Volume 16 Number 2 June 1992
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 1992 (Volume 16) 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 Tomi Pretnar, Bratov Komel 52, Ljubljana.

Informacija za naročnike

Informatica (YU ISSN 0350 – 5596) izide štirikrat na leto, in sicer v začetku marce, junija, avgusta in novembra.

Letna naročnina v letu 1992 (letnik 16) se oblikuje z upoštevanjem tečaja domače valute in znaša okvirno za podjetja DEM 30, za zasebnike DEM 15, za študente DEM 8, za posamezno številko pa DEM 10.

Številka žiro računa: 50101 – 678 – 51841.

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

Tisk: Tiskarna Tomi Pretnar, Bratov Komel 52, Ljubljana.

Na podlagi mnenja Ministrstva za informiranje št. 23/216-92, z dne 27.3.1992, šteje znanstveni časopis Informatica med proizvode informativnega značaja iz točke 13 tarifne številke 3, za katere se plačuje davek od prometa proizvodov po stopnji 5%.

Pri financiranju časopisa **Informatica** *sodeluje Ministrstvo za znanost in tehnologijo, Slovenska 50, 61000 Ljubljana*

Volume 16 Number 2 June 1992
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Letnik 16 Številka 2 Junij 1992 YU ISSN 0350 – 5596

Informatica

Časopis za računalništvo in informatiko

VSEBINA

AN INTRODUCTION TO STRUCTURED SYSTEMS ANALYSIS

INFORMATICA 2/92

Keywords: System analysis, Information systems

Talib Dimij
Univerza v Ljubljani, Ekonomska fakulteta

ABSTRACT: This paper deals with well-known and very useful method for information systems development. This method is Structured Systems Analysis & Design. The reason for this work is to introduce an excellent tool to the system analyst briefly and efficiently. The method consists of two parts: Systems analysis and Systems design. This paper deals with Systems analysis, because the development of a reasonable logical model is the first step to successful development of the information system.

ABSTRAKT: Članek obravnava znano in zelo uporabljivo metodo za izgradnjo informacijskih sistemov. To je metoda strukturirane sistemske analize. Razlog, ki me je vzpodbudil k temu delu je, da uporabniku oz. sistemskemu analitiku predstavim to odlično orodje na kratek in učinkovit način. Metoda je sestavljena iz dveh delov in sicer: sistemska analiza in sistemsko načrtovanje. Ta članek obravnava sistemsko analizo, kajti razvoj zadovoljivega logičnega modela je prvi korak k uspešnemu razvoju informacijskega sistema.

Introduction

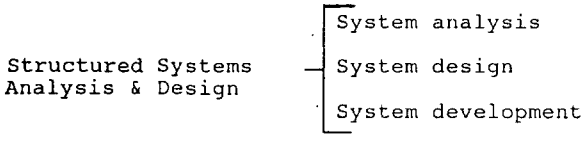
Computer-based information systems are developed in a sequence of steps. This sequence is known as system life cycle.

A huge range of detailed activities is needed to build a computer-based information system. Typical activities include writing a program, designing a form or selecting a piece of equipment. For this reason it is more common to define the life cycle as a sequence of higher level phases. Each phase is made up of more detailed activities than the last, and each has a specific goal (6).

The usual way to define the detailed activities of each phase is to review each phase when it is completed. The review produces a report on the outcome of the phase and also defines the goal as well as a detailed plan for the next phase.

Structured diagram of Structured systems analysis and design is shown in Figure 1. This paper intends to discuss the most important part of this method, this is the System Analysis.

Figure 1. Structured Systems Analysis & Design

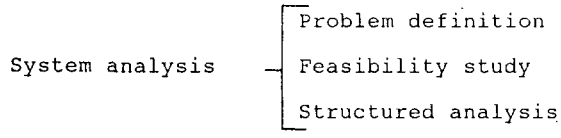


System Analysis

The key to use System analysis is the building of logical model of the required system and statment of systems objectives and constraints.

Figure 2 shows three important phases of the System analysis.

Figure 2. System analysis

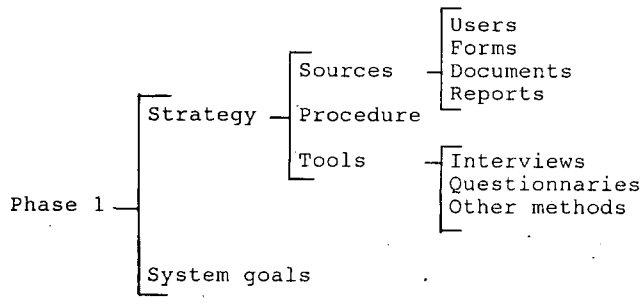


Phase 1 - Problem definition

This is a very important phase. It defines the problem to be solved and sets the direction for the whole system. It also defines the system bounds, which define what parts of the system can be changed and what parts are outside its control. Three important factors are the output of this phase. These factors are the system goals, bounds, and resource limits.

Information must be gathered in an organized way to ensure that nothing is overlooked and that all system detail is captured. All users must be consulted to ensure that every system problem, user requirement and objective is defined.

Figure 3. Problem definition



Search strategy:
It is necessary to develop a search strategy for gathering information. Search strategy is formulated by two important elements. First, identifying the information sources, and then establishing the search procedure for getting the information.

Sources: There are a number of sources of information for a system, such as system users, forms, documents, reports and so on.

System users are usually the firm information source investigated by analysts. From users it is possible to find out the existing system activities and to determine the user objectives and requirements. The usual search methods here are interviews or sometimes questionnaries.

Forms and documents are useful sources of information for system data flows and transactions. The search begins with the analyst obtaining a list of such document from the system users. Analysts then go through the documents to find their data elements and data structures.

Reports indicate the kinds of outputs needed by users. It can be used as a basis for user interviews to determine any new output requirements that user may have.

Procedure: search procedure defines where to begin the search and how to continue. It also indentifies the sequence in which sources will be searched. Search procedure is a top-down approach. That's why, it is better to begin at the top level and proceed to the bottom level to get detailed information. This approach usually commences with a set of interviews with top-level managers to determine the main system functions and activities. The analyst then searches for more detailed information to describe these functions and activities. This detailed information is obtained by interviewing operational personnel.

As soon as possible after the interview has ended, transcribe your notes. Ideally, the notes should concentrate on key ideas; use your memory to fill in the details. Share your summary with interviewee; it provides an excellent opportunity for correcting misunderstandings.

Tools: A number of search methods can be used to gather informtion about a system. These methods are interviewing techniques, questionnaries, and studying documents and reports.

Interviewing is the most common method of gathering information from users. Interviewing is a continuous process that is used by the analyst to gradually build a model of the system and to gain understanding of any system problems.

It is usually wise to begin interviewing at the top levels of the organizational areas, in order to get support and cooperation from management before beginning to look into particular organizational activities. The interview process follows a fairly structured path. We gain an appreciation of the overall system operation from management, then go into detailed operations by interviewing system users at various levels of system operation.

We should not expect to obtain all of the information required from one user in the course of one interview . There is usually a series of two, three or even more interviews with a given user. Usually the analyst begins with an initial interview to meet the users. The first interview is then followed by a number of interviews to gather all the major facts known to the user. Then there may be one or more reinterviews to verify these facts and any models developed by the analyst and to gain additional information to complete the analyst's study.

Effective interviewing is the result of careful preparation. The interview plan specifies the purpose of the interview, the users to be interviewed, the sequence in which the users are interviewed, and specific questions for every interview. A well-conducted interview consists of three distinct parts: an opening, a body, and a closing.

The Opening is the key objective to establish rapports. Begin by identifying yourself, the topics you plan to discuss, and the purpose of the interview. Tell the user why he or she was chosen for the interview. Where appropriate, identify the managers who have authorized the interview.

The Body must be created carefuly. The analyst should generally begins with a relatively broad, open question, and gradually through increasingly specific follow-up questions, focus the interview on particular points of concern. During the interview, you must listen to the answers and jot down the key points. Be selective and do not record every word.

In the Closing of the interview, when you have all information you need, thank the subject for cooperating, and offer to make your written summary available for review.

Questionnaires can be used to find out information about a system instead of interviews. Questionnaires contain all questions that a user answer to provide information sought by the analyst. The questionnaire is then send to the user and replies are analyzed by the analyst. Questionnaires are useful for gathering numerical data or getting relatively simple opinions from a number of people, but they are not very effective for in-depth searches or for identifying system problems or solutions.

Other search methods are involve studying and going through sources such as documents, reports or computer programs and looking for relevant information. Documents, reports and forms are important for identifying data elements and data structure.

System goals:
The main porpuse of the problem definition is to define the goal of the proposed system.

One important guideline for goal setting is to remember that goals should not be unrealizable ideals that are subsequently ignored. They must developed with the practicalities of the organization. One way to ensure that such practicalities can be met is to elaborate the system goal into more detailed subgoals that consider organization constraints. These subgoals are used in later stages to guide detailed analysis and design.

Another guideline for goal definition is the identification of deficiencies in the existing system. The system goal is to remove subdeficiencies. Deficiencies are usually found through interviews or by examining documents about the system performance.

Above mentioned goals are found by examining the internal operation of a system. Goals can

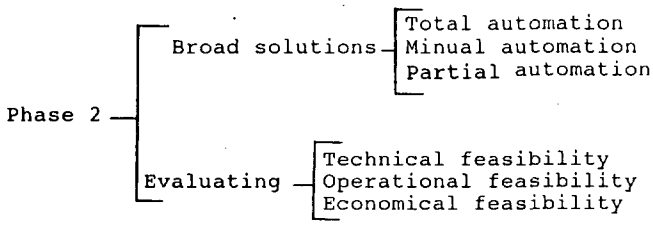
be examining a system external environment. It is always worthwhile to examine what similar organizations do and see whether some of their methods can be used in the proposed system.

Phase 2 - Feasibility study

Problem definition and feasibility study are important because they set the direction for the remainder of the system delvelopment.

As it was mentioned, the first phase defines the system goals. The feasibility study phase determines a feasible way for achieving this goal. It begins once the goals are defined and agreed upon.

Figure 4. Feasibility study



Broad solutions: Feasibility study starts by generating broad possible solutions, which are used to give an indication of what the new system should look like.

Two things must be kept in mind when proposing a broad solution. First it should give people a good idea of what the new system will look like and also convince them that it will work. Another objective is to form an estimate of cost. The solution should also specify what is to be done by computer and what will remain manual. So, what is needed in this phase is a broad statement of the kind of information that will be made available to users and its effect on user operations.

Usually a number of broad solutions are proposed and evaluated to find the best solution. One guideline here is to start with three alternatives, namely:

- a totally automated system;
- a system with minimal automation; and
- a system somewhere in between.

Evaluating the proposal: Three things are done in the evaluation of the proposal:

- **Technical feasibility** is the technology needed for the proposed system. If the technology is available, can it be used in the organization.

- **Operational feasibility** gives informtion about the system efficiency. Can the proposed system fit in with the current operations? Does it provide the right information for the organization's personel? Does this information arrive at the right place in time?

- **Economic feasibility** provide a reasonable estimate of system cost. Is it worthwhile to pursue the system? Many organizations place a great emphasis on economic analysis. Such economic analysis is described as cost-benefit analysis.

Cost-benefit analysis:
It usually includes two steps. One is to produce the estimates of costs and benefits. The other is to determine whether the system is worthwhile once these costs are ascertained.

The goal of first step (**producing costs and benefits**) is to produce list of what is required to implement the system and a list of the new system's benefits.

Cost-benefit analysis is always clouded by both tangible and intangible items.

Tangible items are those to which direct values can be attached. Some tangible costs are:

- Equipment costs for the new system.
- Personnel costs. These include personnel needed to develop the new system (analysts, designers and programmers).
- Material costs. These include manual production and other documentation.
- Other costs. These include consultant's costs, travel budgets and so on.

Intangible items are those whose values cannot be precisely determined. For example how much is saved by completing a system earlier or providing new information to decision makers.

The sum value of costs of items needed to implement the system is known as the cost of the system. The sum value of savings made is known as the benefit of the new system. Once we agree on the costs and benefits we can evaluate whether the project is economically viable.

Determining whether a system is worthwhile, we can use the present value method.

The idea of **the present value method** is to determine how much money it is worthwhile investing now in the order to receive a given return in some years. For example, the present value of $16,105 in Year 5 is $10,000 today at 10 per cent interest. If a system that pays back $16,105 in Year 5 costs $11,000 today, that system is not worthwhile. On the other hand, it is worthwhile if it only costs $9000 today.

To some extent the present value method works backwards. First, the system benefits are estimated for each year from today. Then, we compute the present value of these savings. If the system cost exceeds the present value then it is not worthwhile.

Let us see whether the system is worthwhile at a discount rate of 10 per cent. To do this we find the present value of the benefit at each year. The formula is:

$$\text{Present value} * (1 + r/100)^n = \text{Benefit at Year } n$$

$1/(1 + r/100)^n$ is the discount factor

Thus, for example, the present value of the $40,000 benefit at Year 2 is computed as:

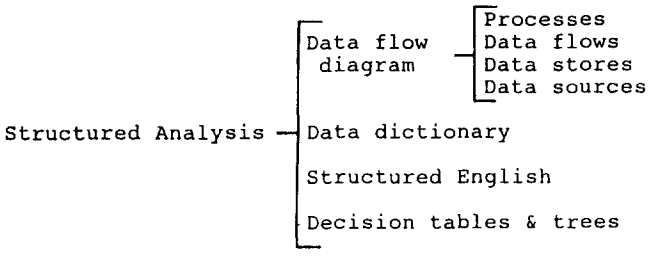
$$\text{Present value} = 40{,}000 / (1 + 10/100)^2 = 33{,}057$$

Phase 3 - Structured Analysis:

Structured Analysis is defined as the use of Data flow diagrams, Data dictionary, Structured English, Decision tables, and Decision trees, to build a target document, the structured specification (4).

These tools provide a communications mechanism between the analysts and the users, and a precise means of recording the specification so that the design can be translated into computer code and implemented (2).

Figure 5. Structured Analysis



Data flow diagram:

The data flow diagram (DFD) is one of the most important tools used by analysts. DFD is the logical model of the system. The model (DFD) does not depend on hardware, software, data structure, or file organization.

A data flow diagram (DFD) is a charting tool which traces a network of data flows through a system and provides information at varying levels of detail. This enables the system requirement to be partitioned, analyzed and specified in manageable pieces (2).

DFD uses four kinds of symbols. These symbols are used to represent four kinds of system components: process, data store, data flow and data source (external entity).

Process: Processes show what system do. Each process has one or more data inputs and produces one or more data outputs. A process transforms incoming data flows into outgoing data flows. Process has a unique name and a number. This name and number appear inside the circle that represents the proces in a DFD. A single process is not necessarily a program. It might represent a series of programs, a single program, or a module in a program.

Data store: A data store is a repository of data. It might represent a data base, a file, or even a piece of file. Each data stores are represented by a thin line, closed at one end. Each data store can be identified by a "D" and an arbitrary number and a name inside the box.

Data flow: Data flows model the passage of data in the system and are represented by lines joining system components. The direction of the flow is indicated by an arrow and the line is labeled by the name of the data flow.

What is the difference between a data store and data flow? A data flow is data in motion; a data store is data at rest (3).

External entity: External entity is an individual or organizational unit outside the boundary of the system that interfaces with system. External entities are either supply input data into the system or use the system output. External entities that supply data into a system are also called sources, and external entities that use system data are sometimes called sinks. External entities are represented by a rectangle.

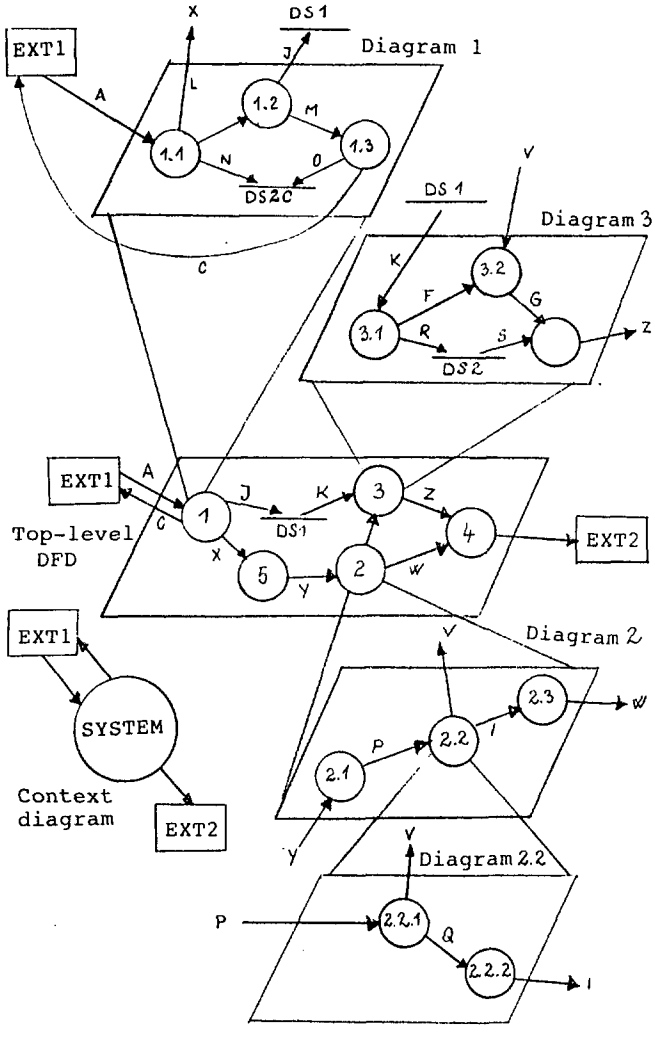
Leveling data flow diagrams: The drawing of data flows is a highly interative process. The best way to begin is to model the whole system by one process. This process is the highest level data flow diagram. It is called as contex diagram.

Context diagram shows all the external entities that interact with the system and the data flows between these external entities and the system.

Each process can be exploded to a further level of detail until the lowest level, called a functional primitive elementary process is reached.

A primitive is a process which generally executes a single function and can be described in less than a page of structured English (2). When the lowest level consists only of primitive, the leveling is complete.

Figure 6. Leveling



Leveling allows an analyst to start with a top-level function and elaborate it in terms of its more detail components. Although leveling is very useful tool, a number of conventions must be observed to ensure that no information is lost as a DFD is leveled. Some these conventions are:

a) Process numbering: The context diagram is usually given the number 0. Processes in a next level (top-level DFD) are numbered consecutively, starting with 1 and continuing until all processes have beeh labeled. This level shows the major processes in the system. In Figure 6 the top-level DFD has five processes numbered 1, 2, 3, 4 and 5.

As each process is leveled, its DFD diagram is given the same number as the process. Thus, in Figure 6, the leveled DFD of process 1 is named Diagram 1. Each process in the leveled DFD receives a number made up of its diagram number, follow by a period, followed by a number within the leveled DFD. Thus the processes in DFD Diagram 1 have process number 1.1., 1.2. and 1.3.

b) Data flow balancing: Data flow balancing requires that all the data flows entering a process are same as those entering its leveled DFD. Similarly, all the data flows leaving the process are the same as those leaving its leveled DFD. If you look at Diagram 2, which is the leveled DFD of Process 2 in the top-level DFD. You should note that the inputs (Y) and the outputs (V and W) of Process 2 are the same as the inputs and outputs of Diagram 2.

c) Data stores introducing: Data stores are local to the leveled diagram. In Figure 6 data store DS2 contains data that is local to Process 3. Hence this data store does not appear in the top-level DFD but only in its leveled diagrams.

