvajanje višjeprioritetnega, ki za njim izvede **wait** na istem semaforju; ta zakasnitev je lahko zelo dolga, če med tem postane aktiven tretji, srednjeprioritetni proces (ta problem lahko rešimo z začasnim zviševanjem prioritete procesu, ki je zasegel semafor, do navišje prioritete med procesi, ki čakajo na tem semaforju).

Pri vseh napakah v zvezi z nepravilno uporabo semaforjev pa je bistveno to, da nastopijo šele med izvajanjem programa, saj jih prevajalnik ne more odkriti.

9.2. Konstrukti multitasking programiranja v programskih jezikih

Rešitev problemov, ki nastanejo pri tradicionalnem pristopu k multitasking programiranju je v uvajanju konstruktov multitasking programiranja v visokonivojski programski jezik in tako omogočiti, da že prevajalnik odkrije čim več napak. Na nivoju jezika naj bi imeli možnost deklarirati procese, jih klicati ter izvajati komunikacijo in sinhronizacijo med njimi, kakor tudi zagotavljati pravilno delitev skupnih virov.

9.2.1. Procesi

Smiselno je proces podoben proceduri. Deklaracija procedure je opis poteka ukazov, klic procedure pa povzroči izvajanje teh ukazov. Enako je pri procesu, s to razliko, da se proces izvaja paralelno s programom, ki ga je klical.

```
PROGRAM process_example;
  PROCESS A
    BEGIN . . . END;
BEGIN  A; . . . END.
```

Deklaracija in klic procesa lahko vsebujeta tudi parametre. Isti proces lahko kličemo večkrat - vsak klic imenujemo konkretizacija (instance). Slabost tega načina je, da so posamezne konkretizacije procesa anonimne in da lahko obstaja veliko število konkretizacij. Omejitev na po eno konkretizacijo pri vsakem procesu pa bi bila prehuda. Rešitev je v uvajanju polja procesov, kjer omejimo število konkretizacij, posamezne konkretizacije pa tudi niso več anonimne /1/,/2/.

```
PROGRAM polje_procesov;
  TYPE task_id = (scan1,scan2,scan3);
  PROCESS task[task_id]
    BEGIN . . . END;
BEGIN
INIT task[scan1],task[scan2],task[scan3];
 . . .
END.
```

V zgornjem primeru lahko obstajajo največ tri konkretizacije.
Stavek **init** zagotavlja hkraten začetek izvajanja vseh procesov.
Tip **process** pozna več sodobnih programskih jezikov (*Concurrent Pascal, Modula-2, Ada*). Najdalj je v to smer gotovo šla *Ada*, pri kateri je **task** temeljni tip in ki nasploh med visokonivojskimi multitasking konstrukti pozna samo **task** ter sofisticiran način komunikacije med taski.
Polja procesov kot tip najdemo pri jeziku *Concurrent Pascal*.

9.2.2. Monitorji

Zaradi opisanih pomanjkljivosti semaforjev so bili razviti različni višjenivojski konstrukti za zagotavljanje medsebojnega izključevanja.
Kot izredno učinkovit mehanizem se je izkazal monitor. To je v bistvu modul z vsemi prej navedenimi lastnostmi ter z dodatno lastnostjo, da samo en proces lahko v določenem trenutku izvaja monitorsko proceduro v določenem monitorju.
Torej je monitor visokonivojski mehanizem za implementacijo izključnega dostopa do skupnega vira /1/,/2/.
Tukaj je treba omeniti tip **condition**. Ta je podoben semaforju; operacije so **wait** in **signal**. Na **condition** čakajoči procesi tvorijo FIFO vrsto.
Klic **wait**(c) sprošča mehanizem medsebojnega izključevanja, zaradi tega, da bi bilo omogočeno drugem procesu vstopiti v monitor in s klicem **signal**(c) obuditi blokirani proces. Klic **signal**(c) mora biti zadnji klic znotraj monitorja, ker s tem ohranimo princip medsebojnega izključevanja (druga možnost je da proces ki izvrši signal ostane blokiran dokler proces ki ga je ta signal obudil ne zapusti monitorja /2/).

Primer: Uporaba monitorja pri asinhronem prenosu podatkov med dvema procesoma. Primer je v jeziku *Pascal Plus*, kjer se imena navzven vidnih objektov začnejo z "*".

```
PROGRAM m_example;
  MONITOR buffering;
    VAR b:BUFFER;
    nonfull,nonempty:CONDITION;

    PROCEDURE *transmit(IN x:DATA);
      BEGIN
        IF full THEN wait(nonfull);
        put(x);signal(nonempty);
      END;
    PROCEDURE *receive(OUT x:DATA);
      BEGIN
        IF empty THEN wait(nonempty);
        get(x);signal(nonfull);
      END;
    FUNCTION full:BOOLEAN;
      BEGIN . . . END;
    FUNCTION empty:BOOLEAN;
      BEGIN . . . END;
    PROCEDURE put(IN x:DATA);
      BEGIN . . . END;
    PROCEDURE get(OUT x:DATA);
      BEGIN . . . END;
  BEGIN
  (* inicializacija bufferja - in,out,n *)
  END; (* monitor *)

  PROCESS producer;
    VAR x:DATA;
    BEGIN
      LOOP produce(x);
           buffering.transmit(x);
      END LOOP;
    END;
  PROCESS consumer;
    VAR x:DATA;
    BEGIN
      LOOP buffering.receive(x);
           consume(x);
      END LOOP;
    END;
BEGIN
  INIT producer,consumer
END.
```

Monitorski moduli so se izkazali kot zelo dobra rešitev za različne probleme in so zato verjetno najpogosteje uporabljan mehanizem medsebojnega izključevanja v modernih programskih jezikih. Dobre lastnosti monitorjev izhajajo iz njihove modularne zasnove in so enake kot za module. Koncept monitorja sicer še vedno potrebuje nizkonivojske operacije **wait**(condition) ter **signal**(condition), vendar so te operacije skrite v monitorju in tako nedosegljive zunanjemu okolju.
Pa vendar imajo tudi monitorji dve pomanjkljivosti:

1. Proces, ki zapušča monitor lahko signalizira le enemu procesu, kar je včasih slabo.
2. Če dovolimo, da se iz monitorja A kliče procedura v monitorju B (gnezdeni monitorski klic) in tukaj ostane blokirana po klicu wait(c), je mehanizem medsebojnega izključevanja sproščen v monitorju B, ni pa v A. Ob predpostavki, da je edini vhod v B skozi A, pride do "*smrtnega objema*".

Na koncu je treba dodati še to, da je monitor zasnovan za zaščito skupnega pomnilnika. Ker distribuirani sistemi, ki postajajo vedno bolj popularni, navadno nimajo skupnega pomnilnika, je to še en razlog za iskanje novega pristopa k medprocesni interakciji.
Monitor se je prvič pojavil v programskem jeziku *Concurrent Pascal*. Za tem je nastalo še nekaj jezikov, ki vsebujejo ta mehanizem, med njimi npr. *Pascal Plus* in *Modula-2*.
Ada vsebuje drugačen mehanizem interakcije med procesi, ki je opisan v nadaljevanju.

9.2.3. Rendezvouz

Razvoj komunikacijskih mehanizmov med procesi je bil v tesni zvezi z razvojem računalniške materialne opreme na katero so bili implementirani. Multitasking programska oprema je bila implementirana na enoprocesorskih računalnikih in so zato procesi komunicirali preko skupnega pomnilnika. Ta model je vplival na koncept komunikacijskih mehanizmov, npr. monitorja.
Novejši koncepti, ki izhajajo iz distribuiranih sistemov, predpostavljajo hkratno pošiljanje podatka in sinhronizacijo. Ob komunikaciji dveh procesov mora biti zahteva za to obojestranska. Če en proces zahteva komunikacijo prej kot drugi, bo blokiran do zahteve drugega. Ko sta oba procesa sinhronizirana, pride do prenosa podatkov /1/.
Če se hočemo izogniti čakanju, moramo vpeljati tretji proces - vmesnik, ki je aktiven task, za razliko od pasivnega monitorja.
Pri rendezvouz-u poznamo simetrični (ko procesa imenujeta drug drugega) in asimetrični princip (ko samo en proces imenuje drugega). Asimetrični princip je boljši, ker je splošnejši in omogoča gradnjo knjižnic procesov. Ta princip je tudi uporabljen pri jeziku *Ada*.
Na abstraktnem nivoju jezika je asimetrični rendezvouz predstavljen z **accept** stavkom v klicanem procesu.

```
PROCESS A;
  VAR x:DATA;
BEGIN B.send(x); END.
```

```
PROCESS B;
  VAR y:DATA;
BEGIN
  ACCEPT send(IN item:DATA); y:=item; END
END.
```

Da proces ne bi čakal na accept (deterministični klic) je vpeljan stavek **select** (nedeterministični klic).

```
  PROCESS protected_var;
    VAR shared_var:DATA;
  BEGIN
    LOOP
      SELECT
        ACCEPT read(OUT x:DATA);
               x:=shared_var;
        END
      OR
        ACCEPT write(IN x:DATA);
               shared_var:=x;
        END
      END SELECT
    END LOOP
  END.
```

Pri izvajanju stavka **select** obstajajo štiri možnosti:

1. Klic read "visi"
2. Klic write "visi"
3. Oba klica "visita"
4. Ni klica

V primerih 1 in 2 se takoj izvrši ustrezni accept stavek. V primeru 3 se naključno izvrši eden od accept stavkov. V primeru 4 je proces blokiran do klica read ali write in potem se izvrši ustrezen accept stavek.

V določenih primerih rabimo še dodaten pogoj za vstop v accept stavek, ki je odvisen od stanja podatkovne strukture v strežnem procesu:

```
PROCESS buffering;
  VAR b:BUFFER;
BEGIN
  LOOP
    SELECT
      WHEN not full =>
        ACCEPT transmit(IN x:DATA); put(x);END
    OR
      WHEN not empty =>
        ACCEPT receive(OUT x:DATA); get(x);END
    END SELECT
  END LOOP
END.
```

Slabost koncepta rendezvouz v primerjavi z monitorji je v tem, da za reševanje istega problema rabimo več procesov, kar predstavlja večjo porabo časa pri razporejevalniku.

10. ZAKLJUČEK

Z naraščanjem deleža stroškov razvoja in vzdrževanja programske opreme pri razvoju sistemov vodenja v realnem času, se vedno bolj kaže potreba po uporabi strukturiranih visokonivojskih programskih jezikov z strogo zahtevo enakosti tipov (imensko ali strukturno), sofisticiranimi mehanizmi abstrakcije ter mehanizmi multitasking programiranja na visokem nivoju. Nujnost uporabe takega programskega jezika je sorazmerna z velikostjo sistema, verjetnostjo pogostega spreminjanja, dopolnjevanja, ali celo prenašanja programske opreme ter s številom programerjev v ekipi. Ob upoštevanju vsega tega in izkušenj iz preteklosti, postaja uporaba sodobnih jezikov nujnost, izbira posamezni aplikaciji ustreznega jezika pa bistvena odločitev pred začetkom dela na projektu.