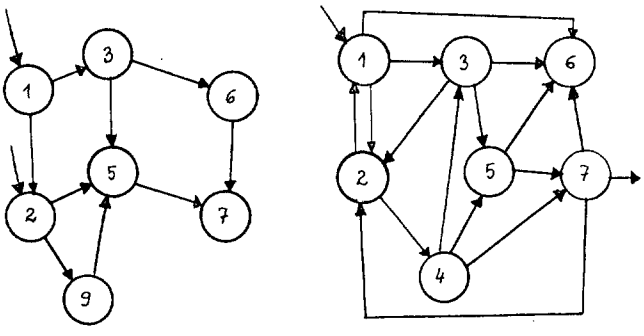
d) External entities introducing: All external entities should appear on the context diagram. They should also appear with a DFD if any process in the DFD has a flow to or from the external entity. In Figure 6, external entity EXT1 has flow to and from Process 1. This external entity appears on the leveled DFD for Process 1.

It is sometimes difficult to know far to level down, how many processes to include on a data flow diagram or even where to start. It is imposible to give precise answers but some guidelines can be suggested.

Most practitioners say that about seven plus or minus two is the ideal number of processes on a data flow diagram. This number can be clearly understood by a visual examination but is not too small to be trivial. A larger number is sometimes too hard to understand; smaller number often includes too little information to be useful.

Figure 7. Reducing complexity by minimizing flow between processes

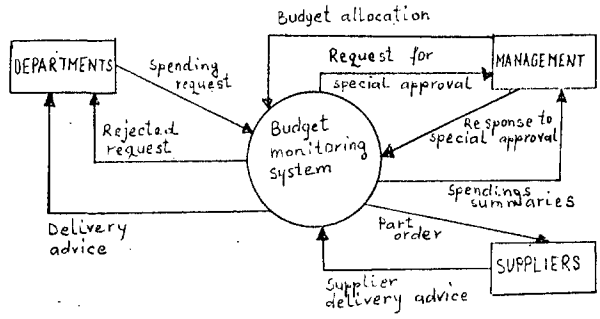
a) Minimized data flow b) Too many flows



However, one should not take a trivial approach and suggest that leveling only involves taking a higher-level process and partitioning it into seven lower level-processes. Consider Figure 7. It contains two data flow diagrams, each with seven processes. One of these diagrams is clearly less complex and hence easier to understand than the other. Figure 7 illustrates another requirement of leveling - the interfaces between processes must be minimized. The analyst must carefully analyze the top-level process. If the interconnections are too complex, the leveling should be re-examined and restarted. This may continue over a number of iterations until a satisfactory solution is found.

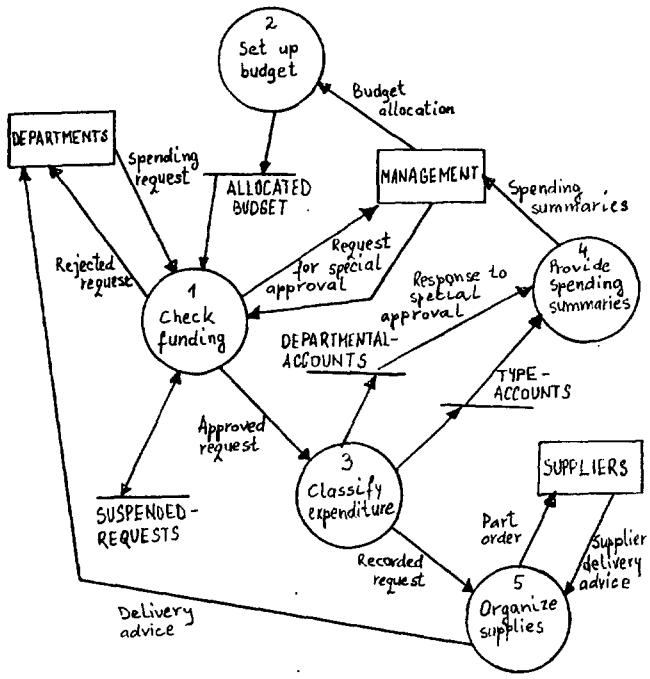
Example: Figure 8 is an example of a context diagram. It models the "budget monitoring system". This system interacts with three external entities, DEPARTMENTS, MANAGEMENT and SUPPLIER. The main data flows from DEPARTMENTS are "spending request". In response, DEPARTMENTS either receive "rejected request" data flows or "delivery advice" data flows. MANAGEMENT receives "request for special approval" data flows to which it responds. MANAGEMENT also sends "budget allocation" data flows to the system and gets "spending summaries" data flows. Suppliers receive "part order" data flows and return "supplier delivery advice" data flows.

Figure 8. Context diagram



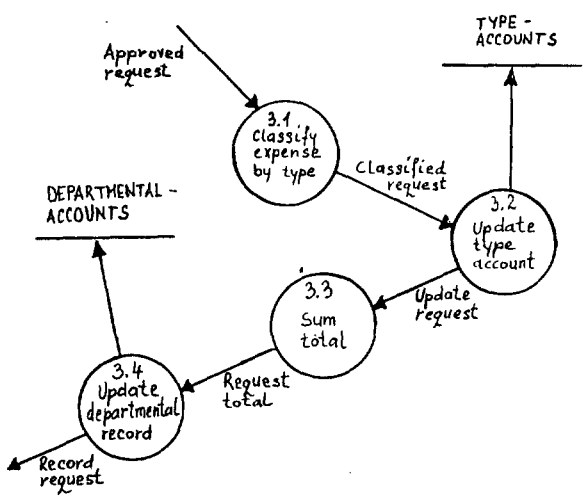
This model does not describe the system in detail. For more detail it is necessary to identify the major system processes and draw a data flow diagram made up of these processes and the data flows between them. The DFD that shows the major system processes is called the top-level DFD (shown in Figure 9). The top-level DFD shows the various processes that make up the system. Each process has a unique name and number. From Figure 9 we see that data flow "spending request" from DEPARTMENTS goes to the "check funding" process. This process looks up the allocated budget and determines whether special permission is needed from MANAGEMENT to proceed with the request. Approved requests go to the "classify expenditure" process where they are entered into data stores, DEPARMENTAL-ACCOUNTS and TYPE-ACCOUNTS. Finally, if required a "part order" for parts originally specified in a "spending request" is placed with suppliers. There are two other processes. One is to "set up budget" and the other to "provide spending summaries".

Figure 9. A top-level DFD diagram



We can continue to expand each process in Figure 9 into a more detailed FDF. As an example, Figure 10 is a detailed description of the "classify expenditure" process labeled as Process 3 in Figure 9. Each process in this detailed DFD is labeled with a 3 followed by a number to show that each is an expansion of Process 3. The input to this DFD is the data flow "approved request". Process 3.1 classifies the entries in each "approved request" by expense type and Process 3.2 uses this classification to update data store TYPE-ACCOUNTS. All the entries for the whole request are summed by Process 3.3, and Process 3.4 uses the total to update data store DEPARTMENTAL-ACCOUNTS.

Figure 10. Expansion of "classify expenditure" process



Data dictionary:

The purpose served by the data dictionary during structured analysis is to maintain definitions of data flows, data stores, and processes. The basic information included in each data flow or data store entry is its name, aliases and composition.

The data flow name identities the content of the data flow and the action taken on it. The alias is any other name by which the data flow may be known.

The compositon can be defined using the convention notations described by DeMarco. The conventions are:

= means is equivalent to, + means and, [ ] means either-or, { } means iteration of the components enclosed, and ( ) means that the enclosed component is optional.

For example the ORDER document can be described as

ORDER = [SUPPLIER-NO / SUPPLIER-NAME] + DATE-ORDERED
+ DATE-REQUIRED + ORDER-NO
+ (ORDER-STATUS) + {ITEM-NO + QTY-ORDERED}

Here ORDER is the structure name. The order is made up of seven data elements: SUPPLIER-NO, DATE-ORDER, DATE-REQUIRED, ORDER-NO, ORDER-STATUS, ITEM-NO and QTY-ORDERED.

In this notation, structure components are described by using the plus sign (+). The square brackets [ ] indicate alternative structures. Thus ORDER may contain SUPPLIER-NO or SUPPLIER-NAME but not both. The round brackets ( ) indicate an optional component. Thus an ORDER may or may not contain the data element ORDER-STATUS. The curly bracket { } indicate that a structure or data elemente can be repeated. So, the structure
{ ITEM-NO + QTY-ORDERED } can be repeated.

Structured English:

Structurd English is used to give unambiguous process description. There are several variations of structure English. It is a very limited subset of the English language. Structured English syntax provides the keywords to structure process specification logic. Process specification logic consists of a combination of sequences of one or more imperative sentences with decision and repetition constructs.

Imperative sentences: An imperative sentence usually consists of an imperative verb followed by the contents of one or more data stores on which the verb operates. In imperative statements we can use commands such as move, get, write, read, compute and so on.

Data flow names appear in lower case between quotes while specific data items in the data flows and data stores are capitalized.

A sequence of imperative statements can be grouped by enclosing with BEGIN and END.

```
BEGIN
   Receive "sale report"
   Get SALES record for PART-NO in
     "sale report".
   TOTAL-QTY = TOTAL-QTY + QTY-SOLD.
   SALE-VALUE = QTY-SOLD * UNIT PRICE.
   TOTAL-VALUE = TOTAL-VALUE + SALE-VALUE.
   Write SALES record.
   Send "summary advice".
END
```

Decisions: Two types of decisions structure usually appear in Structured English. First type shows a structure which allows a choice between two groups of imperative statments. The keywords IF, THEN and ELSE are used in this structures. If a condition is "true" then GROUP A sentences are executed. If a condition is "false" then GROUP B sentences are executed.

```
IF condition THEN
   BEGIN
      GROUP A
   END
ELSE
   BEGIN
      GROUP B
   END
```

The second type shows a choice between any number of groups of imerative sentences. The keywords CASE and OF are used in this structure. The value of a variable is first computed. The group of sentences excuted depend on that value.

```
CASE NAME OF
    A:
     BEGIN
        GROUP A SENTENCES
     END
    B:
     BEGIN
        GROUP B SENTENCES
     END
     .
     .
     .
    Z:
     BEGIN
        GROUP Y SENTENCES
     END
  END
```

Repetition: For specifying itertions we use two commands. First way is to use the WHILE...DO structure. Here a condition is tested before a set of sentences is processsed. The second way uses REPEAT...UNTIL structure. Here the group of sentences is excuted first and then the condition tested.

```
WHILE  condition  DO
  BEGIN
     GROUP A sentences
  END

REPEAT
  BEGIN
     GROUP A sentences
  END
UNTIL  condition
```

An alternative to the WHILE...DO structure is to use the FOR structure.

```
FOR INDEX = INITIAL TO LIMIT
   BEGIN
      GROUP A sentences
   END
```

Decision Tables and Decision Trees:

Algorithms involving multiple nested decisions are difficult to describe using structured English, pseudocode or other techniques.

A better way of describing such logic is to use decision tables or decision trees. These two efficient methods are prefered where one of a large number of actions is to be selected. The action selected depends on a large number of conditions.

Decision tables: A decision table is first devided into two parts - the conditions and the actions. The conditions part states all the conditions that are applied to the data. The actions are the various actions that can be taken depending on the conditions. The table is constructed by using columns so that each column, called role, corresponds to one combination of conditions.

A sample decision table is shown as Figure 11. Assume that the basketball coach has asked us to look through the student records and produce a list of all full-time male students who are at least 185 cm tall and who weigh at least 85 kg.

Figure 11. Decision table

CONDITIONS					
Is the student male?	Y	N			
Is the student taking at least 12 credit hours?	Y		N		
Is the student at least 185 cm tall?	Y			N	
Does the student weigh at least 85 kg?	Y				N
ACTIONS					
List the student's name and address	X				
Reject the student		X	X	X	X

The easiest way to understand a decision table is to read the first question: Is the student male? There are two possible answers: yes (Y) or no (N). If the answer is yes; we can not make a decision. Three more tests must be passed. What if the answer is no? The student is not male, she is not a candidate for the basketball team, and thus can be rejected. Move down the column containing the N, and note the X on the action entry line following "Reject the student".
Move on to the second question: Is the student taking at least 12 credit hours? There are two possible answers: yes or no. If the answer is yes; again, we can not make a decision; two more test remain (must be passed). What if the answer is no? The student can be rejected. Move down in third column and note the X on the action entry line following "Reject the student".

Read the rest of the table. It clearly shows that the student's name and address will be listed only if the answers to all four questions are yes, but that the student will be rejected if the answer to any one is no.

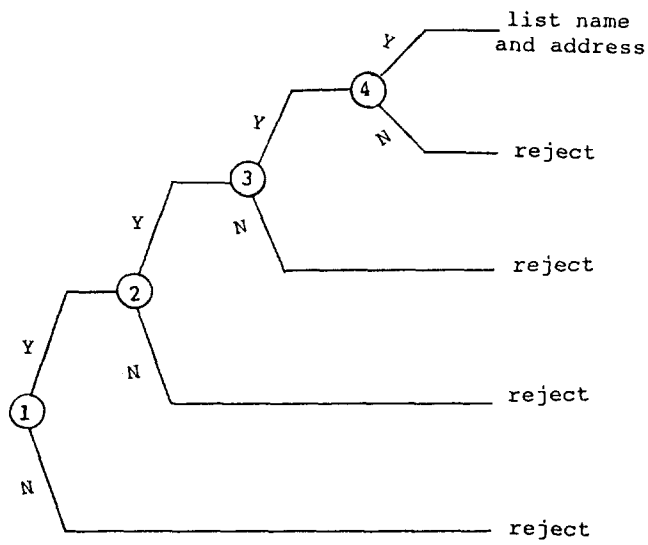
When an algorithm involves more than two or three nested decisions, a decision table gives a clear and concise picture of the logic (3).

Decision trees: As an alternative, the anlyst might use a decision science tool called decision tree. Decesion tree is easy to constuct, and graphically illustrate decision logic.

The decision tree is a graphic representation of the decisions, events, and the consequences associated with a problem. Once the tree is drawn, probabilities can be associated with each branch, and expected values of the outcomes computed (3).

Figure 12 is shown a decision tree for basketball problem logic. The algorithm consists of a series of four nested questions or decisions. Each question or decision is represented as a circle. Begin with the first one: Is the student male? There are two possible answers: yes or no. If the answer is no: the student is rejected; a yes answer leads to another question. Again, there are two possible answers: yes or no. Again, a no means reject the student, while a yes response leads to another question and so on. Follow each branch on the tree to its logical outcome. Student's name and address are listed only if all four questions are answered affirmatively.

Figure 12. Decision tree



1. Is the student male?
2. Is the student taking at least 12 hours?
3. Is the student at least 185 cm tall?
4. Does the student weigh at least 85 kg?

Conclusion:

Structured Systems Analysis as an excellent hierarchical design approach provides a well-organized and manageable system. Throughout the process, we deal with manageable pieces of information which can be controlled. It is the step-by-step evolution of the analysis. We begin with the present physical system, convert it into the present logical system. This logical system provides us with a framework to specify a new physical system.

Litrature:

1) J. Daniel Couger, Mel A. Colter, Robert W. Knapp: Advanced System Development/Feasibility Techniques. John Wiley & Sons, New York, 1982.

2) Denis Connor: Information System Specification and Road Map. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1985.

3) William S. Davis: Systems Analysis and Design. Addison-Wesely Publishing Company, Massachusetts, 1984.

4) Tom DeMarco: Structured Analysis and System Specification. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1979.

5) Chris Gane and Trish Sarson: Structured Systems Analysis, Tools and Techniques. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1979.

6) I. T. Hawryszkiewycz: Introduction to Systems Analysis and Design. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1988.

7) Robert E. Leslie: System Analysis and Design, Method and Invention. Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1986.

8) James C. Wetherbe: Systems Analysis and Design. West Publishing Company, New York, 1988.

SCADA AND ENERGY MANAGEMENT

INFORMATICA 2/92

Keywords: SCADA program,
local power system, load shedding

D. Mrdaković, M. Križman, B. Medica
Odsek za energetiko in vodenje procesov
Institut »Jožef Stefan«, Ljubljana

ABSTRACT: With the appearance of modern SCADA software packages and with the development of VLSI technology, PC computer systems are becoming increasingly used as a workstation. The capacity of such station is determined according to the complexity of a local power system. The main problem is the selection of the suitable hardware and software for efficient supervision and control of the local power system. In this article the criteria for the selection of configurations of suitable hardware and directions of further development in this field are presented.

SCADA IN ENERGETSKO VODENJE IN NADZOR

POVZETEK: Z pojavom modernih SCADA programskih paketov in z razvojem VLSI tehnologije, se PC računalniški sistemi vse bolj pojavljajo kot delovne postaje. Zmogljivost takšne postaje se določi glede na zapletenost internega energetskega sistema. Glavni problem pri tem je pravilna izbira strojne in programske opreme za učinkovito vodenje in nadzor internih energetskih sistemov. V tem članku so predstavljeni kriteriji za izbiro strojne opreme in nakazane smeri nadaljnjega razvoja na tem področju.

1. INTRODUCTION

Traditional SCADA (Supervisory Control And Data Acquisition) systems have for a long time resembled the monolithic direct digital control systems, but as distributed process control matured, both the concept and components began to look inviting. With software developments of both process control and management systems the SCADA systems have infiltrated in the utilities industries where there are distant units of operation, often unattended, that require supervision from a control facility [BLI85].

With further development SCADA systems became more distributed. Most SCADA systems were evolved into networks of computers with functions distributed among numerous machines: man/machine workstations, remote terminal units, application processors, etc [BLI88].

The growing familiarity of SCADA system with the ever present personal computer transformed the PC into an integrated workstation for real-time process control. In addition, development of multitasking operating systems (Unix, OS/2, NT), the key to system integrity and high speed response, was an important factor in the acceptance of the PC.

Although SCADA system has traditionally utilized the operator to 'close the control loop' via supervisory control, the constant development of the SCADA system software has opened the door for the SCADA master to be more than just a monitoring device [BLI89].

With all features of the powerful, low-cost automation and control software, the SCADA system became a perfect tool for energy management.

The purpose of this paper is to obtain the basic information of suitable hardware and software in order to facilitate the choice to the user.

2. ENERGY MANAGEMENT AIM

In Slovenia consist calculated costs for electricity consist of item for power (consumed power in kWh) and item for peak load (maximum 15-minute average peak in month). The purpose of process control is to keep consumption smooth and steady over various periods of a day i.e. to keep peak load as low as possible.

Energy management offers three main methods to apply control strategies:

- Load scheduling,
- Load shedding,
- The use of additional power units (co-generating units, diesel aggregate).

Proper scheduling of production processes introduces time-driven loads and represents backbone or rough tuning in Predicted Demand Control PDC.

Load reduction for a longer period of time, i. e. load shedding, occurs when predicted power (projection to the end of demand interval) is higher then optimum power (target). Thus, it represents fine tuning of load management system.

With the use of additional power units it is possible to minimize load shedding and achieve optimised energy usage. Among those units a co-generating unit usually represents the best choice wherever the process requires industrial steam. it is very often a case that such a turbogenerator-set is already installed in the factory though originally not for the purpose of load management.

Figure 1 depicts the heat and power production and consumption in a local power system.

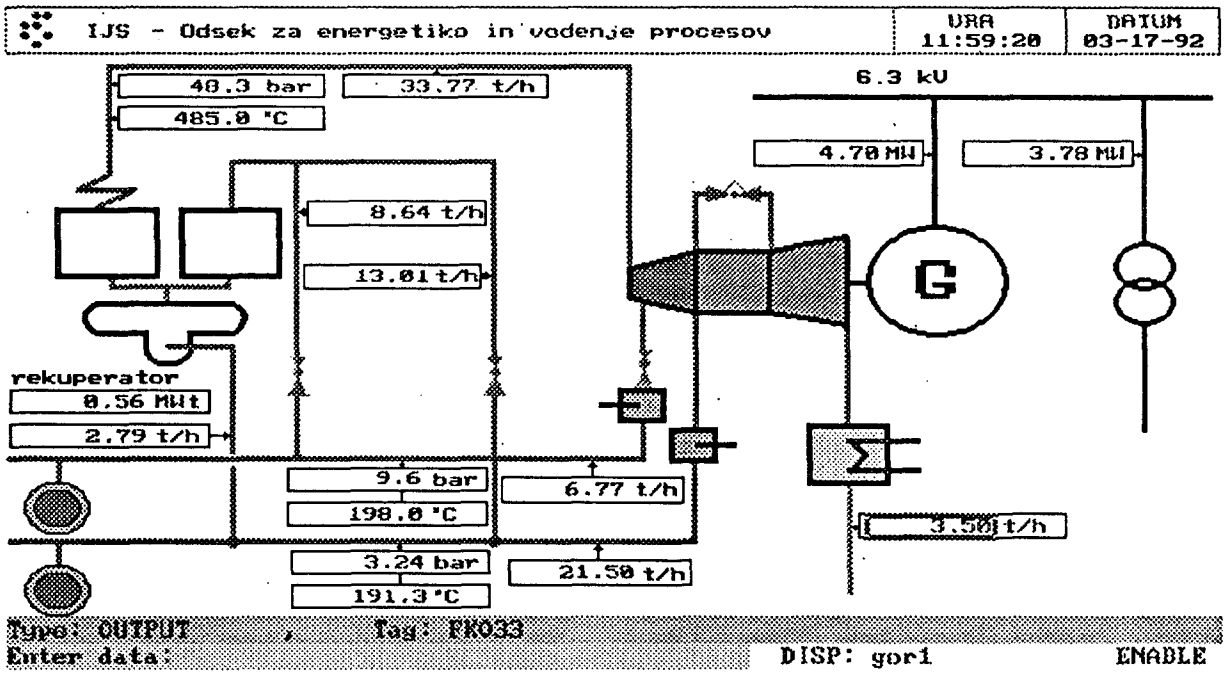


Figure 1: The heat and power production and consumption

By using the SCADA system for energy management implemented through monitoring and supervisory control of a local power system, significant savings can be achieved.

3. CRITERIA FOR PROPER CONFIGURATIONS

The proper selection of suitable software and hardware is of the crucial meaning for efficient supervision and control of local power systems. A priority task is therefore to elaborate the criteria for the right choice of [MRD90]:

- I/O intelligent devices, that operate as slave devices to a host computer,
- PC industrial computer, built to withstand the harsh physical conditions frequently found on the plant floor,

- SCADA software package for load management

3.1. I/O INTELLIGENT DEVICES

For intelligent analog and digital I/O devices, the next criteria has been made [MRD91B]:

- The possibility of locating the control point at each heating or conditioning unit in an energy management application. Thus, considerable installation savings and improved performance can be realised.
- The communication with the host computer over an RS422/485 communication link or LAN (i.e. ARCnet). The RS422/485 link offers excellent noise immunity, long cable lengths and reduces wiring costs. With LAN the increased operating efficiency is achieved.
- The possibility to outline the functions of the application software, thus offload host computer processing time and reduce data link activity (moving control to PLCs).
- The numerous combinations of analog and digital I/O module types.
- Photo-isolated analog or digital power I/O modules.
- The support of different baud rates, simply selectable.
- The possibility of additional programming on the PC with the appropriate development tool (for example: CYRANO visual language).

3.2. PC INDUSTRIAL COMPUTER

Industrial computer can be used as a workstation or a gateway to in-plant communications networks. Engineered specifically for the unique requirements of plant floor environments, the computer has to include following features:

- Passive backplane,
- Dual-fan cooling with removable filter,
- Security door,
- Vibration proof hold-down clamp.

It has to perform reliably under environmental conditions such as:

- Extended temperature,
- Extended vibration and shock,
- Extended voltage transients,
- Extended particulates.

For extremes the computer based on FTSA (Fault Tolerant System Architecture) is required. It protects data integrity and ensures immediate data and system recovery from catastrophic failure. The FTSA PC addresses the primary sources of PC failure - inconsistent power, damaged or destroyed data, user errors and component breakdown - by combining selective redundancy with elaborate diagnostic, regulatory and recovery systems into a synergistic unit, tied together by an enhanced, DOS-compatible BIOS.

3.3. SCADA SOFTWARE PACKAGE

We have made appropriate criteria for the selection of SCADA for supervisory control of a local power system [MRD91A].

- Possibility of the use of programmable blocks which enable the programmer to work with a suitable interpreter. To make the SCADA applicable to supervision and control, it must have besides standard input, output and calculation blocks also trend blocks, PID blocks, on-off control blocks and program blocks.
- Possibility to archive data. Observation of the history enables us to make reports and statistics later.
- Use of standard DOS functions or their presence in the main SCADA menu.
- Use of MMS standard (Manufacturing Message Standard) for the communication with the network. It assures compatibility with the MAP protocol.
- Possibility of the own algorithm development in a higher program language. Supervision and control of local power systems requires the use of optimisation algorithms for the needs of cogeneration units.
- With advantages of DLL (Dynamic Link Library), DDE (Dynamic Data Exchange),

OLE (Object Linking and Embedding), and GUI (Graphic User Interface), the feasibility to operate on the Windows platform is becoming more and more important.

4. CONFIGURATIONS

On the basis of the criteria we decided to use the following equipment [MRD92].

For intelligent I/O devices OPTOMUX units (OPTO22) has been chosen. OPTOMUX units can be configured to operate in multidrop or repeat mode via jumpers located on each unit.

For industrial computers IBM 7541 286/10, IBM 7561 386/25 and TEXAS MICRO 386 (486)/25 have been applied. Features such as a hardened case, filtered air cooling and full function industrial keyboard are included.

On the basis of the criteria we decided to use FIX-DMACS, modern PC-based distributed process control and supervisory control and data acquisition (SCADA) software package, with completely open architecture and on-line configuration.

With this SCADA software package eight applications have been realised in Slovene and two are still in progress.

Computers IBM 7541 were used for smaller applications, while for large and complex applications IBM 7561 and later TEXAS MICRO 386 (486)/25 were used, according to the requirements of specific applications. For process control applications FIX release 3.01 is accepted as a standard, while for distributed process control DMACS release 2.1 and the local area network Token Ring 16 Mbits/sec or DECnet speed 10Mbits/sec are applied.

INTELlution software (FIX and DMACS) was used for process animation, data archiving and collection, automatic elaboration of reports and implementation of our own programs in C language (a program package for load management that has been developed in our department).

A problem that we always faced was the DOS barrier of the first 640 KB. It was overcome by QEMM 386 driver on 386/486 computers, which runs itself, progress database and all background programs in expanded memory. On 286 computers an INTEL ABOVE card/2 MB of LIM 4.0 standard was used.

5. DESCRIPTION OF INSTALLED (APPLIED) SYSTEM IN THE PULP AND PAPER MILL "GORIČANE"

In the factory local heat and power system a microcomputer system is installed. The functions that are provided with this system are:

- monitoring of natural gas purchasing and transforming from main pipeline,
- monitoring of electric energy purchasing from public grid,
- monitoring and controlling of combined heat and power production system,
- monitoring of local heat and power distribution and individual consumer consumption.

Operating of condensing extracting steam turbine is based on heat following method. In order to achieve the cheapest substitution of power from public grid the control system for steam turbine condensing part is set. This system controls operating of steam turbine according to the built in tariff system.

There are two possibilities for steam turbine controlling:

- automatically, control system directly drives the turbine without operator confirmation or
- manually, operator is informed for optimal operating by control system but the final decision is left to him.

At the same time the programs for data continuous monitoring and archiving are upgrade. They include several tasks that were found as useful for factory energy management:

- monitoring of fuel and power consumption and peak load achieved
- elaboration of daily and monthly reports for power purchasing, production and energy supplying to local consumers including the evaluation of energy transformation efficiency.

Historical database was found very useful for factory and energy management. It is

especially used by quality control (estimation of specific energy consumption) and maintenance department, as well.

The described application, besides the improvement of overall energy efficiency on the factory level, has also showed its value in comparison with state of energy monitoring and controlling equipment in paper mills of neighbouring countries in West Europe.

6. CONCLUSION

In this article we have proposed some basic configurations in order to facilitate the selection of the equipment, required for efficient energy management of smaller local power system, to the user. However, the fast changing world of control made SCADA systems less expensive while functionality is increasing. Therefore the world trend in this field is tending to more distributed process control. In addition, with advanced technologies and adding intelligence, the control is moving to lower levels, even down to sensors.

7. REFERENCES

[BLI85] Blickley G. J., "SCADA Systems Affected by Distributed Control", Control Engineering, pp. 79-81, march 1985.

[BLI88] Blickley G. J., "SCADA Systems Merging with Distributed Control", Control Engineering, pp. 60-61, february 1988.

[BLI89] Blickley G. J., "Changes in SCADA Systems Increase Operator's Role", Control Engineering, pp. 95-96, february 1989.

[MRD90] Mrdaković D., "System Software for Process Control in Industry", Informatica, vol. 14, number 3, pp. 56-58, july 1990.

[MRD91A] Mrdaković D., "Criterria for the Choice of Program Packages SCADA Program Blocks for the Needs of Internal Energy System Control", IASTED91, pp. 463-465, 18.-21. february 1991.

[MRD91B] Mrdaković D., "Kriteriji za izbiro periferne merilno regulacijske računalniške opreme, primerne za nadzor in vodenje internega energetskega sistema", MIPRO91, pp. 2-97 - 2-100, 20.-24.05.1991

[MRD92] Mrdaković D., "PC Computer Station for Supervision and Control of an Internal Power System", IASTED92, pp. 491-492, 10.-12. february 1992

INFORMACIJSKI SISTEM IZOBRAŽEVALNE DEJAVNOSTI NA FAKULTETI ZA ELEKTROTEHNIKO IN RAČUNALNIŠTVO V LJUBLJANI

INFORMATICA 2/92

Ključne besede: fakultetni informacijski sistem, poslovanje študijskega sektorja, podatkovni model za področje izobraževalne dejavnosti

Viljan Mahnič
Univerza v Ljubljani
Fakulteta za elektrotehniko in računalništvo
Tržaška 25, Ljubljana

POVZETEK: V članku je opisan informacijski sistem izobraževalne dejavnosti, ki pokriva področja prijave za vpis, sprejemnih izpitov, evidence vpisanih študentov, izpitne evidence, analiz uspešnosti študija ter evidence diplomskih in zaključnih nalog. Sistem skoraj v celoti avtomatizira poslovanje študijskega sektorja, študentom pa omogoča, da se za izpite prijavljajo preko računalnika. Razvit je bil na Fakulteti za elektrotehniko in računalništvo v Ljubljani, prevzele pa so ga tudi nekatere druge fakultete ljubljanske univerze, tako da opisani sistem predstavlja enega izmed temeljev za izgradnjo celovitega visokošolskega informacijskega sistema.

EDUCATIONAL INFORMATION SYSTEM AT THE FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING: The paper deals with the educational information system which covers the following areas: admission applications, entrance examination, enrollment, examination records, various tables and statistical reports, and degrees. Using such a system, the administration work in the student record office is almost fully automatized, and students can apply for examinations using computer facilities. The necessary software was developed at the Faculty of Electrical and Computer Engineering, and was later adapted to the needs of several other faculties of the University of Ljubljana. Therefore, the system described represents the basis for building an integral information system of higher education.

1. UVOD

Fakulteta za elektrotehniko in računalništvo (v nadaljevanju FER) je z 2500 študenti ena izmed večjih fakultet v sklopu ljubljanske univerze. Na njej poteka izobraževanje v okviru dveh vzgojno-izobraževalnih programov: elektrotehnike ter računalništva in informatike. Vsak od teh programov se deli še naprej na smeri. Tako je študij elektrotehnike razdeljen na smeri avtomatika, elektronika, močnostna elektrotehnika in telekomunikacije, študij računalništva pa obsega smeri računalniška logika in sistemi, programska oprema in informatika. V okviru nekaterih smeri je možna še nadaljnja delitev na posamezne izbirne skupine.

Ker je ljubljanska univerza izrazito decentralizirano organizirana, potekajo vse aktivnosti v zvezi s študijsko problematiko (vpis, sprejemni izpiti, izpitna evidenca, evidenca diplomantov, oblikovanje učnih načrtov in določitev izvajalcev za posamezne predmete) na fakultetah. Prav visoka stopnja avtonomije posameznih fakultet pa je eden izmed vzrokov, da vsi dosedanji poskusi izgradnje informacijskega sistema na nivoju celotne univerze [1,2] niso pripeljali do celovite rešitve.

Trajneje sta zaživeli le dve obdelavi:

računalniška obdelava prijav za vpis na višješolski in visokošolski študij (obrazec Prijava za vpis) in računalniška obdelava podatkov o dejanskem vpisu (obrazec SV-20 Vpisni list), ki potekata na centralnem računalniku VAX 8550. Pri obeh je poudarek na obdelavi velikih količin podatkov, ki so namenjeni predvsem zunanjim uporabnikom (n.pr. Zavodu za statistiko), medtem ko končni uporabniki na fakultetah od tega nimajo posebnih koristi.

Z namenom, da bi odpravili to pomanjkljivost, smo na Fakulteti za elektrotehniko in računalništvo razvili lasten informacijski sistem za področje študijske problematike, ki temelji na enotno zasnovani podatkovni bazi in programskem paketu, ki skoraj v celoti avtomatizira poslovanje študijskega sektorja. Informacijski sistem je zasnovan modularno in pokriva naslednja področja: vodenje evidence prijav za vpis, izvajanje sprejemnih izpitov, evidenco vpisnih podatkov, avtomatski zajem prijav za izpite in vodenje izpitne evidence, izvajanje raznih analiz uspešnosti študija ter vodenje evidence diplomskih in zaključnih nalog.

Za realizacijo podatkovne baze in potrebnih programov smo izbrali Clipper [3], tako da celoten informacijski sistem temelji na uporabi osebnih računalnikov. Trenutno so v študijskem sektorju instalirani štirje osebni računalniki, ki zaenkrat še niso povezani v mrežo, pač pa so podatki in obdelave porazdeljeni tako, da je možno vzdrževanje konsistentnega stanja podatkovne baze z vsakodnevnim prenosom nekaterih datotek med posameznimi računalniki preko vmesnika RS232. Instalacija ustrezne lokalne mreže je predvidena za naslednje leto.

V nadaljevanju podajamo najprej opis računalniških programov, ki so potrebni za delovanje celotnega sistema, nato je podrobneje opisan postopek prijavljanja za izpite preko računalnika, temu pa sledi opis podatkovne baze. Posamezne aplikacije uporabljajo tudi druge fakultete, tako da opisani sistem predstavlja osnovo za razvoj informacijskega sistema izobraževalne dejavnosti za celotno ljubljansko univerzo. Zato članek zaključujemo s predstavitvijo načrtov za nadaljnji razvoj visokošolskega informacijskega sistema, ki poteka v okviru skupnega evropskega projekta Tempus JEP 1852-91.

2. OPIS RAČUNALNIŠKIH PROGRAMOV

Paket programov, ki so potrebni za delovanje informacijskega sistema, je zasnovan modularno, tako da vsak modul predstavlja zaokroženo celoto, ki pokriva eno izmed področij informacijskega sistema.

2.1. Prijava za vpis

Podatki o kandidatih, ki so se prijavili za vpis na FER, se zajamejo na nivoju univerze v okviru že omenjene računalniške obdelave prijav za vpis, nato pa se prenesejo na osebni računalnik s pomočjo posebnega programa. Ta program tudi določi, kateri izmed kandidatov morajo opravljati diferencialne izpite zaradi neustrezne predhodne izobrazbe, in vzpostavi šifre končane srednje šole v skladu s šifrantom srednjih šol, ki ga uporablja FER. Preostali programi iz tega modula omogočajo izvajanje naslednjih aktivnosti:

- Naknadni zajem prijave za vpis: Naknadno se zajemajo prijave tistih kandidatov, ki so prenesli prijavo z druge fakultete ali pa so matično prijavljeni na drugi fakulteti, vendar želijo opravljati preizkus znanja tudi na FER.

- Dopolnitev podatkov s prijave za vpis: Za izvedbo sprejemnih izpitov na FER so potrebni tudi nekateri podatki, ki jih obrazec Prijava za vpis ne vsebuje, zato jih zajamemo naknadno. To so podatki o učnem uspehu v tretjem in četrtem letniku srednje šole in podatki o uvrstitvah na republiških tekmovanjih iz matematike, fizike in računalništva. Poleg tega pa lahko ročno nastavljamo indikatorje, ki označujejo, katere preizkuse znanja bo kandidat opravljal (diferencialni izpit, preizkus znanja za en visokošolski izobraževalni program, preizkus znanja za oba izobraževalna programa, ki se izvajata na FER).

- Brisanje prijave za vpis: Če kandidat

dvigne prijavo, se le-ta briše. Prav tako se brišejo prijave tistih kandidatov, ki niso opravili diferencialnih izpitov.

- Spiski kandidatov po različnih kriterijih: Izdelati je možno različne spiske prijavljenih kandidatov. Nekateri imajo vnaprej določeno fiksno obliko, obstaja pa tudi t.i. univerzalni program za spiske, s pomočjo katerega lahko uporabnik interaktivno specificira kriterije za izdelavo seznama in tako generira spiske poljubne oblike.

- Stanje prijav za vpis: Izpišejo se sumarni podatki o številu prijavljenih kandidatov glede na izbrano stopnjo študija, poklic po končani srednji šoli, leto zaključka srednje šole, kraj stalnega bivališča, vojaški rok itd.

- Izpis etiket z naslovi: Izpisati je moč naslove kandidatov za diferencialni izpit, naslove kandidatov za preizkus znanja in naslove srednjih šol.

2.2. Sprejemni izpiti

Programi iz tega modula nudijo računalniško podporo pri izvedbi sprejemnih izpitov v skladu s pravilnikom o sprejemnih izpitih na FER. Z njihovo pomočjo (in s primerno organizacijo dela) uspe FER izračunati in objaviti rezultate sprejemnih izpitov že nekaj ur po zaključku preizkusa znanja. Modul sestavljajo naslednji programi:

- Izdelava obrazcev za odgovore: Računalnik izpiše "obrazce", ki jih kandidati uporabljajo za označevanje pravilnih odgovorov na preizkusu znanja. Na vsako vprašanje je možnih 5 odgovorov. Kandidat označi pravilen odgovor tako, da na obrazcu prekriža ustrezno okence. Da bi zmanjšali možnost prepisovanja, dobijo kandidati različne permutacije nalog, zato se na obrazcu izpiše tudi šifra ustrezne permutacije. Istočasno z obrazcem za odgovore se izpiše tudi poseben formular, na katerega vsak kandidat pred preizkusom znanja vpiše svoje podatke (priimek in ime, datum rojstva, naslov), nato pa formular zalepi v posebno ovojnico. Tako obrazec za odgovore kot formular z osebnimi podatki kandidata sta označena s posebno 6-mestno kodo, ki jo generira računalnik in služi za zagotavljanje anonimnosti med pregledovanjem izdelkov in vnosom rezultatov v računalnik.