11. LITERATURA

/1/ Stephen J. Young: Real Time Languages: Design and Development, Ellis Horwood Limited, Chichester, England, 1982

/2/ David Bustard, John Elder and Jim Welsh: Concurrent Program Structures, Prentice Hall International, UK, 1988

/3/ M. Ben-Ari: Principles of Concurrent Programming, Prentice Hall, New Yersey, 1982

/4/ John W. L. Ogilvie: Modula-2 Programming, McGraw-Hill, New York, 1985

/5/ Logitech Modula-2 Users Manual, Logitech, Fremont, California, 1987

/6/ Kjell Nielsen, Ken Shumate: Designing Large Real-Time Systems with ADA, McGraw-Hill, New York, 1988

/7/ Niklaus Wirth: Programming in Modula-2, Springer-Verlag, 1982

/8/ Shem-Tov Levi, Ashok K. Agrawala: Real-Time System Design, McGraw-Hill, New York, 1990

DINAMIČNA NEVRONSKA MREŽA ZA KLASIFIKACIJO VZORCEV

INFORMATICA 4/90

Jelena Ficzko
U. Rezar
A. Dobnikar
D. Podbregar
Fakulteta za elektrotehniko in računalništvo
Tržaška 25, 61000 Ljubljana

Keywords: Pattern Classification, Invariance to Size and Rotation, Normalization, FFT, Asociative Neural Network, Dynamic Neural Network.

POVZETEK

V članku je predstavljena primerjava med dvema metodama za klasifikacijo objektov na črno belih slikah. Klasična metoda, ki uporablja hitro Fourierjevo transformacijo, sestoji iz postopka normalizacije in Fourierjeve transformacije, katere spektralni deskriptorji so invariantni na rotacijo in translacijo. V prispevku je obravnavana samo klasifikacija neodvisno od rotacije in velikosti objektov. Predpostavljeno je, da se objekt nahaja v središču slike. Alternativni pristop uporablja posebno metodo nevronskega modeliranja. Pokazano je, da predlagana nevronska mreža, ki je kombinacija dinamičnega in asociativnega nivoja, enako dobro, v določenih primerih pa tudi boljše, klasificira objekte kot klasična metoda.

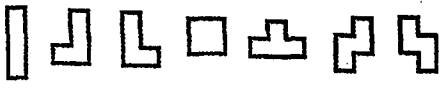
ABSTRACT

Invariant Pattern Classification of the black-white images is studied in two directions. Classical Fast Fourier Transform (FFT) approach consists of normalization procedure for excluding magnification or size influence and FF transformation whose output spectrum descriptors are invariant to rotation, and/or translation. Only pattern classification independent of magnification and rotation is concerned in this paper. It is supposed that the observed object is moved in the center of the image. An alternative aproach uses Neural Network Modeling (NNM) to achive the same effect . It is shown that the proposed Neural Network with the combination of dynamic and asociative properties solves the same problem equally good (or even better) with additional potential capability of considerable higher parallel procesing.

I. UVOD

V članku bomo predstavili rezultate primerjave dveh metod za razpoznavanje objektov na črno belih slikah. Prva metoda, ki bi jo lahko imenovali klasična, uporablja pri razpoznavanju objektov metodo hitre Fourierjeve transformacije, druga alternativna rešitev pa se poslužuje posebne metode nevronskega modeliranja.

Obe metodi smo preizkusili na sedmih različnih objektih (Sl. 1), ki jih dobimo s kombinacijami štirih enakih kvadratov in so dobro poznani igralcem igre TETRIS. Naloga obeh metod je bila razpoznati poljuben objekt, iz danega nabora objektov, ki se od referenčnega objekta svoje vrste razlikuje v velikosti, kotu zasuka okoli masne točke in določeni količini šuma oz. motenj.



Slika 1: TETRIS objekti

Predvidevali smo, da se objekt nahaja v središču slike, kar pomeni, da se njegova masna točka pokriva s središčno točko slike.

II. KLASIČNI ALI FFT PRISTOP

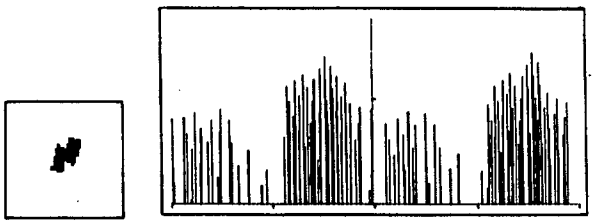
Postopek razpoznavanja objektov na črno beli sliki pri tem pristopu sestoji iz treh korakov. Prvi korak je postopek normalizacije, ki izloči faktor velikosti objekta. Pri normalizaciji najprej pretvorimo kartezične koordinate točk, ki sestavljajo objekt (Sl.2a), v polarno obliko glede na enačbe:

$$\alpha = \text{Arc tg}\left[\frac{x}{y}\right] \qquad 1)$$

$$r = \sqrt{x^2 + y^2} \qquad 2)$$

Po pretvorbi koordinat radije točk objekta normiramo z radijem kroga (En. 3), ki ima isto površino kot objekt. Površino objekta v tem primeru določa kar število točk Np, ki objekt sestavljajo. Rezultat normalizacije je predstavjen na sliki 2b.

$$r_{norm} = \frac{r}{\sqrt{Np / \pi}} \qquad 3)$$



a. b.

Slika 2: Vhodni objekt (a) in njegova predstavitev v normirani polarni obliki (b)

Naslednji korak je izvedba Fourierjeve transformacije nad zaporedjem x(n), ki ga sestavljajo največji radiji točk pri danem kotu. Zaporedje ima torej 360 komponent, katerih vrednosti so normirane razdalje med centrom slike in konturo objekta v točkah, ki odgovarjajo določenim kotom. Izhodno zaporedje transformacije je izraženo z enačbo 4).

$$X_k = \sum_{n=0}^{360} x_n \cdot e^{-j \cdot \frac{2 \cdot \pi}{N} \cdot n \cdot k} \qquad 4)$$

Rezultat Fourierjeve transformacije predstavlja spekter objekta v frekvenčnem področju.

Rotacija objekta se kaže na vhodnem vektorju Fourierjeve transformacije kot rotacija komponent vektorja, lahko pa si jo predstavljamo kot zamik periodičnega zaporedja s periodo x(n) za določeno število elementov zaporedja. Če vhodnemu zaporedju x(n) odgovarja transformacija X(k) (En. 5), potem zaporedju x(n + n0) odgovarja zaporedje predstavljeno z desno polovico enačbe 6).

$$x_n \longrightarrow X_k \qquad 5)$$

$$x_{n - n_0} \longrightarrow X_k \cdot e^{-j \cdot \frac{2 \cdot \pi}{N} \cdot n \cdot n_0} \qquad 6)$$

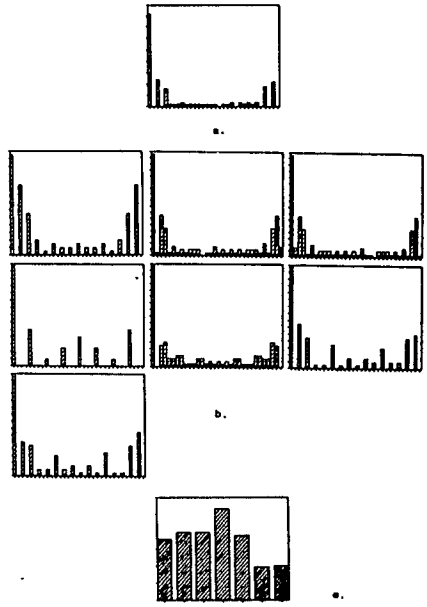
Ker nam je zamik oziroma kot zasuka iz osnovne referenčne lege neznan, smo rešitev našli v amplitudnem spektru (En. 7) dane predstavitve v frekvenčnem prostoru, ki je neobčutljiv na zamik zaporedja.

$$|X_k| = \sqrt{x^2 + y^2} \qquad 7)$$

Tretji ali končni korak te metode je primerjanje amplitudnega spektra danega objekta (Sl. 3a) z amplitudnimi spektri referenčnih objektov (Sl. 3b) po metodi najmanjših kvadratov (Sl. 3c). Rezultat tega koraka pa je končna klasifikacija vhodnega vzorca.

III. PRISTOP Z NEVRONSKIM MODELIRANJEM

Skupna lastnost vseh pristopov z nevronskim modeliranjem so adaptivne uteži, ki v učilni fazi težijo k povezovanju določenega izhodnega vzorca izbranemu vhodnemu vzorcu, tako da bi kasneje dobili ustrezne odzive nevronske mreže na poljubne vhodne podatke iz dane domene.



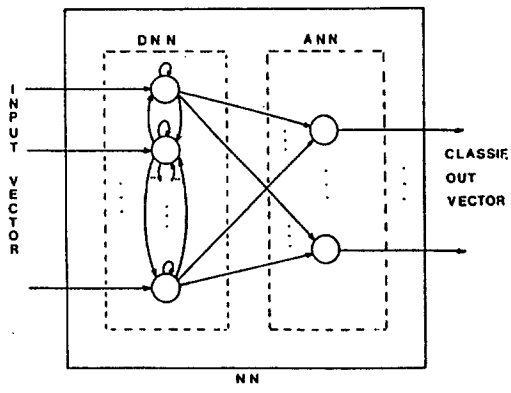
Slika 3: Amplitudni spekter objekta (a), amplitudni spektri referenčnih objektov (b) in vektor napake (c).

Pri našem delu smo se osredotočili na takoimenovani distribuirani asociativni pomnilnik. Glavna razloga, da smo se odločili za reševanje našega problema s tako metodo asociativnega iskanja sta posploševanje, ki zmanjša število potrebnih asociativnih parov in določen nivo tolerantnosti na motnje v vhodnih vzorcih, ki omogoča pravilen izhodni vzorec tudi pri motenem ali nepopolnem vzorcu. Ti dve lastnosti sta tudi lastnosti kognitivnih sistemov ali sistemov z določenim nivojem inteligence.

Vzemimo, da je v našem primeru vhod predstavljen z vektorjem x in izhod z vektorjem y. Asociativni pomnilnik poveže torej pare (x,y) tako da za vhod x dobimo ustrezen izhod y. Med fazo učenja poteka prilagajanje distiburianega pomnilnika oziroma uteži v smislu razlik med dobljenim in željenim odzivom na določeni vhodni vektor.

Ker želimo razpoznavanje objektov ne glede na njihovo velikost oziroma kot zasuka, moramo dodati mehanizme, ki zmanjšujejo potrebno število učilnih parov, saj je velikost asociativnega pomnilnika neposredno odvisna od števila parov.

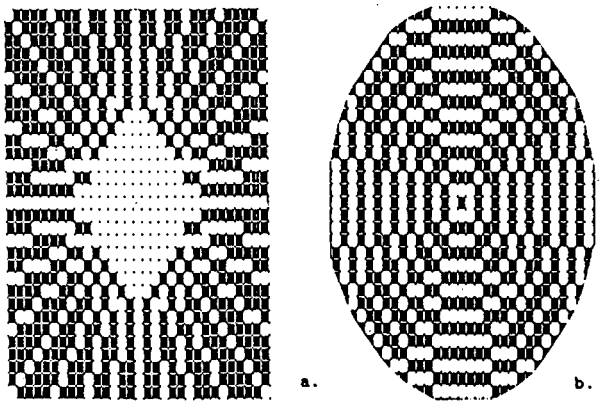
Da bi se izognili dolgemu učenju smo osnovni mreži dodali dodatni dinamični nivo Hopfieldovega tipa (Sl. 4), pri katerem so vsi elementi medsebojno povezani. Ta dodatni nivo omogoča rotacijo vhodnega vzorce in neke vrste uglaševanje na najboljše ujemanje na naslednjem asociativnem nivoju.