- Vnos rezultatov preizkusa znanja: Za vsak izdelek je treba zajeti njegovo 6-mestno kodo ter število pravilnih odgovorov, število napačnih odgovorov in število vprašanj brez odgovora. Da bi zmanjšali možnost napak pri vnosu, se vsi rezultati zajemajo dvakrat. Program vsakokrat preverja veljavnost kode izdelka in testira, ali se seštevek pravilnih in napačnih odgovorov ter vprašanj brez odgovora ujema s številom nalog. Pri drugem vnosu pa se izvrši tudi primerjava podatkov s prvim vnosom. Če se podatki ne ujemajo, jih mora operater vnesti še enkrat in potrditi njihovo pravilnost.

- Dešifriranje rezultatov preizkusa znanja: Ko so vsi rezultati zajeti, se odprejo kuverte z osebnimi podatki o posameznih kandidatih, formularji pa uredijo po abecednem vrstnem redu. Za dešifriranje rezultatov obstajata dva programa. Pri prvem je iniciator dialoga računalnik, ki izpisuje osebne podatke kandidatov po abecednem vrstnem redu, operater pa vpisuje pripadajoče 6-mestne kode. Pri drugem pa vodi dialog operater, ki vtipka priimek in ime kandidata ter kodo, pod katero je opravljal preizkus znanja. Praviloma se za dešifriranje uporablja prvi program, drugi služi le za popravljanje morebitnih napak.

- Izračun in izpis rezultatov sprejemnega izpita: Ta program najprej za vsakega študenta izračuna skupno število točk, upoštevajoč rezultat preizkusa znanja, uspeh v tretjem in četrtem letniku srednje šole ter morebitne uspehe na republiških tekmovanjih iz matematike, fizike in računalništva. Nato izpiše vrstni red kandidatov glede na skupno število točk. Če več kandidatov doseže enako število točk, je boljši tisti, ki je dosegel več točk na preizkusu znanja.

- Izpis etiket z naslovi: Izpisati je možno posebej nalepke z naslovi sprejetih kandidatov in posebej nalepke z naslovi tistih kandidatov, ki niso opravili sprejemnega izpita.

- Analize rezultatov sprejemnega izpita: Programi za analizo omogočajo primerjavo med številom prijavljenih in številom sprejetih kandidatov v odvisnosti od učnega uspeha, končane srednje šole in poklica. Poleg tega je možno izpisati obvestila srednjim šolam, v katerih se za vsako srednjo šolo izpiše seznam njenih kandidatov, za vsakega kandidata pa je prikazan izračun skupnega števila točk in njegova uvrstitev.

2.3. Vpis in spremljajoča dokumentacija

Programi iz tega modula omogočajo zajem podatkov o študentih ob vsakoletnem vpisu na fakulteto in izdelavo ustrezne spremljajoče dokumentacije (potrdilo o vpisu, matična knjiga študentov). Podatki se v celoti zajamejo na osebnem računalniku, nato pa se s posebnim programom predelajo v obliko, ki omogoča njihovo vključitev v datoteko vpisnih listov na centralnem računalniku. Modul Vpis in spremljajoča dokumentacija obsega naslednje programe:

- Zajem vpisnega lista ŠV-20: Program omogoča zajem podatkov z vpisnega lista ŠV-20, .ki ga v dogovoru z obema slovenskima univerzama predpisuje Zavod Republike Slovenije za statistiko. Zasnovan je tako, da pri prvem vpisu na fakulteto črpa podatke o študentu iz obrazca Prijava za vpis, operater pa vnaša samo tiste podatke, ki jih v prijavi ni. Pri nadaljnjih vpisih pa dobi operater na zaslon podatke iz preteklega leta in vnese samo tiste podatke, ki so se medtem spremenili. Program sproti preverja pravilnost vnešenih podatkov (kontrola šifrantov, razne logične kontrole) in opozarja operaterja na morebitne napake.

- Preverjanje potrdila o vpisu: Program na podlagi vpisne številke študenta izpiše podatke o vpisu (študijsko leto, letnik, smer, vrsta vpisa). Zaenkrat se uporablja samo za preverjanje, ali je študent pravilno izpolnil obrazec Potrdilo o vpisu, z minimalnimi spremembami pa bi lahko dosegli, da bi se to potrdilo izpisovalo avtomatsko.

- Izpis matične knjige: Vsaka fakulteta je v skladu s pravilnikom o pedagoški dokumentaciji in evidencah v usmerjenem izobraževanju [4] dolžna voditi matično knjigo svojih študentov. Pred uvedbo računalniške obdelave so referentke pisale to knjigo po več mesecev. Sedaj se ta knjiga izpiše avtomatsko na podlagi podatkov z obrazca SV-20.

- Spiski študentov po različnih kriterijih: Različne spiske študentov potrebujejo tako predavatelji kot različni zunanji uporabniki. Referent lahko izdela spiske študentov za vse letnike in smeri naenkrat, lahko pa (zopet s pomočjo t.i. univerzalnega programa za spiske) oblikuje spiske z izbrano vsebino, tako da interaktivno določi kriterije (n.pr. šolsko leto, letnik, smer, vrsta vpisa, občina stalnega bivališča, štipendija) in podatke (n.pr. vpisna številka, priimek in ime, naslov bivališča, naziv končane srednje šole, uspeh itd.) za svoj spisek.

- Stanje vpisa: Program izdela sumarni pregled števila vpisanih študentov po letnikih, smereh in izbirnih skupinah posebej za višješolski in posebej za visokošolski študij.

- Sklepi: Možen je vnos podatkov o sklepih (pohvalah, kaznih, izjemoma odobrenem vpisu ali spremembi smeri študija), ki so jih v zvezi z določenim študentom sprejeli ustrezni fakultetni organi. Za vsak sklep je treba vnesti datum sklepa, naziv organa, ki ga je sprejel, in njegovo besedilo.

- Izpis matičnih podatkov o študentu: Izpišejo se tisti podatki, ki so bili v ročnem načinu dela zabeleženi na kartotečnem listu (osebni podatki, podatki o vpisu v posameznih šolskih letih, podatki o sklepih).

- Izpis etiket z naslovi: Izpišejo se nalepke z naslovi redno vpisanih študentov za izbrano šolsko leto, letnik in smer študija.

2.4. Evidentiranje opravljenih študijskih obveznosti

Programi iz tega modula omogočajo vodenje izpitne evidence, izdelavo potrdil o opravljenih izpitih in preverjanje pogojev za

napredovanje. Programi temeljijo na predpostavki, da je na podlagi šifre izbirne skupine in letnika moč enolično določiti predmetnik za vsakega študenta. Modul obsega naslednje programe:

- Vnos podatkov s prijavnice: Vnos podatkov o izpitu poteka na podlagi prijavnice (obrazec Obvestilo o izpitu), potem ko je študent izpit že opravljal. Za vsako prijavnico je treba vnesti vpisno številko študenta, šifro predmeta, šifro predavatelja, oceno, datum izpita in podatke o številu polaganj. Običajno se vnašajo vse prijavnice za nek izpitni rok istočasno, zato je program zasnovan tako, da podatki o predmetu, predavatelju in datumu izpita ostanejo na zaslonu in je treba za nadaljnje prijavnice vnesti samo podatke o študentu, oceni in številu polaganj.

- Izpis potrdila o opravljenih izpitih: Izpiše se potrdilo, ki poleg seznama opravljenih izpitov vsebuje tudi sumarno informacijo o številu predpisanih in številu opravljenih izpitov za vsak letnik posebej. Če je študent opravil določene obveznosti izven programa, na katerega je trenutno vpisan, se pregled teh izpitov izpiše posebej.

- Pregled opravljenih izpitov: Ta program izdela pregled opravljenih študijskih obveznosti po posameznih šolskih letih. Namenjen je predvsem referentkam v študentski pisarni, za razliko od potrdila o opravljenih izpitih, ki je namenjeno predvsem študentom oziroma zunanjim uporabnikom. Oblika izpisa je zasnovana tako, da je kolikor se le da podobna kartotečnemu listu, ki so ga referentke uporabljale pred uvedbo računalniško vodene izpitne evidence. Iz izpisa je za vsako študijsko leto razviden predmetnik, ki ga je študent vpisal, pri tistih predmetih, kjer je že opravljal izpit, pa tudi podatki o izpraševalcu, datumu izpita, oceni in številu polaganj.

- Preverjanje pogojev za napredovanje: Ta program preverja tako pogoje za napredovanje v višji letnik kot pogoje za ponavljanje letnika. Student lahko napreduje v naslednji letnik, če opravi predpisano število ur obveznosti iz tekočega letnika ter vse obveznosti iz preteklih letnikov. Poleg tega pa program dopušča, da predpišemo tudi obvezne predmete, brez katerih napredovanje ni možno, četudi je doseženo predpisano število ur. Pogoj za ponavljanje letnika je prav tako izražen s številom ur opravljenih obveznosti iz tekočega letnika, ki pa je seveda nižje kot za napredovanje. Vsak študent lahko ponavlja samo enkrat v času študija. Program preverja pogoje za napredovanje za vse študente izbranega letnika in smeri naenkrat. Rezultat je seznam, v katerem se za vsakega študenta izpišejo podatki o opravljenem številu ur in številu manjkajočih ur študijskih obveznosti po posameznih letnikih ter podatka o številu ponavljanj in opravljenih obveznih predmetih.

2.5. Razne analize uspešnosti študija

Na podlagi podatkov o vpisu in izpitne evidence je možno izvajati razne analize uspešnosti študija:

- Prehodnost iz letnika v letnik: Za izbrano študijsko leto in smer študija je moč za vse letnike naenkrat izpisati podatke o številu študentov (absolutno in procentualno), ki so napredovali v višji letnik.

- Spremljanje napredovanja čiste in celotne generacije: Za izbrano generacijo se izpišejo podatki o napredovanju iz letnika v letnik tako v absolutnih številkah kot v procentih. Pri napredovanju čiste generacije se upoštevajo samo študenti, ki so redno opravljali študijske obveznosti, pri analizi napredovanja celotne generacije pa je možno spremljati tudi ponavljalce in pavzante.

- Kvaliteta študija: Pri tej analizi se za vsakega študenta izpišejo povprečne ocene po posameznih letnikih. Analizo je moč narediti za vse študente izbranega letnika, za vse študente, ki so končali študij v izbranem časovnem obdobju, za vse študente izbrane smeri ali pa samo za izbranega študenta.

- Izračun povprečne ocene po letnikih, smereh in izbirnih skupinah: Izpišejo se podatki o povprečnem številu polaganj, povprečni oceni in povprečni pozitivni oceni za vse letnike, smeri in izbirne

skupine.

- Polaganje izpitov: Izračunati je moč povprečno število polaganj, povprečno oceno in povprečno pozitivno oceno za izbran predmet ali za izbranega visokošolskega učitelja ter izpisati seznam študentov, ki še niso opravili izpita pri določenem predmetu.

- Analiza uspeha po prvem letniku: Analizo je moč izdelati na dva načina: po srednjih šolah ali po poklicu. V prvem primeru se za vsako srednjo šolo izpišejo podatki o številu študentov, ki so bili vpisani v prvi letnik, in številu študentov, ki so uspešno napredovali v drugi letnik. Poleg tega se izpiše njihova povprečna ocena v zadnjem letniku srednje šole in povprečna ocena v prvem letniku študija.

  Pri analizi po poklicu je možno najprej grupirati sorodne poklice v skupine in potem izdelati analizo po teh skupinah. Če želimo analizo za posebej izbrane poklice, oblikujemo skupine z enim samim poklicem. Tudi pri tej analizi se za vsako skupino poklicev izpiše število študentov, ki so bili vpisani v prvi letnik, število študentov, ki so uspešno napredovali v drugi letnik, ter njihova povprečna ocena v zadnjem letniku srednje šole in povprečna ocena v prvem letniku študija.

- Analiza ocene ob zaključku šolanja: Izpiše se število diplomantov, njihova povprečna ocena in povprečna dolžina študija. Analizo je moč izdelati glede na končano srednjo šolo, glede na uspeh na sprejemnem izpitu in glede na uspeh v srednji šoli.

- Rangiranje študentov: Za vsako šolsko leto je možno izdelati rang lestvico za izbrano podmnožico študentov. Ta podmnožica lahko obsega vse študente, ki so bili vpisani v izbranem šolskem letu, lahko pa se omejimo samo na študente izbranega letnika, smeri ali izbirne skupine.

2.6. Diplomske in zaključne naloge

Študent, ki opravi vse obveznosti iz študijskega programa, lahko zaprosi za izdajo teme za diplomsko nalogo (na visokošolskem študiju) oziroma zaključno nalogo (na višješolskem študiju). Postopek je v obeh primerih enak, razlika je le v zahtevnosti naloge in času, ki ga ima študent na voljo za njeno dokončanje. Diplomsko nalogo mora oddati v roku treh mesecev, zaključno nalogo pa v šestih tednih od dviga teme. Programi iz modula Diplomske in zaključne naloge omogočajo računalniško spremljanje vseh aktivnosti od prošnje za izdajo teme, preko odobritve teme in zagovora do vodenja knjige diplomantov, izdelave raznih spiskov in sumarnih pregledov. Tako za diplomske kot za zaključne naloge se uporabljajo isti programi, poseben parameter, ki se nastavi ob klicu vsake procedure pa pove, katero vrsto nalog obdelujemo.

- Prošnja za izstavitev teme: Študent odda prošnjo za izstavitev teme na posebnem obrazcu. Program omogoča zajem podatkov s tega obrazca, preveri, ali je študent opravil vse predpisane obveznosti iz študijskega programa in izračuna povprečno oceno izpitov in vaj. Na koncu program izpiše obvestilo profesorju, ki bo mentor pri izdelavi diplomske/zaključne naloge, s prošnjo, da v študentsko pisarno posreduje naslov in besedilo teme.

- Vnos naslova diplomske/zaključne naloge: Ko je tema odobrena in mentor posreduje naslov ter besedilo naloge, se naslov vnese v računalnik.

- Pregled izdanih tem: Za izbran datum dviga se izpiše seznam študentov, ki so tega dne dvignili temo, skupaj s podatki o mentorju, komentorju, naslovu teme in roku za oddajo.

- Priprava zapisnika o zagovoru: S tem programom v študentski pisarni pripravijo obrazec "Zapisnik o zagovoru" in ga posredujejo predsedniku komisije za zagovor diplomske oziroma zaključne naloge. Komisija v obrazec vnese še podatke o zagovoru (ocena naloge, ocena zagovora, skupna ocena študija).

- Vnos podatkov o zagovoru: na podlagi zapisnika se zajameta datum zagovora in skupna ocena študija.

- Izpis potrdila o diplomi: Program omogoča izpis potrdila o zaključku

študija, ki ga študent dobi po uspešnem zagovoru diplomske/zaključne naloge in s katerim uveljavlja vse pravice iz tega naslova vse do uradne podelitve ustrezne diplome.

- Spiski diplomantov po različnih kriterijih: Izpisati je možno knjigo diplomantov ter izdelati spisek diplomantov, ki izide v okviru publikacije "Objave o študiju na FER", in spisek diplomantov, ki ga FER posreduje na univerzo. Tudi v okviru tega modula obstaja t.i. univerzalni program za spiske, ki uporabniku omogoča izdelavo spiskov z različno vsebino.

- Analiza števila diplomantov: Za izbrano časovno obdobje je moč izpisati dva sumarna pregleda diplomantov: po mentorjih in po smereh študija.

2.7. Arhiviranje podatkov

Sčasoma se v podatkovni bazi naberejo podatki, ki za operativno delo niso več potrebni (n.pr. podatki o študentih, ki so že diplomirali in podatki o študentih, ki so prekinili študij). Da ti podatki ne bi po nepotrebnem obremenjevali sistema, so bili razviti programi za njihovo arhiviranje. Po drugi strani pa programi iz tega modula omogočajo tudi prenos podatkov za posamezne študente iz arhiva v operativno podatkovno bazo.

Programi so zasnovani tako, da se podatki, ki jih želimo arhivirati, najprej prenesejo iz operativne podatkovne baze v začasni arhiv na disku, nato pa se izvrši komprimiranje arhiva in prepis le-tega na diskete. Pri restavriranju arhiviranih podatkov pa je postopek obraten. Arhiv se najprej prepiše z disket na začasni poddirektorij na disku, od koder se potem izločijo potrebni podatki in prenesejo v operativno podatkovno bazo. Modul obsega naslednje programe:

- Arhiviranje na disk: Arhivirati je možno podatke o študentih, ki so končali visokošolski študij, podatke o študentih, ki so končali višješolski študij, ter podatke o študentih, ki so prekinili študij. V prvih dveh primerih zadostuje, da vnesemo datum diplome: arhivirajo se vsi podatki o študentih, ki so diplomirali pred izbranim datumom. Program sam preveri, ali se je diplomant kasneje vpisal na višjo stopnjo študija oziroma ali ima odobreno pravico do polaganja diferencialnih izpitov. V tem primeru se podatki ne arhivirajo.

  Pri arhiviranju podatkov o študentih, ki so prekinili študij, pa je kriterij za arhiviranje sestavljen iz treh podatkov: študijskega leta zadnjega vpisa, datuma, ko je bil opravljen zadnji izpit, in datuma veljavnosti izjemoma odobrene pravice do polaganja izpitov.

  Poleg tega lahko uporabnik izvede tudi arhiviranje podatkov za izbranega študenta, tako da enostavno vtipka njegovo vpisno številko.

- Prepis arhiva na diskete: Program izvrši komprimiranje začasnega arhiva na disku in ga prepiše na diskete.

- Restavriranje arhiva z disket: Arhiv se z disket prepiše na poseben poddirektorij na disku, nakar se izvrši "razpakiranje" in indeksiranje arhiviranih datotek.

- Restavriranje podatkov o izbranih študentih: Program zahteva, da uporabnik vnese vpisne številke tistih študentov, katerih podatke želi ponovno vključiti v operativno podatkovno bazo. Zahtevani podatki se nato izločijo iz začasnega arhiva na disku in vključijo v operativno podatkovno bazo. Program sam poskrbi za reindeksiranje tistih datotek, katerih vsebina se je spremenila.

- Izpis arhiviranih zapisov: Program omogoča, da po vsakem arhiviranju izpišemo seznam študentov, za katere so se arhivirali podatki.

2.8. Pomoč uporabniku

Programi iz tega modula olajšajo uporabniku iskanje raznih šifer, vpisnih številk za posamezne študente ipd. Njihovo izvajanje je moč sprožiti kadarkoli s pritiskom na tipko za pomoč <F1>. Modul sestavljajo naslednji programi:

- Pregled šifrantov
- Iskanje vpisne številke študenta
- Iskanje številke kandidata, ki se je prijavil za vpis
- Iskanje vpisne številke študenta, ki ima

izjemoma odobreno pravico do polaganja izpitov.

3. PRIJAVLJANJE ZA IZPITE PREKO RAČUNALNIKA

Osnovna verzija programskega paketa, ki smo jo opisali v prejšnjem razdelku, omogoča, da se vnos podatkov o izpitu izvrši šele takrat, ko predavatelj vrne v študijski sektor izpolnjeno prijavnico. Vse predhodne aktivnosti, kot n.pr. preverjanje pravice do polaganja, pa potekajo ročno in so prepuščene bodisi delavcem študijskega sektorja ali pa pedagogom.