Slika 4: Nevronska struktura za invariantni klasifikator vzorcev

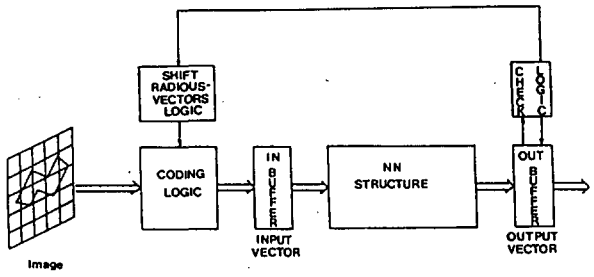
Za zmanjševanje števila učilnih vzorcev, ki pa še vedno omogoča invariantno razpoznavanje glede na zasuk in velikost, smo izbrali posebno kodiranje vhodnega vzorca, ki temelji na presečnih točkah radij vektorjev in koncentričnih krogov (Sl. 5). S tem načinom smo želeli doseči to, da bi se zasuk objekta odražal v rotaciji radij vektorjev.

Nevronska stuktura izvede postopek klasifikacije na naslednji način. Prvi nivo strukture izvede transformacijo, ki odgovarja rotaciji vhodnega objekta za določen kot, drugi nivo pa klasificira vzorce s prvega nivoja. Popolna organizacija invariantnega razpoznavalnika je podana na sliki 6.



Slika 5: Radij vektorji (a) in vzorčevalni krogi (b)

Dani objekt kodiramo na zgoraj podani način in ga pripeljemo na vhod prvega nivoja nevronske strukture. Vzorec povzroči v dinamični mreži najprej preslikavo, ki ustreza rotaciji vzorca za določen kot. Rezultat pa se nato preslika preko asociativne mreže v izhodni vektor. Takšen prehod preko obeh mrež se nato ponovi (enkrat ali večkrat), kar pomeni ponoven zasuk vhodnega vzorca in zopet njegovo klasifikacijo. Klasifikacija je zaključena takrat ko ima izhodni vektor aktivno vedno isto komponento pri vseh prehodih. V nasprotnem primeru se izvede uglaševanje na prvem nivoju in ponovi postopek.



Slika 6: Shema invariantnega klasifikatorja

Za uglaševanje so potrebne določene kriterijske funkcije. Če ne moremo doseči tolerančnih mej kriterijske funkcije, danega vzorca ne moremo razpoznati z določeno stopnjo zanesljivosti. Vsekakor pa obstaja še možnost, da vzamemo za razpoznan tisti objekt, katerega nevron je dosegel najvišjo vrednost.

IV. FAZA UČENJA IN PROCESIRANJA V NEVRONSKI STRUKTURI

Dinamični nivo nevronske strukture je potrebno naučiti na zasuke objektov za določen kot, v našem primeru 22.5 stopinj. Vhodni vektor v dinamični nivo ima 256 komponent (16 radij vektorjev s po 16 komponentami), takšno pa je tudi število nevronov v dinamičnem nivoju. Zaradi polne povezanosti je v fazi učenja torej potrebno določiti 65636 uteži. To storimo s primerjavo izračunanega izhoda dinamičnega nivoja za posamezne referenčne objekte z ustrezno modificiranim vhodnim vektorjem teh referenčnih objektov. Neujemanje povzroči spremembo uteži.

Podobno poteka tudi faza učenja v asociativnem nivoju mreže s tem, da je vhodni vektor v asociativni nivo izhodni vektor dinamičnega nivoja, izhodni vektor asociativnega nivoja pa klasifikacijski vektor za objekt iz slike.

Modifikacijo uteži za oba nivoja lahko podamo z enačbama:

$$dw_{ij} = \alpha \, (y'_i - \sum_j w_{ij} \cdot x_j) \cdot x_j \qquad 8)$$

$$i = 1,\ldots,7 \qquad j = 1,\ldots,256$$

$$dq_{ij} = \alpha \, (z'_i - \sum_j q_{ij} \cdot y_j) \cdot y_j \qquad 9)$$

$$i = 1,\ldots,7 \qquad j = 1,\ldots,256$$

pri tem je y'_i želeni, vsota produktov $w_{ij} \cdot x_j$ izračunani izhod dinamičnega nivoja, α pa učilni parameter, odvisen od števila opravljenih iteracij na množici učilnih parov. Podobno je z'_i želeni, vsota produktov $q_{ij} \cdot y_j$ pa izračunani izhod asociativnega nivoja.

V fazi procesiranja vhodni vektor, dobljen z opisanim kodiranjem slike, vstopa v dinamični nivo, kjer le-ta vzpostavi povezavo med pozicijo objekta na sliki in pozicijo referenčnega objekta iz faze učenja. Izhodni vektor dinamičnega nivoja vstopa v asociativni nivo, ki izvede klasifikacijo objekta.

Procesiranje v obeh nivojih lahko podamo z enačbama:

$$y_i = \sigma \left(\sum_{j=1}^{256} w_{ij} \cdot x_j - T_i \right) \qquad 10)$$

$$i = 1,\ldots,256$$

$$z_i = \sigma \left(\sum_{j=1}^{256} q_{ij} \cdot y_j - \tau_i \right) \qquad 11)$$

$$i = 1,\ldots,7$$

Funkcija σ je določena takole :

$$\sigma(\,.\,) = \begin{cases} 1 \,, & \text{if } (.) > 0 \\ 0 \,, & \text{else} \end{cases} \qquad 12)$$

Na koncu dodajmo še nekaj besed o kapaciteti nevronske mreže. Izhod iz asociativne nevronske mreže je določen z naslednjo enačbo:

$$z = y \cdot W \qquad 13)$$

kjer je W matrika uteži, y pa učilni vektor. Za p učilnih vektorjev pa velja enačba :

$$Z = Y \cdot W \qquad 14)$$

Matrika Y je matrika p učilnih vektorjev. Če je število učilnih vzorcev enako številu vhodov v posamezen nevron, potem uteži lahko enostavno določimo iz enačbe :

$$W = Y^{-1} \cdot Z \qquad 15)$$

V primeru, ko je število učilnih vzorcev večje kot izbrano število vhodov, lahko število vhodov povečamo ali pa uteži izračunamo iz enačbe :

$$W = (Y^T \cdot Y^{-1}) \cdot Y^T \cdot Z \qquad 16)$$

V primeru uporabe zgornje enačbe si zagotovimo samo najboljše ujemanje s pravilno rešitvijo, kar pa pri postopku klasifikacije ne zadostuje. Pravilo o kapaciteti torej zahteva, da je število učilnih parov manjše ali enako številu vhodnih komponent,

saj mora biti za pravilnost delovanja mreže zagotovljena še linearna neodvisnost, ki pa nam je največkrat nepoznana.

V. REZULTATI TESTIRANJA

Ob primerjavi uspešnosti pristopa s FFT in pristopa z nevronskim modeliranjem, smo za prvi test naključno izbrali slike z objekti iz učne množice, za drugi test pa slike z objekti iz celotne vhodne domene. Kasneje smo s tako izbranimi objekti test ponovili, le da smo slikam naključno dodali še 2% šuma.

Rezultati (Tabela 1.) so pokazali, da daje pristop z nevronskim modeliranjem bistveno boljše rezultate v primeru prisotnosti šuma.

	Sum = 0 %		Sum = 2 %	
	NN [%]	FF [%]	NN [%]	FF [%]
domena A	100	95	93	20
domena B	83	93	80	17

Domena A: Naključno izbrani objekti iz učne množice
Domena B: Naključno izbrani objekti

Tabela 1: Rezultati testiranja uspešnosti klasifikacije pri uporabi klasične metode in metode z nevronskim modeliranjem

VI. ZAKLJUČEK

V članku opisana metoda za invariantno klasifikacijo objektov na črno-belih slikah, ki uporablja pristop z nevronskim modeliranjem, je kombinacija asociativne in dinamične nevronske mreže. Ta omogoča sekvenčno iskanje najboljšega klasifikacijskega rezultata s povratno zanko in uglaševanjem. V primerjavi z metodo hitre Fourierjeve transformacije daje boljše rezultate v primeru prisotnosti šuma. Seveda pa je njena prednost tudi v možnosti realizacije z aparaturno opremo, kjer pride paralelnost procesiranja v nevronskih mre'ah dejansko do izraza.

Izboljšanje pristopa z nevronskim modeliranjem kaže iskati v smeri povečanja strukture v dinamičnem nivoju ter povečanju ločljivosti slike.

LITERATURA :

1. K.Fukusima /89/ : Analysis of the Process of Visual Pattern Recognition by the Neocognition, Neural Networks, Vol.2, No.6, 1989

2. G.A. Carpenter, S. Grossberg /87/ : ART 2 : Self-organization of stable category recognition codes for analog input patterns, Applied Optics, 26(23), 4919-4930, 1987

3. T. Kohonen /84/ : Self-organization and Associative Memory, New York, Springer Verlag, 19844. Y.H. Pao : Adaptive Pattern Recognition and Neural Networks, Addison Wesley, 1989

5. M. Seibert, A.M. Waxman /89/ : Spreading Activation Layers, Visual Cascades and Invariant Representations for Neural Pattern Recognition Systems, Neural Networks, Vol.2, No.1, 1989

6. L. Udpa, S.S. Udpa /89/ : Application of Neural Networks to Nondestructive Evaluation, 1-st Int. Conf. on Artif. Neural Networks, London, 1989

7. J. Austin /89/ : ADAM : An Associative Neural Architecture for Invariant Pattern Classification, 1-st Int. Conf. on Artif. Neural Networks, London, 1989

8. L.R. Rabiner, B. Gold : Theory and Aplic. of Digital Signal Proccessing, Prentice-Hall, 1975

9. R.J. Baron /85/ : Visual memories and mental images, Int. Journal of Man-machine Studies, 23, 275-311, 1985

10. E.L. Schwartz /80/ : Computational anatomy and functional architecture of striate cortex : A spatial mapping approach to perceptual coding, Vision Research, 20, 645-669, 1980

11. T.R. Crimmins /82/ : A complete set of Fourier descriptors for two-dimensional shapes, IEEE Trans. SMC-12, No.6, 848-855, 1982

12. K. Arbter : Affine Invariant Fourier descriptors from pixels to features, J.C. Simon (ed.), Elsevier-Sci. Publ. B.V. (North-Holland). 1989

13. F.P. Kuhl : Global Shape Recognition of 3-D Objects Using a Differential Library Storage, Comp. Vision, Graphics and Image Processing, 27, 97-114, 1984

14. A. Dobnikar, A. Likar, D. Podbregar : Optimal Visual Tracking with Artificial Neural Networks, 1-st Int. Conf. on Artificial Neural Networks, London, 1989

15. A.C.C. Cooken, F.W. Kuijk /89/ : A Learning Mechanism for Invariant Pattern Recognition in Neural Networks, Neural Networks, Vol.2, No.6, 1989

Peter Zaveršek
Gorenje Servis, Velenje
Peter Kolbezen
Institut Jožef Stefan, Ljubljana

Keywords: LAN, CATV, Ethernet, Token Bus.

POVZETEK. Članek analizira možnost računalniških komunikacij s pomočjo sistema kabelske televizije. Podane so osnovne smernice za realizacijo ene ali več neodvisnih računalniških lokalnih mrež ali za medsebojno povezavo več različnih lokalnih mrež. Pri tem je sistem CATV uporabljen kot prenosni medij. Nakazane so možnosti uporabe takšnih mrež.