Sprememba zakonodaje, ki omejuje število polaganj v posameznih izpitnih obdobjih (Zakon o spremembah in dopolnitvah zakona o usmerjenem izobraževanju [5]), in zahteva, da študenti, ki že več kot eno leto nimajo statusa, plačajo stroške izpita, sta narekovali, da se tudi ta del izpitnega postopka avtomatizira. Tako je nastal programski paket za avtomatski zajem izpitnih prijav [6], ki se na FER uporablja od 15.1.1991. Sestavljata ga dve skupini programov:

- programi, ki so namenjeni študentom
- programi, ki so namenjeni delavcem študijskega sektorja.

3.1. Programi, ki so namenjeni študentom

Študent, ki želi uporabljati programski paket, mora vtipkati svojo vpisno številko in geslo. Če je kombinacija vpisne številke in gesla pravilna, se na zaslonu pojavi menu, ki omogoča izvajanje naslednjih aktivnosti:

- Prijava za izpit
- Prijava za ustni izpit (če je študent pisni del izpita uspešno opravil s kolokviji)
- Brisanje prijave za izpit
- Sprememba gesla.

Pri prijavi za izpit (ne glede na to, ali gre za izpit v celoti ali samo za ustni del izpita) mora študent vnesti podatke o predmetu in datumu izpita, za katerega se prijavlja. Oba podatka izbere preko ustreznih menujev, ki mu jih posreduje računalnik. Prav tako računalnik sam določi podatke o izpraševalcu. Nato sledi preverjanje pravice do polaganja.

Študent lahko polaga izpite samo iz tistih predmetov, ki jih je vpisal. Vsak izpit lahko opravlja samo trikrat v istem šolskem letu. V vsakem izpitnem obdobju se lahko prijavi samo za en izpitni rok, komisijske izpite pa lahko opravlja samo v rednih izpitnih obdobjih. Študenti, ki že več kot eno leto niso vpisani, morajo izpit plačati. Če študent izpolnjuje vse pogoje za pristop k izpitu, se prijava zabeleži v računalniku, v nasprotnem primeru pa se prijava zavrne.

Poleg tega programski paket dopušča, da se nekateri študenti lahko prijavijo tudi za izpite izven programa, ki so ga vpisali. To so študenti, ki jim je odobreno polaganje izpitov vnaprej, študenti, ki opravljajo diferencialne izpite zaradi prehoda z višješolskega na visokošolski študij, študenti, ki so bili prevedeni na nov učni načrt, in občani, ki pridobivajo znanje iz posameznih predmetov kot samo-izobraževalci.

Preostala dva programa, ki sta na voljo študentom, sta bistveno enostavnejša. Program za odjavo izpita omogoča študentu, da briše prijavo za izpit, vendar samo v primeru, ko le-ta še ni bila posredovana fakultetnemu učitelju. Zasnovan je tako, da študent preko menuja izbere tisto prijavo, ki jo želi brisati. Program za spremembo gesla pa omogoča študentu, da spremeni svoje geslo, ki mu (skupaj z vpisno številko) omogoča uporabo programskega paketa. Vsak študent dobi geslo ob prvem vpisu na fakulteto, potem pa sam skrbi za njegovo tajnost in morebitne spremembe.

Vsi navedeni programi so zasnovani tako, da se posamezne transakcije (prijava in odjava izpita ter sprememba gesla) dodatno beležijo tudi na posebno datoteko, ki omogoča ponovno vzpostavitev konsistentnega stanja podatkovne baze v primeru izgube podatkov zaradi okvare, napake operaterja ipd.

3.2. Programi, ki so namenjeni delavcem študijskega sektorja

Programe, ki so namenjeni delavcem

študijskega sektorja, lahko razdelimo na več skupin:

- Programi za vzdrževanje podatkov o izpitnih rokih
- Programi za pregled podatkov o prijavah in izpis seznamov prijavljenih kandidatov
- Program za vnos ocen (rezultatov izpita)
- Programi za vzdrževanje podatkov o študentih, ki lahko polagajo izpite izven svojega učnega programa
- Programi za prenos podatkov med računalniki.

Poleg tega sta delavcem študijskega sektorja na voljo tudi programa za zajem in brisanje prijav, s pomočjo katerih lahko referent vnese ali briše prijavo tudi v tistih primerih, ko je študentom to onemogočeno.

Programi za vzdrževanje podatkov o izpitnih rokih omogočajo delavcem študijskega sektorja, da avtomatsko generirajo datume izpitnih rokov za celo šolsko leto, nato pa lahko posamezne roke brišejo, dodajo ali pa jim spremenijo datum.

Programi za pregled podatkov o prijavah omogočajo tako sumarni kot poimenski pregled prijav za posamezne izpite. Za vsak izpitni rok je možno izpisati seznam prijavljenih kandidatov, v katerega fakultetni učitelj vpiše ocene in ga vrne v študijski sektor.

Vnos ocen v računalnik poteka na podlagi seznama tako, da računalnik izpisuje na zaslon imena kandidatov v takem vrstnem redu, kot so izpisana v seznamu, referent pa vnaša ustrezne ocene. Če je kandidat odstopil od izpita ali še ni opravil ustnega dela, ga je moč pri vnosu ocen enostavno preskočiti.

Programi za vzdrževanje podatkov o študentih, ki lahko polagajo izpite izven svojega učnega programa, omogočajo vnos in brisanje pravice do polaganja izpitov vnaprej, polaganja diferencialnih izpitov, polaganja posameznih izpitov zaradi prevedbe na nov učni načrt in polaganja izpitov za občane samoizobraževalce.

Da bi študentom preprečili nepooblaščen dostop do izpitne evidence, poteka zajem prijav na samostojnih osebnih računalnikih, programi za prenos pa omogočajo, da se podatki o prijavah, izpitih in pravici do polaganja prenesejo z računalnikov za študente v študijski sektor in obratno. Prenos lahko poteka preko disket ali preko vmesnika RS232.

4. OPIS PODATKOVNE BAZE

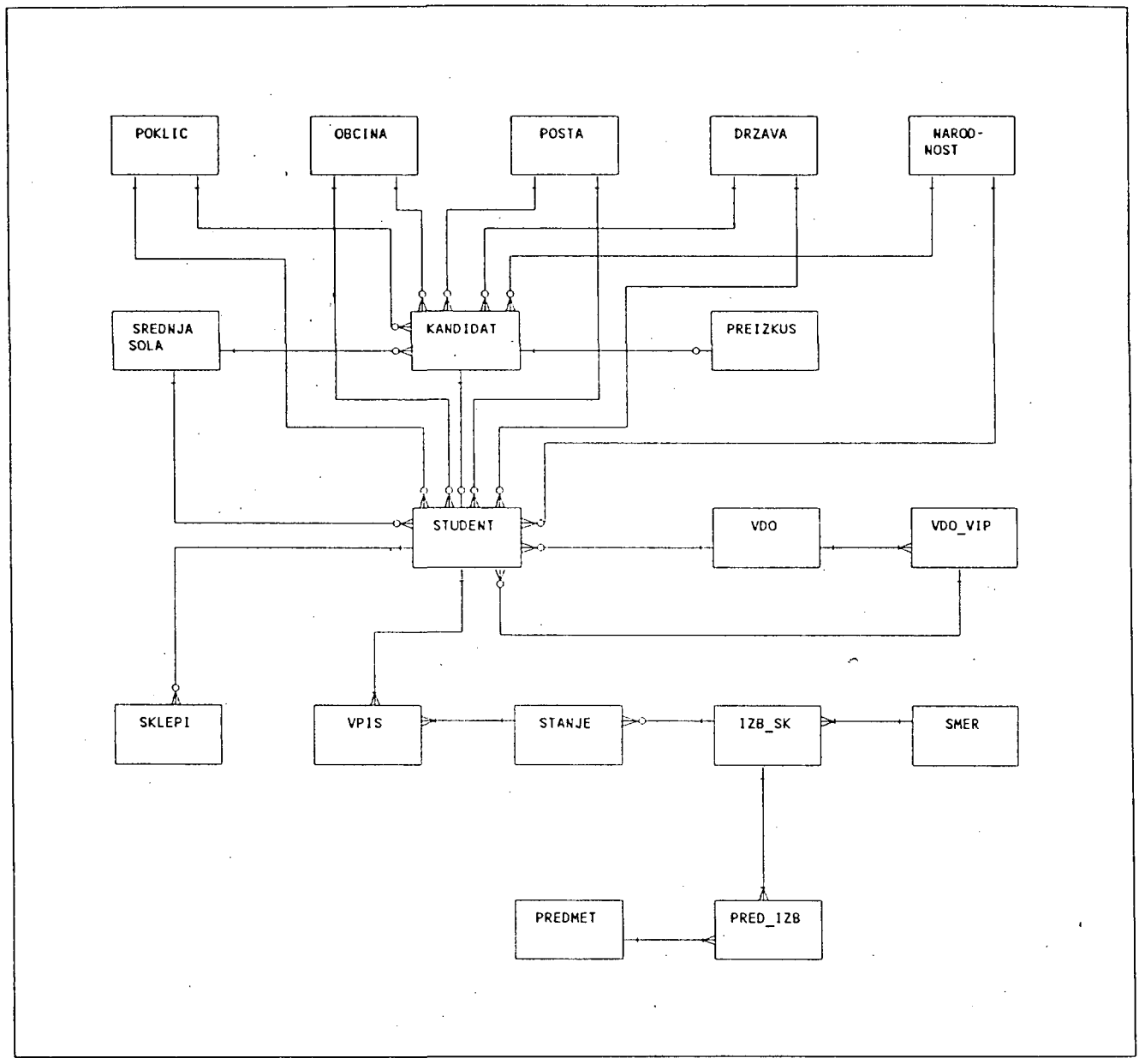
Informacijski sistem izobraževalne dejavnosti na FER temelji na enotni podatkovni bazi, katere opis podajamo v obliki ustreznega relacijskega podatkovnega modela [7, 10], ki prikazuje tako posamezne relacije kot razmerja med njimi. Zaradi boljše preglednosti smo opis razdelili na tri logično zaokrožene enote. Tako najprej opisujemo tisti del podatkovne baze, ki pokriva področja prijave za vpis, sprejemnih izpitov in samega vpisa (slika 1), nato opisujemo relacije za področje izpitne evidence in avtomatskega zajema izpitnih prijav (slika 2), nazadnje pa še podatkovni model za področje diplomskih in zaključnih nalog (slika 3).

4.1. Prijava za vpis, sprejemni izpiti, vpis

Osrednje relacije v podatkovnem modelu, ki pokriva področja prijave za vpis, sprejemnih izpitov in samega vpisa, so KANDIDAT, STUDENT in VPIS. Relacija KANDIDAT vsebuje podatke o kandidatih, ki so se prijavili za vpis na višješolski ali visokošolski študij. Vsakemu kandidatu ustreza en zapis, ki hrani podatke z obrazca Prijava za vpis.

Relacija STUDENT vsebuje matične podatke o študentih, ki so vpisani na fakulteto. Vsakemu študentu ustreza en zapis, ki se kreira ob prvem vpisu, kasneje pa se samo ažurirajo morebitne spremembe (n.pr. naslov stalnega in začasnega bivališča, materialni podatki ipd.). Zajem matičnih podatkov poteka ob vsakem vpisu na podlagi obrazca SV-20 Vpisni list. Pri prvem vpisu se ti podatki vnesejo v celoti, kasneje pa se samo ažurirajo morebitne spremembe.

Relacija VPIS hrani podatke o posameznih vpisih. Ob vsakem vpisu se za vsakega študenta generira nov zapis, ki vsebuje naslednje podatke: vpisna številka študenta,



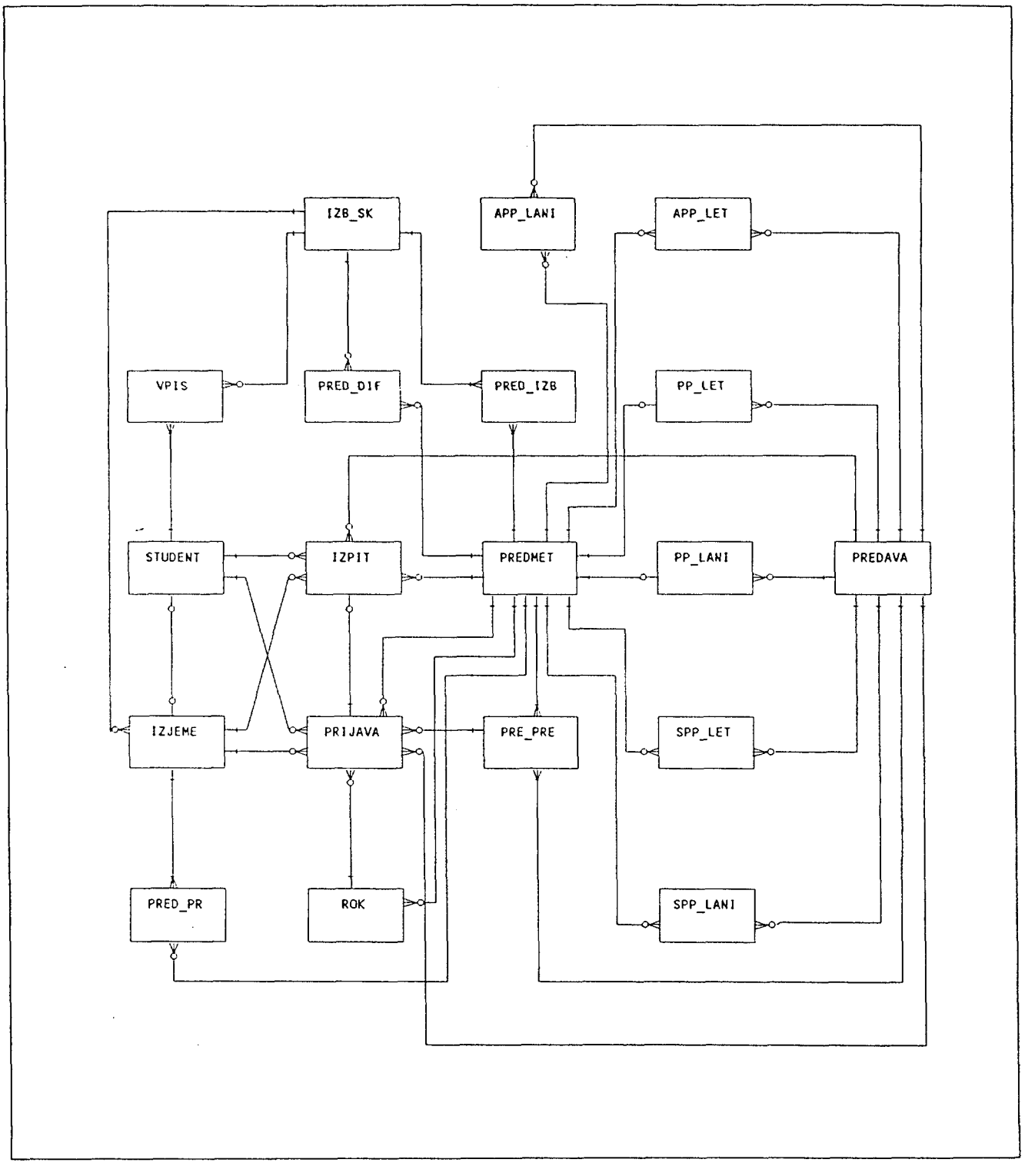
Slika 1: Podatkovni model za prijavo za vpis, sprejemne izpite in vpis

šolsko leto, letnik, smer in izbirna skupina, stopnja študija, vrsta vpisa in način študija.

Relacija PREIZKUS vsebuje rezultate preizkusa znanja. Vsakemu kandidatu ustreza en zapis, ker pa poteka preizkus anonimno, se kot primarni ključ za dostop do posameznih zapisov uporablja posebna koda kandidata.

Veliko podatkov z obrazcev Prijava za vpis in Vpisni list je treba šifrirati. Relacije POKLIC, OBCINA, POSTA, DRZAVA, NARODNOST, SREDNJA SOLA, VDO in VDO_VIP vsebujejo ustrezne šifrante poklicev, občin, pošt, držav, narodnosti, srednjih šol, visokošolskih organizacij in vzgojno-izobraževalnih programov po posameznih visokošolskih organizacijah.

Relaciji SMER in IZB_SK vsebujeta šifrant smeri in izbirnih skupin na tisti fakulteti, kjer je program instaliran. Relacija SMER vsebuje samo osnovno razdelitev študija na smeri, v okviru relacije IZB_SK pa je ta razdelitev bolj podrobna in upošteva tudi posamezne izbirne skupine ter različne variante učnega načrta.



Slika 2: Podatkovni model za izpitno evidenco in avtomatski zajem izpitnih prijav

Relacija STANJE vsebuje za vsako šolsko leto in vsak letnik šifre izbirnih skupin, ki se tisto leto izvajajo. Relacija SKLEPI pa vsebuje dodatne informacije o posameznih študentih, katerih oblike ni moč vnaprej formalizirati. Sem se vpisujejo tisti podatki, ki so se v ročnem načinu vodenja študentske kartoteke beležili v rubriko Opombe na kartotečnem listu, n.pr. sklepi o pohvalah, kaznih, izjemoma odobrenem vpisu, dovoljenju za polaganje izpitov ipd.

Predmetnik vsakega študenta je določen s pomočjo relacij VPIS, PREDMET, IZB_SK in PRED_IZB. Relacija PREDMET vsebuje seznam vseh predmetov, ki se predavajo na fakulteti, relacija PRED_IZB pa predstavlja povezavo med relacijami VPIS, PREDMET in IZB_SK ter omogoča, da na podlagi letnika študija in šifre izbirne skupine določimo pripadajoče predmete.

4.2. Izpitna evidenca in avtomatski zajem izpitnih prijav

Poleg relacij STUDENT, VPIS, IZB_SK, PRED_IZB in PREDMET, ki smo jih spoznali že v prejšnjem podpoglavju, nastopajo v podatkovnem modelu za področje izpitne evidence še relacije, ki vsebujejo podatke o razpisanih izpitnih rokih, prijavljenih kandidatih, opravljenih izpitih, predavateljih posameznih predmetov in študentih, ki imajo izjemoma odobreno pravico do polaganja izpitov izven učnega programa, ki so ga vpisali.

Relacije IZPIT, PREDAVA in PRE_PRE so potrebne ne glede na to, ali zajemamo prijave za izpit preko računalnika ali pa vnašamo prijavnice naknadno, potem ko je izpitni rok že mimo. Relacija IZPIT hrani podatke o dosedanjih polaganjih izpitov pri posameznih predmetih. Primarni ključ vsakega zapisa je sestavljen iz vpisne številke študenta in šifre predmeta. Hranimo samo podatke o zadnjem polaganju: šifri predavatelja in soizvajalca, oceni izpita in vaj, datum zadnjega polaganja, skupno število polaganj in število polaganj v tekočem šolskem letu. Relacija PREDAVA vsebuje register predavateljev, relacija PRE_PRE pa omogoča, da za vsakega predavatelja določimo vse predmete, ki jih le-ta predava, in obratno.

Preostale relacije se nanašajo na avtomatski zajem izpitnih prijav. Tako relacija ROK vsebuje podatke o razpisanih izpitnih rokih (datum izpita in šifro predmeta), relacija PRIJAVA pa podatke o prijavljenih kandidatih (vpisno številko, datum izpita, šifro predmeta, šifro predavatelja itd.).