COMPUTER NETWORKS AND A CATV SYSTEM. This paper describes possibilito of providing computer communications within a system of cable television. CATV could be used as a transfer media in such a system. This enables one or more independent Local Area Networks or bridges between several different LANs to be obtained. Some examples of possible use are stated.

1. Uvod

Tehnologija prenosa slik, govora in računalniških podatkov sodi v informacijsko tehnologijo. Ta se iz dneva v dan bolj razvija in njena uporaba postaja vse bolj množična. Pri tem predstavlja medij za prenos informacij po mreži, ki povezuje številne uporabnike, pomembno komponento informacijskega sistema. Prav zato bi bil za realizacijo takega sistema ugoden večnamenski prenosni medij. Zanj zahtevamo:

- da je dovolj razširjen (dostopen širokemu krogu uporabnikov),
- da je po njem možno prenašati informacije v obe smeri,
- da je prenos zanesljiv, zadovoljivo hiter, in
- da je prenosna mreža v celoti ekonomsko upravičena.

Še bi lahko našli kakšno zahtevo, vendar zadnja (tj. cena) največkrat prevlada.

V primeru, ko želimo realizati novo večjo mrežo za prenos digitalnih podatkov (računalnške mreže), za lokacijsko raztresene uporabnike ali za povezave med oddaljenimi mrežami je smotrno, da razmislimo o možnosti uporabe morebiti že obstoječih prenosnih medijev. Takšni mediji so npr. telefonske parice, optični kabli in kabelska TV (CATV). Včasih lahko ugotovimo, da je cena nove mreže vsaj enaka, če ne večja od stroškov za ustrezno preureditev že obstoječe mreže v večnamensko.

Obstoječe telefonske parice lahko uporabimo za realizacijo računalniške mreže, če ne zahtevamo velikih hitrosti prenosa in če si to lahko privoščimo. Takšna mreža lahko zadovolji le zelo nezahtevne uporabnike, ki ne zahtevajo prenosa velikih količin podatkov od enega do drugega vozlišča v mreži. Če izvzamemo prenos preko optičnih vlaken, ki je zaradi še sorazmerno visoke cene uporaben le v bolj specifičnih primerih, ostane kot druga alternativa le kabelska TV.

CATV je širokopasoven medij, ki zaradi svoje velike pasovne širine dopušča velike hitrosti prenosa in je zato zelo dobra alternativa. V CATV-sistemih normalno prenašamo slikovne informacije. Prenos takšnih informacij mora biti visoko kvaliteten. Zato ni bojazni, da ta medij ne bi ustrezal tudi za realizacijio računalniških komunikacij.

Velika pasovna širina kabelske TV ponuja možnost sočasnega delovanja več različnih računalniških mrež in prenosa slikovnih in govornih informacij na istem prenosnem mediju. V svetu tečejo razvojna prizadevanja v tej smeri. Prve komercialno dostopne mreže ILAN (Integrated Local Area Network), ki bodo omogočale združitev podatkovnih, tonskih in slikovnih informacij, napovedujejo v tem desetletju. V prihodnosti bodo hrbtenico ILAN skoraj zanesljivo tvorila svetlobna vlakna zaradi majhnih izgub in velike neobčutljivosti na električne motnje in klimatske pogoje delovanja. Danes optični prenosi še niso dovolj razširjeni, saj zaradi omejene uporabe še niso dovolj ekonomsko zanimivi. Zato je TV kabel trenutno edini medij, ki je za realizacijo tovrstnih mrež tudi ekonomsko

sprejemljiv.

2. CATV

Kabelska TV je danes vse bolj razširjen prenosni medij. Namenjen je prvenstveno prenosu TV in radio signalov od nekega centralnega mesta do naročnikov. Prednost CATV pred običajnimi individualnimi sistemi je v tem, da je tipično na voljo več različnih TV-programov, slika pa je kvalitetnejša. Vsi kanali morajo zadoščati zahtevam standarda. Ni več individualnih TV anten in izgled kraja je prijaznejši. Nenazadnje lahko kabelska TV omogoča sprejem TV-signala tudi v krajih, kjer direktni sprejem ni možen zaradi naravnih ovir na poti TV-signala (razgiban relief). Dovolj je postaviti anteno na mesto, kjer je sprejem možen. Od tam do glavne distribucijske postaje je že možno voditi signal po CATV, v primeru večjih razdalj pa po optičnem kablu.

CATV je po svoji naravi širokopasoven prenosni medij. Prav zaradi velike pasovne širine in nezasedenosti so možnosti pri tašnih sistemih še neizkoriščene.

2.1 Značilnosti prenosnega medija

Običajni medij za prenos radijskih in TV-signalov je zrak. Prenosne frekvence za prenos signalov obsegajo področje med 47 in 860MHz. Narava elektromagnetnih valov v tem področju je taka, da se le minimalno lomijo; v glavnem potujejo premočrtno ali se odbijajo. Med pretvornikom (običajno na izpostavljeni legi) in sprejemnikom ne sme biti večjih ovir. Če signal na poti do sprejemnika naleti na oviro, ostane sprejemnik v 'senci'. Sprejem ni kvaliteten, ali pa sploh ni mogoč. V mestih, ki so običajno dobro pokrita z radio in TV-signali, se zaradi številnih stavb pojavljajo odboji. Odbiti signal pripotuje do sprejemne antene z rahlo zakasnitvijo v primerjavi z direktnim signalom. Posledica je dvojna slika in sprejem je nekvaliteten kljub dovolj močnemu signalu.

Pri kabelskih sistemih se tem težavam izognemo. Sprejemni center je postavljen na mesto, kjer je sprejem najugodnejši in kjer ni motenj. Signale prenašamo od sprejemnega centra do naročnikov po kablih, zato na njihovi poti ni ovir.

Osnovna naloga kabelske TV je posredovanje radio/TV signalov naročniku. Prenosni medij je zato prvenstveno namenjen prenosu teh signalov, mogoče pa ga je uporabiti še za prenos drugih informacij. Velik del razpoložljivega frekvenčnega območja je namreč neizkoriščen. Neizkoriščeno območje, ki je pod spodnjo frekvenčno mejo, je mogoče uporabiti za prenos poljubnih digitalnih podatkov v povratni smeri, to je od uporabnikov do glavne postaje.

Izgube pri prenosu rastejo z naraščajočo frekvenco prenosa. Zgornja meja frekvenčnega območja je običajno 450MHz (300MHz pri starejših sistemih). Sistemi do 600MHz so uporabni le v manjših CATV razvodih. Tam so izgube v prenosnih kablih sistema CATV zaradi krajših povezav še dovolj majhne. Še višja območja nad mejo 600MHz dandanes niso uporabna. Signale teh frekvenc lahko uporabimo kvečjemu znotraj stavbnega kompleksa (hoteli, poslovne stavbe ...) za interne ali dodatne zunanje kanale in le pod pogojem, da je notranji razvod realiziran dovolj kvalitetno.

Pri delitvi frekvenčnih pasov v smeri naprej/nazaj, ločimo pri CATV sistemih tri razrede:

razred	'nazaj'	'naprej'
Sub-split	5-30MHz	54MHz in več
Mid-split	5-108MHz	162MHz in več
High-split	5-174MHz	234MHz in več

Najbolj je uporabljan sub-split CATV-sistem, kar je razumljivo. Osnovna naloga kabelske TV je prenos radio/TV-signalov do naročnikov. Ob uporabi drugih dveh sistemov (high-split, mid- split) se območje za prenos radio/TV-kanalov zmanjša. Širše možnosti, ki jih zadnja dva sistema ponujata, obstajajo v tem, da je frekvenčni pas v obeh smereh dovolj širok in zato upoaben za realizacijo računalniških mrež.

2.2 CATV mreže pri nas

Informacijska tehnologija je pri nas slabo razvita. Med takšno tehnologijo sodi tudi TV in z njo CATV. Posledica nerazvitosti je, da dandanes še ni dovolj zahtev po hitrih računalniških komunikacijah, ki bi jih bilo možno vključevati v TV-sisteme. Zato uvajanje high- in mid-split sistemov ekonomsko še ni zanimivo.

Obstoječi sistemi

Pri nas je CATV zgrajena po sub-split sistemu. To pomeni, da je večina frekvenčnega območja namenjena prenosu signalov v glavni smeri, t.j. od glavne postaje do naročnika. Sistemi so še v veliki meri starejšega tipa (do 300MHz) in, takšni kot so, ne dopuščajo prenosa povratne informacije (podatkov) od naročnika do nekega centralnega mesta ali do poljubnega drugega naročnika.

Razvod CATV signala je dostikrat speljan brez nekega pravila. Veje, ki se odcepijo od glavnega voda, vodijo v kaskado naročnikov. Tak razvod je podoben razvodu pri stanovanjskih stolpnicah, kjer je običajno speljan glavni vertikalni vod, na njem pa ima vsak naročnik svoj odcepnik.

Dograjevanje sistemov

Starejše sisteme (300MHz) v splošnem lahko dogradimo za prenos podatkov in informacij v povratni smeri in sicer v področju 5- 25MHz. V ta namen jih moramo ustrezno preurediti. V ojačevalna mesta, ki so sicer enosmerna, vgradimo kretnice za povratno smer prenosa (premostitev ojačevalnikov) in če je potrebno dodamo še povratne ojačevalnike. Dodatno vgradnjo sorazmerno enostavno izvedemo v primarnem vodu (glavnem vodu). Težave se porajajo, ko se po raznih odcepih oddaljujemo od glavnega voda. Napeljava se deli v vedno več vej, vse do kaskade odcepnih mest, in vodi od stavbe do stavbe. Za izvedbo prenosa povratne informacije so problematična predvsem odcepna mesta v kaskadi (hišni odcepniki). Takšna mesta so največkrat enostavnejše izvedbe in so predvidena za notranje TV-razvode. Zato je njihovo delovno frekvenčno območje nad 30MHz. Povratnega frekvenčnega območja ne prepuščajo ali ga dušijo bolj, kot bi bilo zaželeno. Ti odcepniki so posebna ovira v nizu, ker zadušijo povratne signale vseh naslednjih naročnikov. Zato jih moramo zamenjati z delilniki in odcepniki, ki so namenjeni predvsem za CATV in delujejo v celotnem frekvenčnem področju 5-450MHz [6].

Podobno veljajo zgoraj zapisane ugotovitve tudi za novejše sisteme (do 450MHz). Razlika je le v tem, da so nekateri elementi že prirejeni za prenos povratne smeri. Če torej želimo imeti prenos povratne informacije (slika, podatki, itd), vgradimo novejše ojačevalnike, ki že imajo premostitvene kretnice za povratno smer prenosa.

JUS in CATV

Pri nas je standardiziran sub-split sistem. Standard JUS N.N6.171 predpisuje razpored frekvenčnih pasov in podaja zahteve za sisteme, ki delujejo v okviru CATV (JUS N.N6.172). Standard predvideva uporabo področij 5-30MHz za povratno in 30MHz-1GHz za osnovno smer prenosa. V povratni smeri so na voljo trije kanali za prenos slike (P1,P2,P3) in zasedajo pas 9-30MHz. Nezasedeno je še področje med 5 in 9MHz. V osnovni smeri razpolagamo z naslednjimi kanali (sledijo si v smislu naraščajoče frekvence): K2-K4, S1-S10, K5-K12 in S11-S38. Ti kanali pokrijejo spekter 47- 444MHz [3,4].