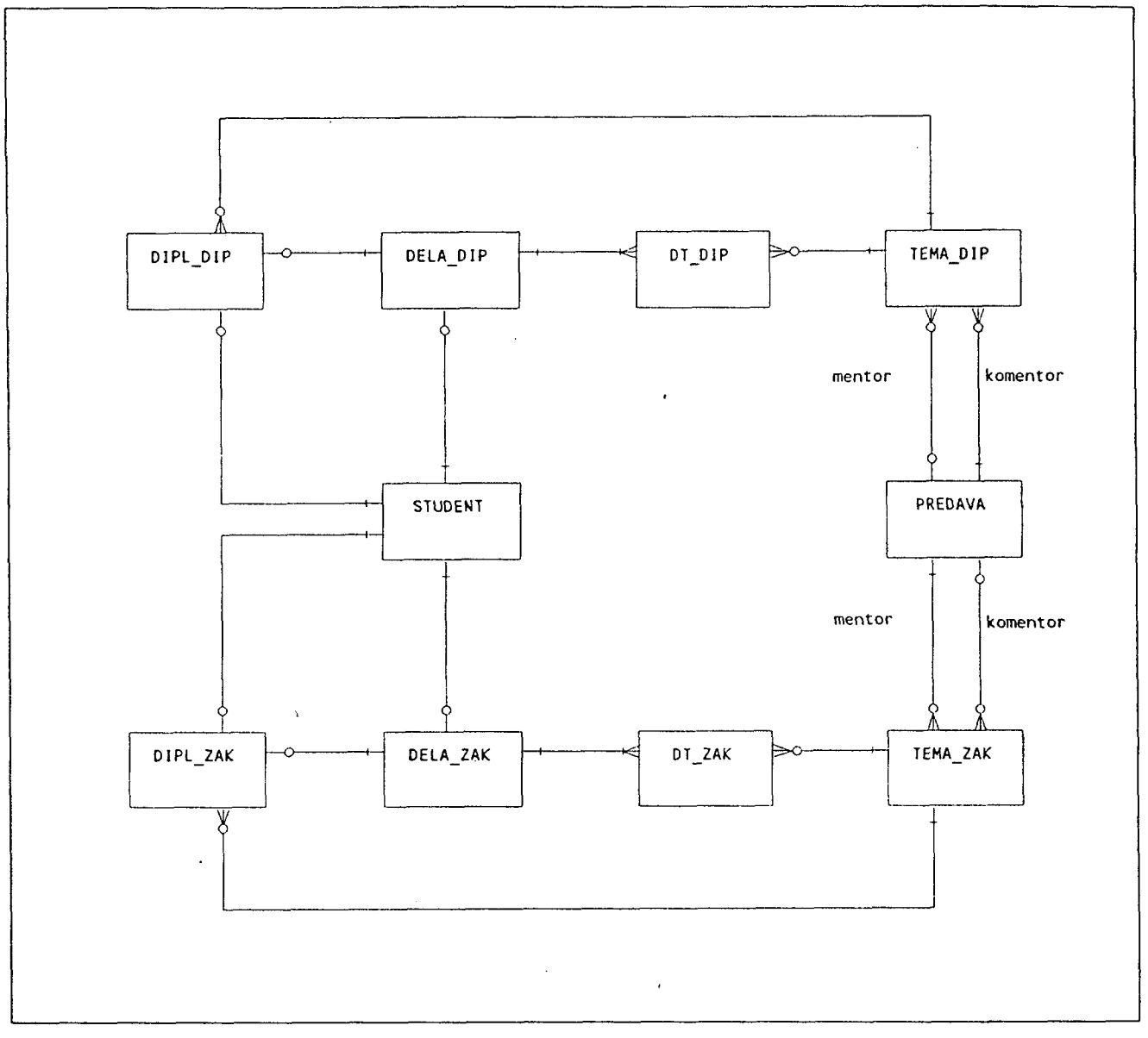
Relacija IZJEME vsebuje podatke o študentih, ki imajo izjemoma odobreno pravico do polaganja (polaganje vnprej, diferencialni izpiti za vpis na visokošolski študij, prevedba na nov učni načrt, občani samoizobraževalci).

Podatki o diferencialnih izpitih pri prehodu z višješolskega na visokošolski študij so shranjeni v okviru relacije PRED_DIF. Ti podatki so odvisni od smeri, na katero je bil študent vpisan na višješolskem študiju in niso vezani na posameznega študenta. Pač pa je treba za vsakega študenta, ki je preveden na nov učni program, posebej hraniti seznam predmetov, pri katerih mora še opraviti izpit. Podatke o teh predmetih vsebuje relacija PRED_PR.

Relacije PP_LET, PP_LANI, APP_LET, APP_LANI, SPP_LET in SPP_LANI omogočajo, da študentu med vnosom prijave za izpit ni treba vtipkati podatkov o izpraševalcu (predavatelju in soizvajalcu), ampak se le-ti določijo avtomatsko v skladu z dogovorom, da mora študent opraviti izpit pri tistem učitelju, ki je nazadnje predaval predmet, za katerga se študent prijavlja. To pomeni, da mora sistem vzdrževati podatke o izvajalcih posameznih predmetov za zadnji dve šolski leti: zgoraj navedene relacije s končnico _LET vsebujejo podatke o izvajalcih v letošnjem (tekočem) šolskem letu, relacije s končnico _LANI pa podatke o izvajalcih v lanskem (preteklem) šolskem letu.

Če predavanja v letošnjem šolskem letu še niso končana, sistem avtomatsko upošteva podatke za lansko šolsko leto, v nasprotnem primeru pa se upoštevajo podatki o izvajalcih v tekočem šolskem letu.

Razdelitev podatkov na tri relacije s predponami PP, APP in SPP je potrebna zaradi različnega načina izvajanja predavanj pri posameznih predmetih. Najbolj običajno je, da vsem študentom predava isti predavatelj: v tem primeru so podatki o izvajalcu zapisani v relaciji PP_LET oziroma PP_LANI. Če pa je študentov preveč za eno samo skupino predavanj, potekajo predavanja v več skupinah, vsaki skupini pa lahko predava drug predavatelj. Skupine se lahko formirajo tako, da se študenti razdelijo po abecedi (v tem primeru so podatki o predavatelju zapisani na APP_LET oziroma APP_LANI) ali pa po smereh, na katere so vpisani (v tem primeru uporabimo relaciji SPP_LET in SPP_LANI).



Slika 3: Podatkovni model za evidenco diplomskih in zaključnih nalog

4.3. Diplomske in zaključne naloge

Relaciji STUDENT in PREDAVA že poznamo: prva vsebuje matične podatke o študentih, druga pa register fakultetnih učiteljev. Preostale relacije lahko razdelimo na dve skupini: prva skupina je namenjena za vodenje evidence diplomskih nalog (zato imajo imena teh relacij končnico _DIP), druga skupina pa služi za vodenje evidence zaključnih nalog (relacije s končnico _ZAK).

Relaciji TEMA_DIP in TEMA_ZAK vsebujeta podatke o temah diplomskih oziroma zaključnih nalog. Vsaka tema ima svojo številko, naslov ter podatke o mentorju in komentorju. Zato ima povezava med relacijama TEMA_DIP in PREDAVA dva pomena: nek učitelj lahko pri nekaterih temah nastopa kot mentor, pri drugih pa kot komentor. Isto velja za relaciji TEMA_ZAK in PREDAVA.

Ko študent dvigne temo za diplomsko/zaključno nalogo, se podatki o dvigu zabeležijo v okviru relacije DELA_DIP oziroma DELA_ZAK. Zaradi splošnosti smo predpostavili, da lahko študentu odobrimo več tem (če n.pr. prve teme ni uspešno zaključil, se mu kasneje odobri nova), prav tako pa lahko isto temo

obdeluje več študentov. To vodi do razmerja M:N med relacijama DELA_DIP (DELA_ZAK) in TEMA_DIP (TEMA_ZAK), ki smo ga odpravili z uvedbo relacij DT_DIP oziroma DT_ZAK.

Ko študent uspešno zagovarja diplomsko (zaključno) nalogo, se podatki zabeležijo v okviru relacije DIPL_DIP oziroma DIPL_ZAK. Vsakemu študentu, ki konča študij, pripada en zapis na eni od teh relacij.

5. NADALJNJI RAZVOJ

Uspešna uvedba računalniške obdelave študentske kartoteke na Fakulteti za elektrotehniko in računalništvo je vzbudila zanimanje tudi pri drugih članicah ljubljanske univerze. Prve štiri module programskega paketa, ki ga opisujemo v tem članku, je odkupil Računalniški center univerze in jih prilagodil za potrebe nekaterih fakultet. Tako danes že na večini fakultet poteka obdelava prijav za vpis in vpisnih podatkov s pomočjo tega paketa, na nekaterih fakultetah pa so prešli tudi na računalniško vodenje izpitne evidence. S tem je nastala osnova za izgradnjo celovitega informacijskega sistema, ki bi pokrival področje študijske problematike na ljubljanski univerzi.

Žal pa je med prilagajanjem programov različnim zahtevam posameznih fakultet nastalo več inačic opisane programske opreme, kar otežuje vzdrževanje programov in njihovo medsebojno povezovanje v enoten informacijski sistem na nivoju univerze.

Zato se je - podobno kot tudi v nekaterih razvitih državah (glej n. pr. Frackmann [8]) - pojavila potreba po reintegraciji obstoječih obdelav v enoten sistem, ki bo temeljil na dosedanjih rezultatih in izkušnjah Fakultete za elektrotehniko in računalništvo, obenem pa bo vključeval tudi potrebe ostalih fakultet in zahteve zunanjih uporabnikov (Zavod za statistiko, Ministrstvo za šolstvo in šport itd.). V ta namen je bila formirana delovna skupina, ki jo sestavljajo sodelavci Fakultete za elektrotehniko in računalništvo, Centra za razvoj univerze in Računalniškega centra univerze, delo pa poteka v okviru skupnega evropskega projekta, ki ga sofinancira Evropska skupnost s sredstvi iz programa Tempus.

Zaradi decentralizirane organiziranosti ljubljanske univerze je delovna skupina predlagala dvonivojsko strukturo bodočega informacijskega sistema [9]. Spodnji nivo naj ustreza potrebam posameznih fakultet, kjer nastaja in se obdeluje večina podatkov, gornji nivo pa naj predstavlja sintezo podatkov, ki sicer nastajajo na fakultetah, a so potrebni za delovanje univerze kot celote.

Podatkovna baza in programi, ki smo jih opisali v tem članku, predstavljajo primerno osnovo za realizacijo spodnjega nivoja. Predlagamo, da realizacija tega nivoja temelji na uporabi osebnih računalnikov, ki - povezani v ustrezno mrežo - predstavljajo primerno osnovo za postopno dograjevanje informacijskih sistemov posameznih fakultet z ostalimi obdelavami (n.pr. kadrovska evidenca, računovodstvo in finance). Za realizacijo gornjega nivoja pa predlagamo centralni računalnik.

ZAHVALA

Za realizacijo informacijskega sistema izobraževalne dejavnosti na Fakulteti za elektrotehniko in računalništvo ima nedvomno veliko zaslug vodstvo fakultete, ki je ves čas zagotavljalo ustrezno organizacijsko in finančno podporo. Prav tako sistem ne bi zaživel v praksi, če ne bi naletel na podporo pri končnih uporabnikih, za kar ima največ zaslug vodja študijskega sektorja Silva Lavrenčič. Velik del programerskega dela so opravili tudi Uroš Marušič, Mitja Bensa in Igor Skraba. Vsem se za njihov prispevek najlepše zahvaljujem.

LITERATURA:

[1] B. Mihevc et al.: Visokošolski informacijski sistem. Izhodišča, oris stanja in zasnova novega sistema, Center za razvoj univerze, Ljubljana, 1979.

[2] A. Kopar: Integralni visokošolski informacijski sistem, Predštudija, Center za razvoj univerze, Ljubljana, 1984.

[3] E. Tiley: Using . Clipper, Que Corporation, Carmel, Indiana, 1988.

[4] Usmerjeno izobraževanje, zakon s komentarjem in izvršilnimi predpisi ter smernicami za oblikovanje vzgojnoizobraževalnih programov, Časopisni zavod Uradni list SR Slovenije, Ljubljana, 1981.

[5] Zakon o spremembah in dopolnitvah zakona o usmerjenem izobraževanju, Uradni list SRS št. 25, 12. julij 1989.

[6] V. Mahnič: Uporaba računalnika za zajem in obdelavo izpitnih prijav, Zbornik del XV. simpozija o informacijskih tehnologijah "Sarajevo – Jahorina 1991", marec 1991.

[7] J. Martin: Information Engineering, Savant Institute, 1986.

[8] E. Frackmann: Information for institutional administration and management in German higher education, CRE-action (Quarterly published by the Standing Conference of Rectors), 1991/3.

[9] B. Vilfan, V. Mahnič, B. Drobež, Z. Lapajne: Visokošolski informacijski sistem - študentska in izpitna evidenca, Univerza v Ljubljani, Center za Razvoj Univerze in Fakulteta za elektrotehniko in računalništvo, Ljubljana, februar 1992.

[10] POSE-DMD Data Model Diagrammer, Reference Guide, CSA Research Pte Ltd., Singapore, 1991.

AN INFORMATIONAL APPROACH OF BEING-THERE AS UNDERSTANDING II*

INFORMATICA 2/92

Keywords: Heideggerian Being-there information, informational formulas, philosophy, text formalization, understanding

Anton P. Železnikar
Volaričeva ulica 8
61111 Ljubljana

This text is a continuation of the essay pertaining to the formalization of texts and, in particular, to Heideggerian philosophy of Being-there as understanding [BT, §31]. To the text paragraphs 1 to 15 in the previous part, in this continuation, paragraphs 16 to 18 are formalized in a sentence by sentence fashion. In Section 3 of this continuation, paragraphs as a whole are treated. First, the informational connection among sentences is explained via the metaphysical link between the equally marked operand and operator entities in different formula systems of text sentences. This type of informational connectedness are supplementary formulas of the kind $\alpha \models \alpha$, $\alpha \models \varphi(\alpha)$, etc., where α's on the left and the right side of operator $\models$ belong to different sentence formulas. Then, all 18 paragraphs are presented by 18 formula systems together with the Greek operand, Fraktur operand, and operator lists. These lists include data on formal entities appearances in particular sentence formulas. Lastly, paragraph systems can be composed into a unique formula system depicting the so-called initial informational program of §)31 in [BT]. The problem which pertains to the same operands appearing on different places in informational formulas is discussed exhaustively and in a formalized way in Section 5.1. At the end of the essay a dictionary of used operands and operators is appended, however, in this part of the essay, only the Greek operands are listed.

The presented formula system of the text of §31 [BT] is in no way a final decision. Improvements can be made through considering the original German text [SZ] and also translations of this text into other languages, for instance, [BV].

Informacijski pristop k biti-tu kot razumevanju II*

To besedilo je nadaljevanje spisa, ki obravnava formalizacijo besedil in še posebej, Heideggrovo filozofijo biti-tu kot razumevanje [BT, §31]. V tem delu so k prejšnjim odstavkom 1-15 formalizirani še odstavki 16-18 poglavja 31, in sicer postopno, stavek za stavkom. V poglavju 3 tega spisa so prikazani odstavki besedila kot formalizirane celote. Najprej se pojasnjujejo informacijske povezave med stavki s pomočjo metafizičnih oblik med enako označenimi opernadnimi in operatorskimi entitetami, ki nastopajo v različnih formulskih sistemih stavkov besedila. Ta vrsta informacijske povezanosti so dodatne formule oblike $\alpha \models \alpha$, $\alpha \models \varphi(\alpha$, itd., kjer entitete, označene z α, na levi in desni strani operatorja $\models$ pripadajo različnim stavčnim formulam. Nato je vseh 18 odstavkov predstavljenih z 18 formulskimi sistemi, skupaj s seznami grških in frakturnih operandov ter operatorjev. Ti seznami vključujejo podatke o pojavitvah formalnih entitet v posameznih stavčnih formulah. Naposled so odstavčni sistemi združeni v edinstveni informacijski sistem, ki je upodobitev t.i. začetnega informacijskega programa za besedilo §31 v [BT]. Problem pojavljanja enih in istih operandov na različnih mestih v informacijskih formulah je obravnavan izčrpno in na formaliziran način v odstavku 5.1. Na koncu spisa je dodan slovar uporabljenih operandov in operatorjev, vendar v tem delu le za grške operande.

Opisani formulski sistem besedila 18531 [BT] nikakor ni končna verzija. Izboljšave sistema so mogoče ob upoštevanju izvirnega nemškega besedila [SZ] pa tudi prevodov besedila v druge jezike, npr. [BV].

2.16. *THE SIXTEENTH PARAGRAPH OF § 31 [BT]*

[16.1] The disclosedness of the "there" [$\vartheta_{disclose}(\tau_{there})$] in understanding is itself a way of Dasein's potentiality-for-Being [$\mathfrak{W}_{way}(\pi_{for\text{-}Being}(\mathfrak{D}))$].

Informational formula for this sentence is, for instance,

(16.1) $(\vartheta_{disclose}(\tau_{there}) \subset \mathfrak{U}) \models (\vartheta_{disclose}(\tau_{there}) \models \mathfrak{W}_{way}(\pi_{for\text{-}Being}(\mathfrak{D})))$ □

[16.2] In the way [$\subset \mathfrak{W}_{way}$] in which its Being [$\mathfrak{B}(\vartheta_{disclose})$] is projected both upon [$\models_{project} \circ \models_{upon}$] the "for-the- sake-of-which" [φ_{sake}] and upon significance (the world) [$\sigma_{sign}(\mathfrak{W}_{world})$], there lies [$\subset_{lie}$] the disclosedness of Being in general [$\vartheta_{disclose}(\mathfrak{B}) \models_{general}$].

For this sentence the following formula can be posited:

(16.2) $(\vartheta_{disclose}(\mathfrak{B}) \models_{general}) \subset_{lie} ((\mathfrak{B}(\vartheta_{disclose}) \models_{project} \circ \models_{upon} (\varphi_{sake}, \sigma_{sign}(\mathfrak{W}_{world}))) \subset \mathfrak{W}_{way})$ □

[16.3] Understanding of Being [$\mathfrak{U}(\mathfrak{B})$] has already been taken [$\models_{have} \circ ((\models_{already} \circ \models_{been}) \circ \models_{take})$] for granted [$\mathfrak{G}_{grant}$] in projecting [$\mathfrak{P}_{project}$] upon [$\models_{upon}$] possibilities.

A formula with nested operator composition for this sentence is

(16.3) $\mathfrak{U}(\mathfrak{B}) \models_{have} \circ ((\models_{already} \circ \models_{been}) \circ \models_{take}) ((\mathfrak{G}_{grant} \subset \mathfrak{P}_{project}) \models_{upon} \pi)$ □

[16.4] In projection [$\subset \pi_{project}$], Being is understood [$\models_{\mathfrak{U}} \mathfrak{B}$], though [$\models_{though}$] not ontologically conceived [$\not\models_{onto} \circ \models_{conceive} \mathfrak{B}$].

The straightforward formula for this sentence is

(16.4) $((\models_{\mathfrak{U}} \mathfrak{B}) \subset \pi_{project}) \models_{though} (\not\models_{onto} \circ \models_{conceive} \mathfrak{B})$ □

[16.5] An entity [α] whose kind of Being [$\mathfrak{K}(\mathfrak{B}(\alpha))$] is the essential projection of Being-in-the-world [$\models_{essen} \pi_{project}(\mathfrak{B}_{in\text{-}the\text{-}world})$] [implies] has [$\models_{have}$] understanding of Being [$\mathfrak{U}(\mathfrak{B})$], and has this as constitutive for its Being [$\models_{const} \mathfrak{B}(\alpha)$].

One can posit the following formula in accord with the last sentence:

(16.5) $(\mathfrak{K}(\mathfrak{B}(\alpha)) \models_{essen} \pi_{project}(\mathfrak{B}_{in\text{-}the\text{-}world})) \Rightarrow (\alpha \models_{have} (\mathfrak{U}(\mathfrak{B}) \models_{const} \mathfrak{B}(\alpha)))$ □

[16.6] What [$\mathfrak{W}_{what}$] was posited dogmatically [$\models_{posit} \circ \models_{dogma}$] at an earlier stage [$\sigma_{stage}(\varepsilon_{early})$] now gets [$\models_{now} \circ \models_{get}$] exhibited [$\varepsilon_{exhibit}$] in terms of the Constitution of the Being [$\mathfrak{C}_{const}(\mathfrak{B})$] in which Dasein as understanding is its "there" [$\tau_{there}(\mathfrak{C}_{const}(\mathfrak{B}))$].

The following system of two formulas can arise as an informational adequateness to the last sentence:

(16.6) $(\sigma_{stage}(\varepsilon_{early}) \models_{posit} \circ \models_{dogma} \mathfrak{W}_{what}) \models_{now} \circ \models_{get} \varepsilon_{exhibit}(\mathfrak{C}_{const}(\mathfrak{B}))$; $((\mathfrak{D} \subset \mathfrak{C}_{const}(\mathfrak{B})) \models_{as} \mathfrak{U}) \models \tau_{there}(\mathfrak{C}_{const}(\mathfrak{B}))$ □

[16.7] The existential meaning [μ_{exist}] of this understanding of Being [$\mathfrak{U}(\mathfrak{B})$] cannot be satisfactorily clarified within [$\not\models_{satisfy} \circ \subset_{clarify}$] the limits [$\lambda_{limit}$] of this [$\varphi_{(1.1)}, \varphi_{(1.2)}, \cdots, \varphi_{(18.1)}$] investigation [$\iota_{investigate}$] except on the basis of the Temporal Interpretation of Being [$\mathfrak{I}_{inter}(\mathfrak{T}_{temporal}(\mathfrak{B}))$].

This sentence is a system pertaining to the entire investigation of Being-there as understanding; so,

(16.7) $\mathfrak{I}_{inter}(\mathfrak{T}_{temporal}(\mathfrak{B})) \models_{clarify} (\mu_{exist} \models_{of} \mathfrak{U}(\mathfrak{B}))$; $(\mu_{exist} \models_{of} \mathfrak{U}(\mathfrak{B})) \not\models_{satisfy} \circ \subset_{clarify} \lambda_{limit}(\iota_{investigate})$; $\varphi_{(1.1)}, \varphi_{(1.2)}, \cdots, \varphi_{(18.1)} \models \iota_{investigate}$ □

2.17. *THE SEVENTEENTH PARAGRAPH OF § 31 [BT]*

[17.1] As *existentialia* [$\models_{as} \mu_{multi}(\varepsilon_{exist})$], states-of-mind [$\mu_{multi}(\mathfrak{S}_{mind})$] and understanding characterize [$\models_{char}$] the primordial disclosedness of Being-in-the-world [$\pi_{prim}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world}))$].

A simple formula for this sentence is

(17.1) $((\mu_{multi}(\mathfrak{S}_{mind}), \mathfrak{U} \models_{as} \mu_{multi}(\varepsilon_{exist})) \models_{char} \pi_{prim}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world}))$

Entity $\mu_{multi}(\mathfrak{S}_{mind})$ means a multiplicity of minds, that is, for instance, $\mathfrak{S}_{mind\_1}, \mathfrak{S}_{mind\_2}, \ldots$. Similar is meant by $\mu_{multi}(\varepsilon_{exist})$ where existentialia come into the game. □

[17.2] By way of having a mood [$\mathfrak{W}_{way}(\mathfrak{M})$], Dasein 'sees' [$\models_{see}$] possibilities, in terms of which it is.

The phenomenalizing formula for this sentence is

(17.2) $(\mathfrak{W}_{way}(\mathfrak{M}) \models \mathfrak{D}) \models_{see} \pi(\mathfrak{D} \models; \models \mathfrak{D})$ □

[17.3] In the projective disclosure of such possibilities [$\pi_{project}(\vartheta_{disclose}(\pi(\mathfrak{D})))$], it already has [$\models_{have} \circ \models_{always}$] a mood in every case.

The formula we posit is approximate, for instance,

(17.3) $(\mathfrak{D} \subset \pi_{project}(\vartheta_{disclose}(\pi(\mathfrak{D})))) \models_{have} \circ \models_{always} \mathfrak{M}$ □

[17.4] The projection [$\pi_{project}$] of its ownmost [$\mathfrak{D}$] potentiality-for-Being [$\pi_{for\text{-}Being}$] has been delivered over to [$\models_{deliver}$] the Fact of its thrownness [$\varphi_{fact}(\tau_{throw})$] into the "there" [$\subset \tau_{there}$].

There is simply

(17.4) $\pi_{project}(\pi_{for\text{-}Being}(\mathfrak{D})) \models_{deliver} (\varphi_{fact}(\tau_{throw}(\mathfrak{D})) \subset \tau_{there})$ □

[17.5] Has not Dasein's Being [$\mathfrak{B}(\mathfrak{D})$] become more [$\not\models_{more}$] enigmatical [$\varepsilon_{enigma}$] now [$\models_{now}$] that [$\models_{that}$] we have explicated [$\models_{explicate}$] the existential constitution of the Being of the "there" [$\varepsilon_{exist}(\gamma_{const}(\mathfrak{B}(\tau_{there})))$] in the sense of thrown projection [$\subset \sigma_{sense}(\tau_{throw}(\pi_{project}))$]?

As in some previous cases, in this formula we use the one [o_{one}] as a neutral discursive entity instead of the we as a more definite one. The last sentence informs as a question pertaining to Dasein's Being [$\mathfrak{Q}_{quest}(\mathfrak{B}(\mathfrak{D}))$]. Thus,

(17.5) $((\mathfrak{B}(\mathfrak{D}) \not\models_{more} \circ \models_{now} \varepsilon_{enigma}) \models_{that} ((o_{one} \models_{explicate} \varepsilon_{exist}(\gamma_{const}(\mathfrak{B}(\tau_{there})))) \subset \sigma_{sense}(\tau_{throw}(\pi_{project})))) \models \mathfrak{Q}_{quest}(\mathfrak{B}(\mathfrak{D}))$ □

[17.6] It has indeed.

The formula for this affirmatively [ι_{indeed}] answering [$\mathfrak{A}_{answer}$] sentence is longer than one could expect, for it must have a reference according to the previous question. Thus,

(17.6) $(\mathfrak{A}_{answer} \models_{to} \mathfrak{Q}_{quest}(\mathfrak{B}(\mathfrak{D}))) \models \iota_{indeed}$ □

[17.7] We [one] must first [$\models_{first}$] let [$\models_{let}$] the full enigmatical character of this Being [$\varphi_{full}(\gamma_{char}(\varepsilon_{enigma}(\mathfrak{B}(\mathfrak{D}))))$] emerge [$\models_{emerge}$], even [$\models_{even}$] if all we can do is to come [$\models_{come}$] to a genuine breakdown [$\gamma_{genuine}(\beta_{breakdown})$] over its 'solution' [$\sigma_{solution}(\mathfrak{B})$], and to formulate [$\models_{formulate}$] anew [$\nu_{anew}$] the question [$\mathfrak{Q}_{quest}$] about [$\models_{about}$] the Being of thrown projective Being-in-the-world [$\tau_{throw}(\pi_{project}(\mathfrak{B}_{in\text{-}the\text{-}world}))$].

A formula for the last sentence is the following:

(17.7) $(\varphi_{full}(\gamma_{char}(\varepsilon_{enigma}(\mathfrak{B}(\mathfrak{D})))) \models_{first} \circ (\models_{let} \circ \models_{emerge})) \models_{even} (((o_{one} \models_{come} \gamma_{genuine}(\beta_{breakdown}) \models_{over} \sigma_{solution}(\mathfrak{B})) \Rightarrow (o_{one} \models_{formulate} \nu_{anew}(\mathfrak{Q}_{quest} \models_{about} (\mathfrak{B} \models_{of} \tau_{throw}(\pi_{project}(\mathfrak{B}_{in\text{-}the\text{-}world}))))))$ □

2.18. *THE EIGHTEENTH PARAGRAPH OF § 31 [BT]*

[18.1] But [$\models_{but}$] in the first instance, even [$\models_{first\text{-}} \circ \models_{even}$] if we are just to bring [$\models_{just} \models_{bring}$] into view [$\models_{into}\ \nu_{view}$] the everyday kind of Being [$\varepsilon_{everyday}(\mathfrak{R}(\mathfrak{B}))$] in which [$\mathfrak{R}(\mathfrak{B})$] there is understanding with [$\models_{with}$] a state-of-mind, and if we are to do so [$\models_{do}\ \sigma_{so}$] in a way which is phenomenally adequate [$\varphi_{phenomenal}(\alpha_{adequate}(\mathfrak{W}_{way}))$] to [$\models_{to}$] the full disclosedness of the "there" [$\varphi_{full}(\vartheta_{disclose}(\tau_{there}))$], [$\Rightarrow$] we must work out [$\models_{must} \circ \models_{work\_out}$] these *existentialia* concretely [$\gamma_{concrete}(\tau_{this}(\mu_{multi}(\varepsilon_{exist})))$].

Finally, we can posit a complicated formula for the last sentence of section §32 [BT] by

(18.1) $((\models_{but} \circ (\models_{first} \circ \models_{even})$

$$(o_{one} \models_{just} \circ \models_{bring} \varepsilon_{everyday}(\Re(\mathfrak{B})) \models_{into} \nu_{view});$$
$$(\Re(\mathfrak{B}) \models (\mathfrak{U} \models_{with} \Theta_{mind})));$$
$$(((o_{one} \models_{do} \sigma_{so}) \models_{in} \varphi_{phenomenal}(\alpha_{adequate}(\mathfrak{W}_{way}))) \models_{to} \varphi_{full}(\vartheta_{disclose}(\tau_{there}))) \Rightarrow$$
$$(o_{one} \models_{must} \circ \models_{work\_out} \Upsilon_{concrete}(\tau_{this}(\mu_{multi}(\varepsilon_{exist}))))$$

As we see, the last formula is an implicative expression with three components in its premise. By this formula our informational investigation of the Heideggerian Being-there as understanding is completed so far. □

3. FORMULA SYSTEMS OF PARAGRAPHS AS INFORMATIONAL ENTITIES

3.0. *PARAGRAPHS AS INFORMATIONAL ENTIRETIES*

What is the informing between sentences of a particular paragraph, their mutual »communication«? We have to say more in concern to this communicative, that is, informational connectedness; how does it arise and which forms does it take at all? How does a paragraph of sentences inform in its entirety? Is this connectedness of sentences in a paragraph automatic, that is, conditioned in an axiomatic way? What is the essential difference between a mathematical and an informational connectedness of this type?

The answer is that the connectedness among informational formulas is posited axiomatically, so, one does not need to describe this connectedness explicitly, by additional formulas. This connectedness is always metaphysical, where the basic connective form $\alpha \models \alpha$ can be decomposed [FIP] anew, in a specific way and never definitely, in each particular case of connectedness. The question is what is the subject of informational impact among formulas possessing particular markers of the one and the same operand? We shall discuss this matter more exhaustively in section 5.1.

Let us comment the informational formula systems in the paragraph by paragraph manner. In this case we proceed systematically, so, we point out the following facts in the framework of a paragraph informational system: (1) operands, (2) operators, and (3) operand informational connections in single and in different formulas of the paragraph system. This information might be relevant in respect to the possibilities of further (informational) development of the paragraph systems (PS). Operands of a PS can be found in the Operand Dictionary, where they are explained verbally and in the shortest form in English, German, and Slovene. These operands perform as regular informational entities which inform and are informed. Certainly, they could be formalized initially, additionally, and fundamentally, that is, foundationally. We shall show the operand connectedness by the references of formulas in which they appear.

Operators of a paragraph formula system will be listed together with information in which formulas of the system do they appear. The shortest verbal explanation (in English, German, and Slovene) of particular informational operators will be found in the Operator Dictionary at the end of the essay. The metaphysical connections of the type $\alpha \models \alpha$, $\alpha \models \varphi(\alpha)$, ... will be listed together with operands and sometimes discussed separately.

Another aspect of the formalistic treatment of sentences in a paragraph is to observe and classify the so-called input, output, and processing (understanding) informational operands. It may happen that a paragraph is not a complete informational system in the sense of its input and output informational variables and in the sense of input-output processing variables which all together constitute an »informational machine«. Some parts of such informational system may be »distributed« in other paragraphs, thus, only the entire section of paragraphs is, for instance, more or less a complete system, but, certainly, remaining informationally open for informing (in the sense of operand »signaling«) and structural impacts onto the already established informational system. At this point the traditional linguistic theory encounters a new, uncommon »sight« of the problem which may substantially exceed the domain of the linguistic semiotics as a scientific discipline. Namely, the new substance of the problem concerns the domain of informational arising.

3.1. *A FORMULA SYSTEM OF THE FIRST PARAGRAPH OF §31 [BT]*

The informational formula system of the first paragraph is the following:

(1.1) $\mathfrak{S}_{mind} \in \sigma_{exist}$;
$(\mathfrak{B}_{there} \models \sigma_{exist}) \models \mathfrak{B}_{there}$;
(1.2) $(\mathfrak{S}_{mind}, \mathfrak{U} \models_{const} \mathfrak{B}_{there}) \models_{equi\_p}$;
(1.3) $(\mathfrak{U} \subset_{always} \mathfrak{S}_{mind}) \models_{supp}$;
(1.4) $\mathfrak{M}(\mathfrak{U}) \subset_{always} \mathfrak{U}$;
(1.5) $\mathfrak{B}_{\mathfrak{D}} \subset \mathfrak{D}$; $\mu_{basic} \in \mathfrak{B}_{\mathfrak{D}}$;
$((\mathfrak{B}_{\mathfrak{D}} \models_{int} \mathfrak{U}) \models \varepsilon_{fund}) \Rightarrow (\mu_{basic} \in \mathfrak{B}_{\mathfrak{D}})$;
(1.6) $((\mathfrak{U} \in \mathfrak{K}_{cogn}) \models_{int} \delta_{exist}) \subset \mathfrak{U}_{prim}$;
$\mathfrak{B}_{there} \models (\mathfrak{U}_{prim} \subset \mathfrak{B}_{there})$;
$\mathfrak{U} \neq \mathfrak{E}_{expl}$

This system is the initial (textual) informational program corresponding to the first paragraph. To this program one can imagine the so-called dictionary (library) of informational operands being informationally determined as basic informational programs in operand dictionary. This dictionary can be understood to be a regular part of the so-called operating system of an informational machine which performs the informing in a dedicated field, for instance, informing of Being- there as understanding. Further, we must not forget that operators, in a given formula situation, pertain always to concrete operands, particularizing operands' specific faculties of informing. And last but not least, metaphysical connections of a formula system have to be pointed out with the aim to offer possibilities for a further metaphysical decomposition of the formula system.

Let us show the appearing operands for paragraph (1) in the Greek and Fraktur alphabetical order and the occurring operators:

δ_{exist}: 1.6; ε_{fund}: 1.5;
μ_{basic}: 1.5, 1.5; σ_{exist}: 1.1, 1.1;

$\mathfrak{B}_{\mathfrak{D}}$ or $\mathfrak{B}(\mathfrak{D})$: 1.5, 1.5;
$\mathfrak{B}_{there}$: 1.1, 1.1, 1.2, 1.6, 1.6;
$\mathfrak{D}$: 1.5; $\mathfrak{E}_{expl}$: 1.6;
$\mathfrak{K}_{cogn}$: 1.6; $\mathfrak{M}$: 1.4;
$\mathfrak{M}(\mathfrak{U})$: 1.4; w_{mind}: 1.1, 1.2, 1.3;
$\mathfrak{U}$: 1.2, 1.3, 1.4, 1.5, 1.6; $\hat{A}_{prim}$: 1.6;

$\models$: 1.1, 1.5, 1.6; $\models_{const}$: 1.2;
$\models_{equi\_p}$: 1.2; $\models_{int}$: 1.5, 1.6;
$\models_{supp}$: 1.3; $\Rightarrow$: 1.5;
$\subset$: 1.5, 1.6; $\subset_{always}$: 1.3, 1.4;
$\in$: 1.1, 1.5, 1.6; $\neq$: 1.6;

Additional formulas for a possible supplementary metaphysical decomposition are:

$\sigma_{exist} \models \sigma_{exist}$: 1.1—1.1;
$\mathfrak{B}_{there} \models \mathfrak{B}_{there}$: 1.1—1.1;
$\mathfrak{S}_{mind} \models \mathfrak{S}_{mind}$: 1.1—1.2—1.3;
$\mathfrak{M}(\mathfrak{U}) \models \mathfrak{U}$: 1.4 and connections to other occurrences of $\mathfrak{U}$ and vice versa;
$\mathfrak{U} \models \mathfrak{U}$: 1.2—1.3—1.41.5—1.6

etc. These and other formulas can be used for the supplementary explanation of the formula system (1). As we see, entities Being-there $\mathfrak{B}_{there}$ and understanding $\mathfrak{U}$ dominate in the first paragraph formula system. It seems that this paragraph determines, in part, the kernel part of the understanding system and that something to be understood and the characteristic results of understanding (for instance, meaning, significance, etc. of something) are postponed to the informational game of concerned entities to formula systems of paragraphs which follow.

3.2. *A FORMULA SYSTEM OF THE SECOND PARAGRAPH OF §31 [BT]*

The formula system for the second paragraph is

(2.1) [A comment];
(2.2) $((\mathfrak{D} \models; \models \mathfrak{D}) \models \tau_{there}(\mathfrak{D}))$
$\Leftrightarrow (\mathfrak{W}_{world} \models \tau_{there})$;
$\mathfrak{B}_{there}(\mathfrak{W}_{world}) \models \mathfrak{B}_{in}$;
(2.3) $(\mathfrak{B}_{in} \models \tau_{there}) \Rightarrow (\mathfrak{D} \models; \models \mathfrak{D})$;
(2.4) $((\mathfrak{B}_{in\text{-}the\text{-}world} \models_{discl} \mathfrak{U}) \subset \varphi_{sake}$;
$\varphi_{sake} \Rightarrow$
$((((\mathfrak{D} \models; \models \mathfrak{D}) \models \tau_{there}(\mathfrak{D})) \Leftrightarrow$
$(\mathfrak{W}_{world} \models \tau_{there}))$;
$(\mathfrak{B}_{there}(\mathfrak{W}_{world}) \models \mathfrak{B}_{in})$;
$((\mathfrak{B}_{in} \models \tau_{there}) \Rightarrow (\mathfrak{D} \models; \models \mathfrak{D})))$;
(2.5) $(\xi_{sign} \subset \mathfrak{U}(\varphi_{sake})) \models_{discl}$;
(2.6) $((\mathfrak{U} \models_{discl}) \models_{as}$
$(\mathfrak{U}(\varphi_{sake} \models_{discl}; \xi_{sign} \models_{discl}))$
$\rightarrow_{equi\_p} \mathfrak{B}_{in\text{-}the\text{-}world}$;
(2.7) $\mathfrak{W}_{world}(\xi_{sign}) \models_{discl} \mathfrak{W}_{world}$;
(2.8) $(\varphi_{sake}, \xi_{sign} \subset_{discl} \mathfrak{D}) \Leftrightarrow$

$((\mathfrak{D} \models_{as} \mathfrak{B}_{in\text{-}the\text{-}world}) \models \mathfrak{D})$

Let us see the list of informational operands and operators in the last system:

ξ_{sign}: 2.5, 2.6, 2.7, 2.8; τ_{there}: 2.2, 2.3, 2.4, 2.4; $\tau_{there}(\mathfrak{D})$: 2.2, 2.4;
φ_{sake}: 2.4, 2.4, 2.5, 2.6, 2.8;

$\mathfrak{B}_{in}$: 2.2, 2.3, 2.4, 2.4;
$\mathfrak{B}_{in\text{-}the\text{-}world}$: 2.4, 2.6, 2.8;
$\mathfrak{B}_{there}$: 2.2, 2.4; $\mathfrak{B}_{there}(\mathfrak{W}_{world})$: 2.2, 2.4;
$\mathfrak{D}$: 2.2, 2.2, 2.3, 2.3, 2.4, 2.4, 2.8, 2.8;
$\mathfrak{U}$: 2.6, 2.6; $\mathfrak{U}(\varphi_{sake})$: 2.5;
$\mathfrak{W}_{world}$: 2.2, 2.4, 2.7; $\mathfrak{W}_{world}(\xi_{sign})$: 2.7;

$\models$: 2.2, 2.3, 2.4, 2.8; $\models_{as}$: 2.6, 2.8;
$\models_{discl}$: 2.4, 2.5, 2.6, 2.7;
$\subset$: 2.4, 2.5; $\subset_{discl}$: 2.8;
$\Rightarrow$: 2.3, 2.4; $\Leftrightarrow$: 2.2, 2.8;
$\rightarrow_{equi\_p}$: 2.6

Possible metaphysical informing between and within formulas can be added in a spontaneous way, according to the initial formula situations.