Mid- in high-split sistema pri nas nista standardizirana. Ob dejstvu, da smo pri nas omejeni na sub-split sistem, bi bilo dvosmerno komunikacijo možno izvesti ob uporabi enega TV-kanala v povratni in enega v normalni smeri. Poleg povratnih TV-kanalov, imamo v povratnem frekvenčnem pasu še nekaj prostora, vendar pa nekateri proizvajalci uporabljajo nižje frekvence za odzivne signale pri nadzoru delovanja mreže in za prenos določenih drugih podatkov. Zato se nižjih območij raje izogibamo. Tako ohranimo popolno kompatibilnost tudi pri morebitni nadgradnji sistema.

2.3 Tendence razvoja

CATV se razvija na dveh področjih. Prvo področje predstavlja topologija sistemov. Da bi lahko izboljšali kvaliteto signala, mora biti pot do naročnika čim krajša (manjše število ojačevalnih mest) in čim bolj direktna (t.j. manjša razvejitev signala). Zelo ugoden je t.i. način 'zvezdišč', ki ga uporabljajo nekateri proizvajalci (npr. Fuba). Po tem sistemu lahko iz glavne razdelilne postaje izhaja več primarnih vodov (A-vodi), ki se v primarnih ojačevalnih mestih delijo na sekundarne (B) vode. Število ojačevalnikov v sekundarnih vodih je majhno. B-vodi se ne vejijo pretirano. Iz vsake sekundarne ojačevalne točke lahko poteka do osem končnih (C) vodov. Slednji vodijo do končnih pasivnih razvejišč (zvezdišč). Na razvejišča, v katerih se sistem zaključuje, so priključeni naročniki.

Tak sistem je mnogo bolj ekonomičen za realizacijo dvostranskega prenosa podatkov. Število prehodov preko ojačevalnih mest je manjše. Prav tako je sorazmerno kratka razdalja od naročnika do glavnega (primarnega) voda, od koder se podatki prenašajo direktno do glavne postaje.

Druga smer razvoja poteka na frekvenčnem področju prenosa. Ker je pritisk velikega števila TV-programov na kabelske sisteme vedno večji, so upravljalci CATV-omrežij prisiljeni večati število razpoložljivih TV kanalov. Frekvenčno območje morajo zato širiti še v področje proti 860MHz. Zaradi velikih izgub pri prenosu tako visokih frekvenc, so potrebni kvalitetnejši kabli in večje število ojačevalnih mest z nizkošumnimi ojačevalniki. Ovir za uvajanje mid- in high-split sistemov ne bo več. Ta dva sistema bosta brez večjih težav omogočila poleg prenosa TV signalov tudi realizacijo ene, ali celo več neodvisnih računalniških mrež. Na račun te pridobitve se bo potrebno odpovedati trem oz. dvanajstim TV-kanalom ter radijskem spektru.

3. Računalniške mreže

Računalniških mrež je več vrst; od preprostih lokalnih, ki povezujejo nekaj računalnikov, do kompleksnejših WAN (Wide Area Networks), ki povezujejo tudi računalnike po celem svetu. Doslej se je uveljavilo več standardov za mreže, od katerih ima vsak svoje dobre in slabe lastnosti.

3.1 Osnovni standardi

Preglejmo osnovne značilnosti mrež, ki jih opisujejo IEEE standardi za LAN:

CSMA/CD (IEEE 802.3)

Carrier Sense Multiple Access / Collision Detection je standard, kateremu ustreza splošno znana Ethernet mreža. Načelna topologija take mreže je, da so vsi uporabniki priključeni na vodilo, na katerem so v vseh pogledih enakopravni. Nihče nima prioritete. Ko nekdo potrebuje vodilo, začne oddajati. Vmesnik med uporabnikom in vodilom skrbi, da se uporabnik ne more vključiti na vodilo (začeti oddajati), če na vodilu že poteka neka druga komunikacija. Med oddajo vmesnik nenehno primerja sprejete znake z oddanimi. Če sta oddani in (skoraj) istočasno sprejeti znak enaka, pomeni da poteka oddaja normalno. V primeru pa, da se razlikujeta, je to znak, da sta začela istočasno oddajati dva uporabnika. V takšnem primeru pride do t.im. kolizije. Vmesnik, ki zazna kolizijo, pošlje na linijo kratek opozorilni signal (Jamming Signal) in sproži naključni časovni generator. Po pretečenem naključnem času vmesnik ponovno poskuša vzpostaviti zvezo. Kolizija je pravilno zaznana, če je zaznana, še predno je oddan celoten paket podatkov. Na širokopasovnem sistemu traja ta čas $T_0 = 4\tau$, pri čemer je τ čas, ki ga signal potrebuje za prelet celotne linije. Signal premaga razdaljo med dvema vmesnikoma v najslabšem primeru času T_0. Zato mora biti tudi minimalna dolžina podatkovnega paketa enaka T_0. Ta zahteva omejuje dolžino kabla in s tem uporabo takšne mreže v okviru CATV. Ker dolžine razvoda ni mogoče prosto izbirati, je uporaba možna le ob zmanjšani hitrosti prenosa vsaj za faktor 10, ali ob omejitvi mreže na en sam krak CATV-omrežja.

Token Bus (IEEE 802.4)

Pri tej vrsti mreže gre za princip podajanja žetona. Uporabniki, ki so priključeni na skupno vodilo, v logičnem smislu predstavljajo zaključen komunikacijski prostor, po katerem si podajajo žeton. Fizična izvedba vodila ni zaključena.

Postaja (uporabnik), ki trenutno ima žeton, ima za nek vnaprej določen čas pravico dostopa do medija. V tem času sme oddati enega ali več podatkovnih paketov, prav tako pa lahko kliče druge postaje (uporabnike) in sprejema njihove odgovore. Ko se čas izteče, ali ko vozlišče zaključi komunikacijo, pošlje postaja žeton naslednjemu logičnemu uporabniku.

Servisne funkcije takšne mreže so bolj bogate kot funkcije Ethernet mreže (CSMA/CD) in obsegajo:

- inicializacijo komunikacijskega prstana (dodelitev logičnih naslovov ob startu ali ob porušitvi sistema)
- dodajanje novih postaj
- izključevanje obstoječih postaj (neprostovoljno ali prostovoljno)
- korekcija napak (če začneta dve postaji oddajati istočasno, ali če se naslovljena postaja ne oglasi)

'Token Bus' omogoča takojšnjo vključitev v CATV-omrežje, ker dolžina prenosnega kabla ne vpliva na kvaliteto komunikacije.

Token Ring (IEEE 802.5)

Medtem, ko Token Bus predstavlja komunikacijski prstan le v logičnem smislu, ga Token Ring predstavlja tudi fizično. Podatki se prenašajo v krogu od enega uporabnika do drugega, dokler se ne vrnejo do tistega, ki jih je odposlal. Tudi tu ima pravico dostopa do medija tisti uporabnik, ki ima žeton. Le ta sme oddati novo sporočilo, ki vsebuje naslov prejemnika. Vsak uporabnik odda sporočilo svojemu nasledniku. Naslednik ga sprejme; če je sporočilo zanj, si naredi kopijo in ga nato odda naprej. Če sporočilo ni zanj, ga samo odda naslednjemu. Postopek se v krogu ponavlja pri vsakem uporabniku. Ko sporočilo prepotuje cel krog, ga pošiljatelj umakne iz obtoka.

Čeprav je ta topologija krožna, največkrat izgleda kot zvezda. V središču zvezde se nahaja poseben koncentrator, na katerega so priključeni uporabniki. Koncentrator je zgrajen tako, da uporabniki tvorijo krog. Zaradi krožne topologije ta standard ni primeren za direktno vključitev v CATV.

Metropolitan Area Network (IEEE 802.6)

V primerjavi z že opisanimi mrežami, je Metropolitan Area Network (MAN) mreža širših razsežnosti. Predvidena je za prenos podatkov, zvočnih in video signalov na večje razdalje (med 5 in 50km), kar je že izven dosega LAN. Standard je še v nastajanju, osnovna ideja pa je razširljiva mreža, ki naj bi povezovala večje mesto (od tu izraz Metropolitan). Poglavitni namen takega sistema je zagotovitev hitrega prenosa podatkov in informacij na večje razdalje. Mreža naj bi uorabljala že obstoječe medije (CATV, optične in telefonske kable) [1].

3.2 Širokopasovne LAN

Veliko število mrež za prenosni medij uporablja koaksialni kabel. Če se po njemu prenašajo čisti digitalni signali, poteka prenos v osnovnem pasu (baseband), če pa se s signali modulira neka višja frekvenca, poteka prenos v širokopasovnem pasu (broadband).

Pri širokopasovnem prenosu se sprejemna in oddajna frekvenca med seboj razlikujeta. V primeru usmerjenega prenosa kot je v CATV-sistemu, sta frekvenčna

pasova ločena glede na smer prenosa. Vsi udeleženci imajo eno skupno sprejemno in eno skupno oddajno frekvenco. Oddani signali potujejo od poljubnega uporabnika do repetitorja (Head End Translator), ki jih sprejme, pretvori na oddajno frekvenco (ki je višja) in jih ponovno odda. Za realizacijo mreže v tem smislu so primerne mreže, ki so grajene po standardih IEEE 802.3 in 802.4.

Širokopasovni medij dovoljuje, da je mogoče imeti na enem mediju istočasno več neodvisnih in tudi različnih mrež, možno pa je sočasno prenašati še video in audio signale. Kot primer takšne mreže naj navedemo mrežo, ki je instalirana na North State Texas University (NTSU) [1]. Na njej delujejo poleg šestih dvosmernih TV-kanalov še tri neodvisne računalniške mreže. Število stavb, ki jih sistem povezuje, pa se je do nedavnega povzpelo od osem na 60 z okoli 7000 priključki.

Standard IEEE 802.3

Pri širokopasovni Ethernet mreži (Standard IEEE 802.3) je standardni razpon med oddajno in sprejemno frekvenco 156,25MHz (stari sistemi), oz. 192,25MHz (novejši sistemi). Standardni frekvenčni pasovi so:

Oddaja (MHz)	Sprejem zamik 156.25MHz (MHz)	Sprejem zamik 192.25MHz (MHz)
35,75-53,75	192-210	228-246
41,75-59,75	198-216	234-252
47,75-65,75	204-222	240-258
53,75-71,75	210-228	246-264
59,75-77,75	216-234	252-270
65,75-83,75	222-240	258-276

Omenjene pasove je možno vključiti direktno v mid- oz. high-split sisteme. Po podatkih iz ameriške literature, kjer so takšni sistemi že v uporabi, potrebujemo za prenos podatkov pri polni hitrosti (10Mb/s) v vsaki smeri tri TV-kanale, kar pomeni $3 \times 6 = 18$MHz (V ZDA je širina TV-kanalov 6MHz). Ker je to sorazmerno veliko - skupaj 36MHz za predajo in sprejem - ponujajo nekateri proizvajalci mreže, ki niso povsem standardne, zasedajo pa v obeh smereh skupaj le 12 ali 24MHz (na račun manjše hitrosti prenosa ali drugačnega kodiranja/moduliranja signala). Takšne izvedbe niso povsem kompatibilne s standardnimi komponentami za Ethernet.

Standard IEEE 802.4

Za razliko od standarda IEEE 802.3, je bil standard IEEE 802.4 že od samega začetka predviden za širokopasovni prenos. Prenosni medij je standardni 75-Ohmski TV koaksialni kabel. Po tem standardu je možno podatke modulirati na tri načine, od katerih za CATV sisteme ustreza le AM/PSK modulacija.

Odvisnost potrebne širine frekvenčnega pasu od hitrosti oddaje je naslednja:

Hitrost(Mb/s)	Širina(MHz)
1	1,5
5	6
10	12

Standardni frekvenčni zamik oddaja/sprejem znaša 192,75MHz in direktno ustreza mid- in high-split sistemom. Standard sam ne določa frekvenc prenosa. Za ilustracijo si oglejmo dane možnosti frekvenc za ameriške standarde:

Oddaja(MHz)	Sprejem(MHz)
23,75	216
29,75	222
35,75	228
41,75	234
47,75	240
53,75	246
59,75	252
65,75	258
71,75	264
77,75	270
83,75	276
89,75	282
95,75	288
101,75	294

3.3 Vključitev LAN v CATV

Vidimo, da je možno 'Ethernet' in 'Token Bus' brez večjih težav priključiti na CATV-omrežje. Vprašanja, ki se pri tem le porajajo, so širina frekvenčnega pasu, ki ga mreža zahteva, in razvejanost CATV-omrežja. Oba sta pogojena od zunaj - z izvedbo in velikostjo CATV-sistema.

Pri odgovoru nimamo zavezanih rok. Za realizacijo mrež bi lahko uporabili posamezne krake CATV-omrežja. S tem bi zmanjšali čas potovanja signalov (zaradi manjše dolžine) in povečali prepustnost (manjše število uporabnikov). Na drugi strani obstaja možnost, da uporabimo CATV kot most za povezavo večjega števila LAN in s tem pokrijemo dokaj široko področje. Ker poteka večina komunikacij znotraj posameznih LAN, t.j. na kratke razdalje, je ta zadnja možnost še posebno zanimiva.

Iz povedanega lahko zaključimo, da pri nas v splošnem ni zadržkov za vključitev računalniških mrež v CATV-sisteme. Težavo predstavlja le razdelitev frekvenčnih pasov za smer naprej in nazaj. V smeri nazaj (t.j. v povratni smeri) je namreč na voljo le sorazmerno ozek frekvenčni pas. Problem ni nerešljiv. Rešitev obstaja v zmanjšanju hitrosti prenosa ali v izvedbi učinkovitejšega kodiranja. Tako eno kot drugo pomeni manjšo hitrost

prenosa in tudi možno nezdružljivost s tržno dosegljivimi sistemi.

Možna hitrost prenosa je pri uporabi CATV večja, kot pri uporabi telefonskih linij. Če je v nekem naselju razpredena CATV,-potem povezuje veliko uporabnikov; pri malem številu uporabnikov bi bilo težko pokriti stroške izgradnje CATV. Skoraj zanesljivo je tudi, da bo tisti, naj bo to posameznik ali firma, ki zmore nakup računalniške opreme, zmogel pokriti tudi stroške CATV-priključka.

Pri investiciji v CATV-mrežo, ki povezuje celo naselje, je smiselno, da CATV uporabimo tudi za medsebojno povezovanje računalnikov. Pri tem naletimo na dejstvo, da niso dosegljivi modemi, ki bi delovali v področju sub-split sistemov. Ovira ni nepremagljiva.

4. Zaključek

Na področju, kjer se razpreda CATV, obstajajo tudi potrebe za računalniške komunikacije. CATV-omrežje se lahko pokaže kot idealen prenosni medij, če želimo medsebojno povezati uporabnike v mestu ali v večjem naselju. V ta namen so uporabni tudi TV razvodi v stavbah ali zaključeni sistemi za komplekse stavb. Ugodna je uporaba CATV-mreže tudi, če potrebujemo računalniško mrežo za avtomatizacijo stavb (signalizacija, nadzor, ...). V to kategorijo sodijo tudi poslovni in hotelski sistemi [5]. Zaključena TV mreža znotraj poslovnega, hotelskega ali podobnega kompleksa lahko pokriva poleg prenosa TV-programov tudi lokalne potrebe po računalniških (ena ali več LAN) in govornih komunikacijah (lokalno telefonsko omrežje z možnostjo priključitve na že obstoječe javne linije). Preko priključka na CATV je mogoče povezati LAN v okviru stavbe ali kompleksa stavb z mrežami na širšem področju, t.j. povsod tam, kjer deluje CATV.

Obstaja več IEEE standardov za mreže. Od njih sta za vključitev v CATV primerna IEEE 802.3 CSMA/CD (Ethernet) in IEEE 802.4 Token Bus. Zaradi narave CSMA/CD-mreže je ugodno, če razsežnosti takšne mreže niso prevelike. CSMA/CD je na splošno primeren za veliko število manj zahtevnih, bolj naključnih uporabnikov. 'Token Bus' pa na drugi strani najbolj racionalno deluje z manjšim do srednjim številom intenzivnih uporabnikov. Večje razdalje med uporabniki ne motijo delovanje mreže 'Token Bus'.

Prenos v obe smeri je pri CATV, ki je sicer enosmeren prenosni medij, izveden z delitvijo frekvenčnega pasu. En del frekvenčnega področja je namenjen prenosu v smeri naprej do uporabnikov, drugi, ožji pas, pa od uporabikov do glavne razvodne postaje. Sistemi, ki obratujejo pri nas še ne predvidevajo dvostranskih prenosov informacij. Obstaja pa možnost nadgradnje v tej smeri. Ker so sistemi običajno komercialno usmerjeni, ostaja za uporabo v povratni smeri le manjši del frekvenčnega območja. Tendence v svetu kažejo na povečanje števila prenašanih kanalov. Zgornja meja prenosnega območja se širi proti višji frekvenčni meji. To bo omogočilo povečanje števila obojestranskih kanalov in razširilo možnosti računalniških komunikacij v okviru sistema CATV.

Literatura

[1] Thomas W. Madron, Local Area Networks - The Second Generation, John Wiley & Sons, Inc. , 1988

[2] BK 450MHz - Verstärkerpunkt, Fuba, Nr.: 826 044, 12/1988

[3] JUS N.N6.171, 1986

[4] JUS N.N6.172, 1989

[5] P.Kolbezen, P.Zaveršek, Transputerji v sistemih za delo v realnem času, Informatica 3/90 (1990), pp. 35-43

[6] Kabelski razdelilni sistemi, Elrad, April 1990

[7] Priročnik za projektiranje kabelskih razdelilnih sistemov, Elrad

THE SYSTEM OF KNOWLEDGE

INFORMATICA 4/90

Oliver B. Popov
Institute for Informatics
Faculty of Natural Sciences and Mathematics
C.M. University of Skopje
Gazi baba b.b., 91000 Skopje

Keywords: machine cognition, intensional reasoning, knowledge carriers, concepts, complexes, epistemic reasoning.

As ambitions and demands increase in Artificial Inteligence, the need for formal systems that facilitate the study and the simulation of machine cognition has become inevitability. These series of papers explores and developes the foundation for a formal system for propositional reasoning about knowledge. The semantics of every meaningful expression in the system is fully determined by its intension, the set of complexes in which the expression is confirmed. The knowledge system is based on three zero-order theories of epsitemic reasoning for consciousness, knowledge and entailed knowledge.

So porastot na ambiciite i potrebite vo Veštačkata intelegencija, kako neminovnost se nametnuva izučuvanjeto na formalnite sistemi koišto ovozmožuvaat simulacija na mašinskoto osoznavanje. Dadenata serija od statii gi istražuva i razviva osnovite na eden formalen sistem na izjavno rasuduvanje za znaenjeto. Semantikata na sekoj izraz so smisol vo sistemot vo potpolnost e opredelena so negovata intenzija, množestvoto na kompleksi vo koišto izrazot se potvrduva. Sistemot na znaenje se zasnovuva na tri teorii od nulti red na soznajnoto rasuduvanje za svesta, znaenjeto i impliciranoto znaenje.

THE ONTOLOGY OF KNOWLEDGE

"The totality of meaning is never fully rendered ..." Merlay-Ponty

1. Extension and Intension

Since the publication of Frege's work (1), the functionality principle according to which the meaning of an expression is a function of the meanings of its constituents has generally been accepted in linguistics. However, due to the ambiguities present in the context of natural languages Frege recognized the need for a distinction between an extension or denotation of an expression ϕ in some language L and its intension or sense. To reduce the number of possible constituents let us use the term 'entity' to represent things, items, objects or individuals. The few examples bellow illustrate why the distinction between an intension and an extension is necessary.

For instance, take a particular word such as 'book'. The intension of 'book' is a definition, such as " a lengthy treatment of a subject, printed and binded, and comes on the market as comodity". The extension of 'book' is the set of all books in the world. Simpliciter, if the word 'book' denotes a concept of a book then the intension is all entities that define the concept, while the extension is all those entities that 'fall' under the concept.

Certain authors consider the distinction between an intension and extension in the semantics of natural languages to be analogous between an episodic and a semantic memory (2).The episodic memory holds particular facts which correspond to extensions, while the semantic memory holds general principles which are intensions. For Umberto Eco, the intension of an exprssion corresponds to the theory of signification and the extension of an expression corresponds to the theory of communications. Our interpretation is more of a traditional one, the intension of an expression is a function over the set of possible worlds.

Assume na arbitrary language L and given an expresion ϕ in L, denote the extension of ϕ with EXT(ϕ) and its intension with INT(ϕ). When the expression ϕ is a sentence, then the EXT(ϕ) is the set 2.

Does the definition of' extension of the expression presented above help in any way to extract the meaning that each expression carries within itself? Consider the sentences ϕ = "Two plus two are four" and ψ = "John and Ann are step-siblings". The principle of functionality is satisfied. If both sentences are true (false) then the sentences have the same extension. But ϕ and ψ have entirely different meanings and thus represent different informational sources for a reasoning agent.

The distinction between an extension and an intension is even more apparent in the examples where the principle of functionality fails, such as oblique or intensional contexts. The famous example is the 'morning star-evening star' paradox. Using the modal operator 'necessity', the sentence "Necessarily the morning star is identical with the morning star" is true with respect to all possible worlds. So, if one is to follow the functionality principle and replace the second occurrence of 'morning star' with 'evening star' the resulting sentence "Necessarily the morning star is equal to the evening star" is true. On the contrary, the second sentence is false. Why? It is quite possible that there is a world in which the stars are not identical, although both constituents "morning star" and "evening star" have the same extension.

The former example could also be examined in an epistemic context. The following sentence "If the morning star = the evening star, then Ann knows that the morning star appears in the morning if and only if Ann knows that the evening star appears in the morning" is not generally considered to be logical truth. By logical truth of a sentence ϕ in a language L one understands that ϕ is true under all possible interpretations of L. But this is consistent with our undersstanding of epistemic notions. In order for ϕ to be a logical truth, ϕ should have an additional premise "Ann knows that the morning star = evening star". Indeed,the additional premise can be regarded as a deviation from the standard logic. However, the deviation is acceptable as long as one tries to capture different epistemic and doxastic inclinations.

One possible solution to the problem is to avoid possible world semantics and modalities. It is simple enough and wrong. Assume that Mr. Jernej is a student in the Computer Science Faculty at the University of Ljubljana. While pursing his degree he supported himself by working at nights at the University library, where one of his coworkers used to be Mrs.Stefani. After the graduation, Mr. Jernej is hired by Institute "Jozef Štefan". Since Mr. Jernej currently works for IJS, the expression "coworker of Jernej" is coextensive with the expression "memeber of IJS". Now consider the expression "former coworker of Mr. Jernej" which is coextensive with Mrs. Stefani. Replacing the expression "coworker of Mr. Jernej" with the "member of IJS" yields the expression "former member of IJS" which is not coextensive with Mrs. Stefani. The principle has failed again without mentioning any possible worlds. Tense modalities are vivid examples for demonstrating an intensional context (3).

Clearly, the extension of an expression must somehow depend on the syntactical context in which it occurs. An expresson ϕ used in an ordinary context has an extension EXT(ϕ) while used in a referentially oblique context its extension becomes INT(ϕ). To preserve the principle of functionality one might introduce for each expression ϕ a new one denoted by ϕ' or the concept of ϕ, such that the extension of ϕ' is the intension of ϕ. As a consequence the decomposition of any expression could be done in terms of both extensions and intensions (3), which depends on the nature of the context in which each individual construct appears.

Since none of the terms concerning the context of the use an expression in some language L, such as possible state of affairs, possible worlds, points of reference or 'indexes'. satisfies our intiution, a new term 'complex' is introduced which supplants all aforementioned terms with a similar meaning. This is also consistent with Hintikka's quotation of Savage's sugestion in (4), that it is better to think in terms of some 'small worlds'.

Complex is simply a set of propositions whichg are cognitively accessible to a reasoning agent and sets of complexes which stand for those propositions. The non-empty denumerable set of complexes shall be denoted by K. Thus, given a sentence ϕ its intension, INT(ϕ) is exactly the proposition expressed by the sentence ϕ. The idea is to regard the object of knowledge to be a proposition. Hence, knowledge is an empirical relation between an entity (reasoning agent) and a proposition.

Is there a need for the notion of complexes in the treatment of knowledge? Assume that there are two physically identical complexes, yet what different agents know with respect to those complexes is not necessarily identical.

The position taken here is a conceptualist one, that is complexes are not only necessary as primitives for analysis of epistemic and doxastic notions,but they are indispensible if one is to capture what is vacuously called a multitude of conceptions or imagination. Both complexes and intension are possibly the initial steps toward a comprehensive theory of meaning and understanding.

2. Knowledge-carriers

The condition imposed on the propositions that constitute the epistemic enviroment of an agent is that they be true. The question is whether propositions are the enities capable of truth and falsity, and in general what kind of entities are likely candidates to be 'truth bearers'? The choice is limited to sentences and propositions.

Sentences are examined first. A sentence is any complete and correct string of expressions in a given language according to its gramatical rules. It is common to distinguish between sentence types and sentence tokens. Thus, by writing "The sun is hot", "The sun is hot", one might say that there is one sentence type inscribed twice or that there are two sentence tokens.

This provides the first argument for the rejection of sentences as possible candidates for 'truth-bearers',since it is not clear to which entity (type or token) one should attribute the truth values. The second argument against their candidacy is the dependence of sentences on the underlying language and physical existance. The analogy with the numerals and numbers is quite illustritative.

The sentences can be inscribed and erased, and thus cease to exist. The propositions expressed by those sentences will still be present and true. The same propositon can be expressed by different sentences. This extra-linguistic character makes the propositions suitable for knowledge-carriers,a fortiori truth bearers since knowledge has also extra-systematic significance with respect to any particular language being used to express it.

Each proposition is identified with a set of possible complexes in which the proposition is true. In other words a proposition is a function from the set of possible complexes to the set 2. Hence, a plausible context of a reasoning agent is a set of propositions in which every proposition is the set of complexes in which the proposition is 'true'.

However, the identification of the object of knowledge as a set of possible complexes is not without problems. Consider the following two propositions:

$\Pi 1$: "Two plus two are four."

$\Pi 2$: "The earth is round."

Denote with $\phi 1$ and $\phi 2$ the sentences expressing the two propositions $\Pi 1$ and $\Pi 2$ respectively. If the sentence $\phi = \phi 1 \Leftrightarrow \phi 2$ is logically true then the sentences $\phi 1$ and $\phi 2$ are logically equivalent. This entails that the propositions $\Pi 1$ and $\Pi 2$ are logically equivalent which implies that they are true in the same set of possible worlds. Why is it so? The relation between propositions and complexes equates the notions of logical equivalence and semantic equivalence. Thus it is not possible to distinguish between a propositional equivalence and a propositional identity,a phenomenon which is a consequence of set theory.

But the propositions $\Pi 1$ and $\Pi 2$ are far from being identical. Although both true, they have entirely different contexts of information and convey different meanings. To better understand the difficult problem of propositional identity it is helpful to explore the ontological status of the propositions.

3. Concepts

To assert that entity has an ontological status is to recognize the existance of its intrinsic structure. The structure then can be contrasted with other entities of the same or different types. An example of the later was the analysis of sentences and propositions. So the focus here is on entities of the same type, i. e. propositions.

The existance of an inner structure entails two different characteritsics of an entity with reference to the complexity of its structure. When the structure of an entity is reducible to more primitive entities than the original entity we have a case of a compound entity or a molecular one. On the other hand, when the original structure is irreducuble to more primitive entities then one speaks about atomic entities.

For propositions, the argument put forward is the following: propositions are compound entities and their essential constituents are concepts. Regarding the concepts as primitive constructs of propositions does not imply that concepts themselves do not posses inner complexity on their own. A concept may or may not be the subject of further logical analysis. Certain concepts may also entail the existance of other concepts which fall under the original concepts.

Chronologically, the notion of concept is among the primal entities in the vocabulary of reasoning (5). Concepts have always been suitable as a reference, but quite elusive for a rigorius definition. McCarthy in (6) openly admits of using the term 'concept' in the study of first-order theories without any attempt to discuss it. However, the acceptance of concepts as essential entities does not prevent us from giving at least some explanatory account of their existance.

Certain authors (5,1) think that it is important to disassociate concepts from conceptions and ideas. They apply to concepts the same type of 'abstract detachment' as they do to propositions. It is true that conceptions and ideas could basically be treated as psychological entities which mirror some internal 'mental state' of an agent. But to deny them any role in the formulation of concepts amounts to the denial of possibility in the process of reasoning. What is really important to realize is that concepts have the same kind of extra-linguistic property as propositions do.

Just what kind of entities are used to express concepts? The general agreement is that there are two (5,7). Expressions of reference, which can be undetermined such as "something" or "everyone", or determined such as "morning star" or "Peter". Also, propositional functions (or open sentences) which contain a locus which is later substituted with a referring expression. Consider the following proposition:

Π : "The color of blood is red."

In this case the referring expression is "the color of blood" and the propositional function is "... is red". The set of concepts that appear in the proposition Π is <being a color,being a blood, being red>. No truth assignment could be given to propositional functions. A distinction, therefore, should be made between open sentences which are concept-expressing entities and closed sentences used to express propositions and are subject to an assignment of truth-value.

As Quine (7) pointed out, any closed sentence could be transformed to an open sentence. For the proposition Π the transformation yields:

$T\Pi$: "x is the color of blood and
x is red."

Identifying propositions with a set of complexes in which the propositions were true has a two-fold implication with respect to concepts. First, concepts are related to the complexes through propositions of which they are essential constituents. Second, in view of the fact that propositions were knowledge-carriers, concepts must be considered as constituents of knowledge. This implies that in a way concepts are the primary building blocks of knowledge.

With reference to the inner complexity of concepts a distinction is made between linear and non-linear concepts. Simpliciter, a concepts is linear if it is not subject to logical analysis; a concepts is non-linear if it is. By a logical analysis understand the process of reduction of a concept to another concept which is a synonym or an epistemic alternative of the original concept.

A necessary or analytic truth is one that holds in all complexes. For example, consider the concepts 'mother' and 'green'. The claim is that the concept of a mother is a non-linear one and the concept of green is a linear one. A possible logical analysis of the concept of mother is a

'female-parent', but no parallel analysis is applicable to the concept of being green. Someone may argue that analysis of 'green' could produce another concepts like 'wavelenght','reflection' and 'absorption'. These are synthetic or contingent truths because these truths are dependent on the existance of 'greeness' in the resonant complex. A resonant complex is a complex in which the logical and physical truth are coextensive. It is plausible to accept that the resonant complex represents reality.

The question of synonymity is very important and extremely difficult. It is apparent that intensions are not sufficient for a satisfactory treatment of synonymity. Therefore, in this case no distinction is going to be made between identical and synonym propositions.

Other than synonymity, there are many modal-like relations that could be defined among concepts. Thus given two concepts $\Delta 1$ and $\Delta 2$ the relations of agreeability, disagreeability, universal agreeability, relevance, and universal relevance can be defined.

Concept $\Delta 1$ is agreeable with concept $\Delta 2$ if and oly if there is at least one complex in which both concepts $\Delta 1$ and $\Delta 2$ apply to the same entity. For example, the concept of being a professor and him knowing the subject that he is teaching.

Concepts $\Delta 1$ and $\Delta 2$ are disagrreable if and only if there is no complex in which both concepts apply to the same entity. The concept of bachelor and the concept of being married is an example of disagreeable concepts.

Concepts $\Delta 1$ and $\Delta 2$ are universally agreeable if and only if there is no complex in which the application of $\Delta 1$ to an entity does not entail the application of $\Delta 2$ to the same entity. The propositions

$\Pi 1$: "John and Ann have one parent in common."

$\Pi 2$: "John and Ann are step-siblings"

contain respectively the concepts of 'having one parent in common' and 'being step-siblings'. One can observe that if the first concept applies to some entities (for example John and Ann) the second applies to the same entities by necessity.

A concept Δ is relevant if and only if there is an entity in at least one complex that falls under the concept Δ. The example of e relevant concepts is the concept of being a prime number.

A concept Δ is universally relevant if there is at least one entity in each possible complex that falls under the concept Δ. The concept of self-identity is the example in this case, assuming of course that each complex contains at least one entity.

A concept is actually a selector. It selects entities from the universe of all posssible entities and those entities represent its extension. If the concept is empty then the extension is the empty set. An example of an empty concept is the concept of "a round square", since it is logically impossible.

The argument put forward in (5) is that the decomposition of a proposition to concepts is order-sensitive. For instance, take the proposition

Π: "John was on the top of Jim."

and assume that the proposition Π is true. Some of the possible combinations for the concepts that appear in the proposition Π are represented by the following sets ⟨being John, being on the top, being Jim⟩, ⟨being Jim, being on the top, being John⟩, and ⟨being John, being Jim, being on the top⟩. The first set of concepts reflects the proposition Π, while the second set of concepts yields another proposition:

Σ: "Jim was on the top of John."

The third set of concepts is not determined at all with respect to the original meaning of the proposition Π.

An analogy underlining the distinction between ordered and unordered sets of conceptscan be found in geography. An agent is given an assignment to visit some cities. To know the ordering is like having a complete map of the tour with all the cities and the roads that interconnect. Here concepts represent cities. To have concepts without an ordering amounts to being a conscious only that cities exist on the map. It may be even the case that a road between some cities does not exist. What is missing is the map (or the appropriate mappings between concepts).

To conclude, a proposition may be defined as an ordered sequence of concepts subject to a truth-value assignment.

The prima facie denial of the attribution of truth values to concepts should not be understood as too categorical. As we shall see in the next article, the role of the concepts as the building blocks of knowledge must essentially be recognized through some formal system that can deal with them.

BIBLIOGRAPHY

(1) Frege, G. "Uber Sinn und Bedeutung", Zeitschrift fur Philosphie and Philosophische Kritik , No. 100, (1892),pp. 25-50

(2) Sowa, J. "Conceptual Structures: Information Processing in Mind and the Machine, Addison-Wesley, 1983.

(3) Gallin, D."Intensional and Higher-order Modal Logics", American Elsevier, New York, 1975.

(4) Hintikka, J. "The Paradigm of Epistemic Logic in Reasoning about Knowledge, ed. by Halpern, J.,(1986), Morgan Kaufmann,pp. 63-81

(5) Bradley,R.,Swartz,N. "Possible worlds", Hackett Publishing Company,1979.

(6) McCarthy, J. "First-order Theories of Individual Concepts and Propositions" in Machine Intelligence ed. by Michie, D., Ellis Horwood,(1979),pp.120-147.

(7) Quine,W.V. "Mathematical Logic", Harper and Row,1940.

NOVICE IN ZANIMIVOSTI

NEWS

The European Journal of Information Systems (EJIS)

The EJIS is providing a distinctive European perspective on the theory and practice of information systems, drawing on European traditions of research and application. The journal contains a lively mixture of research articles and case studies of general interest to the growing number of informations systems professionals in academia, industry, commerce and government. By emphasizing the exchange of ideas between academics and practitioners, EJIS aims to keep everyone informed of developments in this broad, fast moving field, and also to provide a critical view of technology, management and policy issues.

Papers submitted for publication should be original and interesting, such that they provide a contribution to the recorded knowledge within the field of information systems. This field is taken to include the technical, economic, organizational and social aspects of the development, use and management of computer-based information systems. Because information systems, as a field of study, is relatively young and its boundaries have yet to stabilize, advice can be obtained from the editors as to whether papers on particular topics are likely to be appropriate. English reprints of articles previously published in another language are considered, where they are of interest to the wider information systems community.

EJIS welcomes both theoretical and practical papers: in many cases, the best papers contain an amalgam of theory and practice. Papers may propose new theories, or extend or criticize existing theories within information systems, including those dealing with the methodology of information systems research. Papers of a tutorial nature, or papers that largely review existing literature are acceptable where they make a definite contribution. Papers based on empirical research, such as questionnaire survey, controlled experiments or more qualitative case studies, and practical papers that describe particular problems and provide an insight into current practice are welcomed.

EJIS is published by Macmillan Press Ltd, in association with the Operational Research Society. The journal reaches not only scholars and libraries, but also over 3,000 practitioners throughout Europe. It is edited at the London School of Economics and guided by an active editorial board of leaders in the field.

Articles for submission should be sent to the Joint Editors: Jonathan Liebenau and Steve Smithson, Department of Information Systems, London School of Economics, Houghton Street, London WC2A 2AE, Fax: 071 242 0392. New publications should be sent to the Book Review Editor, Tony Cornford, at the same address. The Editorial Board includes editors from Australia, Greece, UK, USA, USSR and Yugoslavia (Vlatko Cerić, Department of Economics, University in Zagreb).

Instructions to authors are in short the following: papers should be written in Oxford English, up to 6000 words, with a title page bearing the title, names and addresses of editors and the author, biography of no more than 100 words and a summary of no more than 200 words, etc. Manuscripts can be submitted on disk wherever possible.

EJIS is published bimonthly. Individual subscription outside EEC is £50, institutional £99. Payment may be made in any currency to ***Richard Gedye, Macmillan Press Ltd., Houndmills, Basingstoke, RG21 2XS, UK. Fax 0256 810526.***

A.P. Železnikar

Cybernetica

A Quarterly Review of the International Association for Cybernetics

IAC is publishing and distributing (mainly to universities and institutions) the Journal Cybernetica (500 copies) and IAC Newsletter (700 copies) to the following countries (51): Algeria, Argentina, Australia, Austria, Belgium, Brazil, Bulgaria, Canada, Chile, China, Cuba, Czechoslovakia, Denmark, Federal Republic of Germany, Finland, France, German Democratic Republic, Greece, Great Britain, Luxembourg, Hong Kong, Hungary, India, Israel, Italy, Japan, Libya, Malaysia, Morocco, Mexico, Netherlands, New Zaeland, Nigeria, Northern Ireland, Norway, Poland, Portugal, Rumania, Saudi Arabia, Singapore, South Africa, Spain, Sweden, Switzerland, Taiwan, Turkey, Uruguay, USA, USSR, Venezuela, and Yugoslavia.

The annual subscription for Cybernetica is 2000 BF for individuals and 4000 BF for institutions. The address is: ***International Association for Cybernetics, Palais André Rijckmans, B.5000 Namur, Belgium.***

A.P. Železnikar

Avtorsko stvarno kazalo časopisa Informatica, letnik 14 (1990)

Authors Subject Index of the Journal Informatica, Volume 14 (1990)

Članki — Articles

(1990), No. 3, 31-34.

Rihar, M., Nekaj praktičnih vidikov uporabe Case orodij, Informatica 14 (1990), No. 1, 83-89.

Šurla, D. and Z. Budimac, Detection of the Intersection of Two Simple Polyhedra, Informatica 14 (1990), No. 1, 49-54.

Urbančič, Tanja, Avtomatsko učenje vodenja dinamičnih sistemov, Informatica 14 (1990), No. 4, 39-48.

Zaveršek, P. in P. Kolbezen, Računalniška mreža in CATV, Informatica 14 (1990), No. 4, 81-86.

Žalik, B. in N. Guid, Vgradnja Eulerjevih operatorjev in njihova uporaba pri tvorbi teles s translacijskim pomikanjem, Informatica 14 (1990), No. 2, 32-38.

Železnikar, A.P., Giving Priority to Informational Technology? Informatica 14 (1990), No. 1, 1.

Železnikar, A.P., An Introduction to Informational Algebra, Informatica 14 (1990), No. 1, 7-28.

Železnikar, A.P., Time and Temporality as Information, Informatica 14 (1990), No. 2, 12-31.

Železnikar, A.P., Understanding as Information (Part One), Informatica 14 (1990), No. 3, 8-30.

Železnikar, A.P., Understanding as Information II (Part Two), Informatica 14 (1990), No. 4, 5-30.

Žerdin, Š., N. Guid, and B. Žalik, Minimalne razdalje med geometrijskimi objekti, Informatica 14 (1990), No. 4, 52-57.

Novice in zanimivosti — News

Blaznik Polona in J. Šilc, Prva mednarodna konferenca o Moduli-2, Informatica 14 (1990), No. 1, 96-97.

Gams, M., Kritika članka "Zen and the Art of Modular Engineering, Informatica 14 (1990), No. 2, 70.

Owen, K., Edward Fiegenbaum (Interviewed for Expert Systems), Informatica 14 (1990), No. 1, 103.

Palian, M., EUUG Conference Report, Informatica 14 (1990), No. 1, 90-95.

Vrtačnik, Metka, 13. Mednarodno ONLINE srečanje (12.-14. decembra 1989 v Londonu), Informatica 14 (1990), No. 2, 65-69.

Železnikar, A.P., Kaos in informacija, Informatica 14 (1990), No. 1, 95.

Železnikar, A.P., Organization and Information Systems (Bled, September 13-15, 1989), Informatica 14 (1990), No. 1, 98.

Železnikar, A.P., O računalniški industriji v Sloveniji, Informatica 14 (1990), No. 1, 99.

Železnikar, A.P., Razmislek o Iskri kot korporaciji, Informatica 14 (1990), No. 1, 100-103.

Železnikar, A.P., Information Technology in Slovenia, Informatica 14 (1990), No. 2, 44.

Železnikar, A.P., Professor Saša Prešern in Australia, Informatica 14 (1990), No. 2, 44.

Železnikar, A.P., Upravljanje visokotehnološke korporacije, Informatica 14 (1990), No. 2, 45-64.

Železnikar, A.P., Nova knjiga profesorja Joza J. Dujmovića, Informatica 14 (1990), No. 2, 70.

Železnikar, A.P., Pojasnilo urednika, Informatica 14 (1990), No. 2, 70-71.

Železnikar, A.P., The Breakthrough of Unix, Informatica 14 (1990), No. 3, 87-89.

Železnikar, A.P., Information and Postmodernism, Informatica 14 (1990), No. 3, 89-93.

Železnikar, A.P., LIRA Jugoslovenska konferencija Logika i Računarstvo, Informatica 14 (1990), No. 3, 94.

Železnikar, A.P., Slovensko računalništvo v kapitalizmu, Informatica 14 (1990), No. 3, 94.

Železnikar, A.P., The European Journal of Information Systems (EIJS), Informatica 14 (1990), No. 4, 91.

Železnikar, A.P., Cybernetica (A Quarterly Review of the International Association for Cybernetics), Informatica 14 (1990), No. 4, 91.

PHILOSOPHY, SCIENCE, TECHNOLOGY

ON THE WAY TO INFORMATION

Anton P. Železnikar

In this volume Anton P. Železnikar confronts the conceptual, theoretic, and formal problems of information to unfold the meaning behind the redefined notion of information.

The first essay, **On the Way to Information,** is an introduction into the realm of information which seems to be given to us for ever, however, we do suddenly and unexpectedly ascertain that before invisible perspective of this realm is anew opening to us. Things inform and are informed in a spontaneous and circular way by their informing, counter-informing, and embedding of information. We can say: *As long as a being is informing, it is being.*

Information Determinations invite us to rethink the entire cultural and individual information. Some new, linguistically nonlegal terms would be useful, for instance, informism, informingness, informable, informativism, informatical, informatism, to informatize, informatization, etc. Information is understood as informational process irrespective of forms and processes of life, intelligence, and culture (autopoiesis, metaphysics, behavior, creativity, language, philosophy, ideology, dialectic, science, technology, ethics, aesthetics, art, crisis, incapability, etc. qua information).

Principles of Information is the basic text for the concept of information as the mastery over itself. On the basis of these concepts it will be possible to develop the so-called informational logic, algebra, a theory of discourse, time, understanding, etc. concerning philosophy and theory of information as well as technology of informational systems (beings, programs, artificial machines, etc.).

The next essay treats **God as Information** within a being's metaphysics and shows the informational regularity and human reasonableness of God as the kernel information, the informational splitting (parallelism) of human metaphysics enabling the so-called discourse with the Informational Other and the reasonableness of concept of God as the most pretentious and all-embracing information.

Information and Postmodernism presents the philosophic and cultural relevance of the notion of information in the postmodern era.

On the Way to Information

1

Anton P. Železnikar

Ljubljana 1991

Soon on your bookshelf!

Over 220 pages of exiting reading which concerns the philosophy and research of information in neural science (mind-brain domain), psychology, science, and technology (artificial intelligence). This book is a redesign of essays published by the author in *Informatica, Cybernetica,* and other journals.

Volume 2 will include the formal theory of information and its application in several domains of philosophy (time, understanding), linguistcs (formalization of semiotics), mathematics (theory of foundations), sociology (theory of discourse), etc.
Volume 1 appears at the beginning of 1991, published by the author (> > $9.95).

Informatica

Informatica *is published four times a year in Winter, Spring, Summer and Autumn by the Slovene Society Informatika, Iskra Delta Computers, Stegne 15C, 61000 Ljubljana, Yugoslavia.*

Informatica

Informatica *is published four times a year in Winter, Spring, Summer and Autumn by the Slovene Society Informatika, Tržaška cesta 2, 61000 Ljubljana, Yugoslavia.*

Volume 14 Number 2 April 1990 YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

CONTENTS

Volume 14 Number 4 October 1990 YU ISSN 0350–5596

Informatica

A Journal of Computing and Informatics

CONTENTS