3.3. *A FORMULA SYSTEM OF THE THIRD PARAGRAPH OF §31 [BT]*

Formulas of the third paragraph constitute a part of the understanding system, that is,

(3.1) $\models_{ont,some} (\mathfrak{U}(\alpha) \models (\mu_{sign}(\models_{able\_man} \alpha), \mu_{sign}(\models_{match} \alpha), \mu_{sign}(\models_{comp} \alpha)))$;
(3.2) $(\mathfrak{W}_{what} \not\models_{comp} \alpha; \mathfrak{B} \models_{as} (\mathfrak{B} \models; \models \mathfrak{B})) \subset (\mathfrak{U} \models_{as} \varepsilon_{exist})$;
(3.3) $((\Re(\mathfrak{B}) \subset \mathfrak{D}) \models_{as} \pi_{for\text{-}Being}) \subset_{exist} \mathfrak{U}$;
(3.4) $(\alpha \subset_{comp} (\mathfrak{D} \not\models \alpha_{present\text{-}at\text{-}hand})) \models_{extra}$; $(\mathfrak{D} \models_{prim} \mathfrak{B}_{possible})$;
(3.5) $(\mathfrak{D} \models (\mathfrak{D} \models; \models \mathfrak{D})) \models \pi(\mathfrak{D})$;
(3.6) $((\mathfrak{D} \models_{pertain} \pi_{for\text{-}Being}) \models_{pertain} \mathfrak{D}) \subset_{always} ((\mathfrak{B}_{possible} \models_{essen\_for} \mathfrak{D}) \models_{pertain} (((\pi_{solicitude}(\mathfrak{D}) \models_{for} \omega); (\pi(\mathfrak{D}) \models_{concern} \mathfrak{W}_{world})) \dashv_{char}))$;
(3.7) $(((\mathfrak{D} \subset \mathfrak{B}_{possible}) \models_{exist}) \neq_{sharply} ((\pi_{log} \models_{empty}), \gamma_{cont}(\alpha_{present\text{-}at\text{-}hand}))) \Leftarrow_{so\text{-}far\text{-}as} (\alpha \models \alpha_{present\text{-}at\text{-}hand})$
(3.8) $(\gamma_{modal}(\pi_{at\text{-}hand})) \dashv_{as} (\pi \models_{sign} ((\alpha \not\models_{act}), (\alpha \not\models \nu_{at\_any\_time})))$;
(3.9) $\pi \models_{char} \alpha_{merely\_poss}$;
(3.10) $(\pi \models_{ont} \lambda_{lower}) \models_{than} (\alpha_{act}, \nu)$;
(3.11) $(\pi \models_{as} \varepsilon_{exist}) \Vdash \upsilon_{way}$; $((\models_{char} \mathfrak{D}) \subset \upsilon_{way}) \models_{ont}$;
(3.12) $\pi \models_{as} (\mathfrak{E}_{exist} \models; \models \mathfrak{E}_{exist})$;
(3.13) $(\mathfrak{U} \models_{as} \pi_{for\text{-}Being}) \models_{prov} ((\varphi_{basis} \models \pi) \models \varphi_{basis})$

From the last formula system we can extract informational operands and operators in the form of the following list:

α: 3.1, 3.1, 3.2, 3.4, 3.7, 3.8, 3.8;
α_{act}: 3.10; α_{merely_poss}: 3.9;
$\alpha_{present\text{-}at\text{-}hand}$: 3.4, 3.7;
$\gamma_{cont}(\alpha_{present\text{-}at\text{-}hand})$: 3.7;
$\gamma_{modal}(\pi_{present\text{-}at\text{-}hand})$: 3.8;
ε_{exist}: 3.2, 3.11; λ_{lower}: 3.10;
μ_{sign}: 3.1, 3.1; ν: 3.10;
$\nu_{at\_any\_time}$: 3.8;
π: 3.8, 3.9, 3.10, 3.11, 3.12, 3.13;
$\pi(\mathfrak{D})$: 3.5, 3.6; $\pi_{for\text{-}Being}$: 3.3, 3.6, 3.13;
π_{log}: 3.7; $\pi_{at\text{-}hand}$: 3.8;
$\pi_{solicitude}(\mathfrak{D})$: 3.6; υ_{way}: 3.11, 3.11;
φ_{basis}: 3.13, 3.13; ω: 3.6;

$\mathfrak{B}$: 3.2, 3.2, 3.3; $\mathfrak{B}_{possible}$: 3.4, 3.6, 3.7;
$\mathfrak{D}$: 3.3, 3.4, 3.4, 3.5, 3.5, 3.6, 3.6, 3.7;
$\mathfrak{E}_{exist}$: 3.12, 3.12; $\Re$: 3.3;
$\mathfrak{U}$: 3.1, 3.2, 3.3, 3.13; $\mathfrak{U}(\alpha)$: 3.1;
$\mathfrak{W}_{what}$: 3.2; $\mathfrak{W}_{world}$: 3.6;

$\models$: 3.1, 3.2, 3.5, 3.12, 3.13;
$\models_{able\_man}$: 3.1;
$\models_{as}$: 3.2, 3.11, 3.12, 3.13;
$\models_{char}$: 3.9, 3.11; $\models_{comp}$: 3.1;
$\models_{concern}$: 3.6; $\models_{empty}$: 3.7;
$\models_{essen\_for}$: 3.6; $\models_{exist}$: 3.7;
$\models_{extra}$: 3.4; $\models_{for}$: 3.6;
$\models_{match}$: 3.1; $\models_{ont}$: 3.10, 3.11;
$\models_{ont,some}$: 3.1; $\models_{pertain}$: 3.6;
$\models_{prim}$: 3.4; $\models_{prov}$: 3.13;
$\models_{sign}$: 3.8; $\models_{than}$: 3.10;
$\not\models$: 3.4, 3.8; $\not\models_{act}$: 3.8;
$\not\models_{comp}$: 3.2; $\dashv_{as}$: 3.8;
$\dashv_{char}$: 3.6; $\Vdash$: 3.11;
$\subset$: 3.2, 3.3, 3.7, 3.11; $\subset_{always}$: 3.6;
$\subset_{comp}$: 3.4; $\subset_{exist}$: 3.3;
$\Leftarrow_{so\text{-}far\text{-}as}$: 3.7; $\neq_{sharply}$: 3.7

In this way, the survey of the involved informational entities of the third paragraph is completed.

3.4. *A FORMULA SYSTEM OF THE FOURTH PARAGRAPH OF §31 [BT]*

The fourth paragraph is constituted by the following formula system:

(4.1) $(\pi \models_{\text{as}} \varepsilon_{\text{exist}}) \not\models_{\text{sign}} ((\pi_{\text{for-Being}} \models; \models \pi_{\text{for-Being}}) \models \sigma_{\text{sense}}(\lambda_{\text{liber}}(\iota_{\text{indiff}})))$;

(4.2) $(\mathfrak{S}_{\text{mind}} \subset_{\text{essen}} \mathfrak{D}) \models_{\text{already}} (\mathfrak{D} \subset \pi_{\text{def}})$;

(4.3) $(\mathfrak{D} \models_{\text{as}} (\pi_{\text{for-Being}} \models; \models \pi_{\text{for-Being}})) \models_{\text{pass\_by}} ((\pi_{\text{for-Being}})$; $\mathfrak{D} \models_{\text{waive}} \pi(\mathfrak{B}(\mathfrak{D}))$; $(\mathfrak{D} \models_{\text{seize}} \pi(\mathfrak{B}(\mathfrak{D}))) \models \mu_{\text{mistake}}$;

(4.4) $\varphi_{(4.3)} \Leftrightarrow ((((\mathfrak{D} \models \mathfrak{B}_{\text{possible}}) \models_{\text{deliver}} \mathfrak{D}) \models \pi_{\text{thrown}})) \models_{\text{through}}$;

(4.5) $(\mathfrak{D} \models \pi(\mathfrak{B}_{\text{free}})) \models_{\text{for}} \pi_{\text{for-Being}}(\mathfrak{D})$;

(4.6) $(\mathfrak{B}_{\text{possible}}(\mathfrak{D}) \models_{\text{trans}} \mathfrak{D}) \models \alpha_{\text{poss\_ways}}$

The list of informational operands and operators appearing in this formula system is:

$\alpha_{\text{poss\_ways}}$: 4.6; $\varepsilon_{\text{exist}}$: 4.1;
ι_{indiff}: 4.1; λ_{liber}: 4.1;
$\lambda_{\text{liber}}(\iota_{\text{indiff}})$: 4.1; μ_{mistake}: 4.3;
π: 4.1, 4.3, 4.3, 4.5; $\pi(\mathfrak{B}(\mathfrak{D}))$: 4.3, 4.3;
π_{def}: 4.2;
$\pi_{\text{for-Being}}$: 4.1, 4.1, 4.3, 4.3; 4.5;
$\pi_{\text{for-Being}}(\mathfrak{D})$: 4.5; $\pi(\mathfrak{B}_{\text{free}})$: 4.5;
π_{thrown}: 4.4; σ_{sense}: 4.1;
$\sigma_{\text{sense}}(\lambda_{\text{liber}}(\iota_{\text{indiff}}))$: 4.1; $\varphi_{(3.4)}$: 4.4;

$\mathfrak{B}$: 4.3; $\mathfrak{B}(\mathfrak{D})$: 4.3, 4.3;
$\mathfrak{B}_{\text{free}}$: 4.5; $\mathfrak{B}_{\text{possible}}$: 4.4, 4.6;
$\mathfrak{B}_{\text{possible}}(\mathfrak{D})$: 4.6;
$\mathfrak{D}$: 4.2, 4.2, 4.3, 4.3, 4.4, 4.4, 4.5, 4.5, 4.6, 4.6; $\mathfrak{S}_{\text{mind}}$: 4.2;

$\models$: 4.1, 4.3, 4.4, 4.5, 4.6;
$\models_{\text{already}}$: 4.2; $\models_{\text{as}}$: 4.1, 4.3;
$\models_{\text{deliver}}$: 4.4; $\models_{\text{for}}$: 4.5;
$\models_{\text{pass\_by}}$: 4.3; $\models_{\text{seize}}$: 4.3;
$\models_{\text{through}}$: 4.4; $\models_{\text{trans}}$: 4.6;
$\models_{\text{waive}}$: 4.3; $\not\models_{\text{sign}}$: 4.1;
$\subset$: 4.2; $\subset_{\text{essen}}$: 4.2;
$\Leftrightarrow$: 4.4

So, the operand and operator list is completed.

3.5. *A FORMULA SYSTEM OF THE FIFTH PARAGRAPH OF §31 [BT]*

The formula system of the fifth paragraph is constituted by the following seven formula subsystems:

(5.1) $(\mathfrak{U} \models \mathfrak{B}(\pi_{\text{for-Being}})) \not\models (\alpha_{\text{still-out}} \models_{\text{as\_not\_yet}} \pi_{\text{at\_hand}})$; $((\mathfrak{B}(\pi_{\text{for-Being}}) \models_{\text{as}} \alpha) \not\models_{\text{essen}} \pi_{\text{at\_hand}}) \models_{\text{with}} (\mathfrak{B}(\mathfrak{D}) \subset \sigma_{\text{exist}})$;

(5.2) $(\mathfrak{D} \forall (\mathfrak{D} \models_{\mathfrak{U}}; \mathfrak{D} \not\models_{\mathfrak{U}})) \models (\mathfrak{D} \models; \models \mathfrak{D})$;

(5.3) $(\mathfrak{D} \models_{\text{as}} \mathfrak{U}) \models_{\text{know}} ((\mathfrak{D} \models_{\text{cap}}), \pi_{\text{for-Being}}(\mathfrak{D}) \models_{\text{cap}})$;

(5.4) $\mathfrak{K}_{\text{know}}(\mathfrak{D}) \not\models_{\text{first}} \mathfrak{P}_{\text{imm}}(\mathfrak{D})$; $\mathfrak{K}_{\text{know}}(\mathfrak{D}) \subset (\mathfrak{B}(\tau_{\text{there}}) \models_{\text{essential}} \mathfrak{U})$;

(5.5) $(((\mathfrak{D} \models \mathfrak{U}) \models \mathfrak{D}) \models \tau_{\text{there}}(\mathfrak{D})) \Rightarrow (\mathfrak{D} \models_{\text{astray}}; \mathfrak{D} \models_{\text{fail\_to\_recognize}} \mathfrak{D})$;

(5.6) $((\mathfrak{U} \models_{\text{accomp\_by}} \mathfrak{S}_{\text{mind}}) \models_{\text{exist\_surr}} \tau_{\text{thrown}}) \Rightarrow (\mathfrak{D} \forall_{\text{already}} (\mathfrak{D} \models_{\text{astray}} \mathfrak{D}; \mathfrak{D} \models_{\text{fail\_to\_recognize}} \mathfrak{D}))$;

(5.7) $(\mathfrak{D} \subset \pi_{\text{for-Being}}(\mathfrak{D})) \models_{\text{deliver}} (\pi(\mathfrak{D} \models_{\text{first\_find}} \mathfrak{D}) \subset \pi(\mathfrak{D}))$

Operands and operators of the system are:

α: 5.1; $\alpha_{\text{still-out}}$: 5.1;
π: 5.7, 5.7; $\pi_{\text{at\_hand}}$: 5.1, 5.1;
$\pi(\mathfrak{D})$: 5.7, 5.7; $\pi(\mathfrak{D} \models_{\text{first\_find}} \mathfrak{D})$: 5.7;
$\pi_{\text{for-Being}}$: 5.1, 5.1, 5.3, 5.7;
$\pi_{\text{for-Being}}(\mathfrak{D})$: 5.7; σ_{exist}: 5.1;
τ_{there}: 5.5; $\tau_{\text{there}}(\mathfrak{D})$: 5.5;
τ_{thrown}: 5.6;

$\mathfrak{B}$: 5.1, 5.1, 5.4; $\mathfrak{B}(\mathfrak{D})$: 5.1;
$\mathfrak{B}(\pi_{\text{for-Being}})$: 5.1, 5.1; $\mathfrak{B}(\tau_{\text{there}})$: 5.4;
$\mathfrak{D}$: 5.2, 5.2, 5.3, 5.3, 5.4, 5.4, 5.5, 5.5, 5.6, 5.6, 5.7, 5.7; $\mathfrak{K}_{\text{know}}$: 5.4, 5.4;
$\mathfrak{K}_{\text{know}}(\mathfrak{D})$: 5.4, 5.4; $\mathfrak{P}_{\text{imm}}$: 5.4;
$\mathfrak{P}_{\text{imm}}(\mathfrak{D})$: 5.4; $\mathfrak{S}_{\text{mind}}$: 5.6;
$\mathfrak{U}$: 5.1, 5.3, 5.4, 5.6;

$\models$: 5.1, 5.2, 5.5; $\models_{\text{accomp\_by}}$: 5.6;
$\models_{\text{as}}$: 5.1, 5.3; $\models_{\text{as\_not\_yet}}$: 5.1;
$\models_{\text{astray}}$: 5.5, 5.6; $\models_{\text{cap}}$: 5.3;
$\models_{\text{deliver}}$: 5.7; $\models_{\text{essential}}$: 5.4;
$\models_{\text{exist\_surr}}$: 5.6;

$\models_{\text{fail\_to\_recognize}}$: 5.5, 5.6;
$\models_{\text{first\_find}}$: 5.7; $\models_{\text{know}}$: 5.3;
$\models_{\mathfrak{U}}$: 5.2; $\models_{\text{with}}$: 5.1;
$\not\models$: 5.1; $\not\models_{\text{essen}}$: 5.1;
$\not\models_{\text{first}}$: 5.4; $\not\models_{\mathfrak{U}}$: 5.2;
$\subset$: 5.1, 5.4, 5.7; $\Rightarrow$: 5.5, 5.6;
$\forall$: 5.2; $\forall_{\text{already}}$: 5.6

In this list, entity $\pi(\mathfrak{D} \models_{\text{first\_find}} \not\models)$ should be mentioned as a complex functional possibility π.

3.6. *A FORMULA SYSTEM OF THE SIXTH PARAGRAPH OF §31 [BT]*

Within this paragraph only one sentence is considered since the second one is taken as a suppressed comment:

(6.1) $((\mathfrak{U} \models \mathfrak{B}_{\text{exist}}((\mathfrak{D} \models \pi_{\text{for-Being}}(\mathfrak{D})) \models \mathfrak{D})) \models_{\text{discl}} \mathfrak{B}_{\text{exist}}) \models_{\text{cap}} \mathfrak{W}_{\text{what}}(\mathfrak{B}_{\text{exist}})$;
(6.2) [A comment to that what has to follow]

In short, there is:

$\pi_{\text{for-Being}}$: 6.1; $\pi_{\text{for-Being}}(\mathfrak{D})$: 6.1;

$\mathfrak{B}_{\text{exist}}$: 6.1, 6.1; $\mathfrak{D}$: 6.1, 6.1;
$\mathfrak{U}$: 6.1; $\mathfrak{W}_{\text{what}}$: 6.1;
$\mathfrak{W}_{\text{what}}(\mathfrak{B}_{\text{exist}})$: 6.1;

$\models$: 6.1; $\models_{\text{cap}}$: 6.1;
$\models_{\text{discl}}$: 6.1

The analysis of this paragraph as a whole can bring new views of its role within the understanding system.

3.7. *A FORMULA SYSTEM OF THE SEVENTH PARAGRAPH OF §31 [BT]*

The seventh paragraph is informationally described by the system of the first nine sentences' subsystems:

(7.1) $(\mathfrak{U} \models_{\text{as}} \mathfrak{D}_{\text{discl}}) \forall_{\text{pertain}} \mathfrak{W}_{\text{basic\_state}}(\mathfrak{B}_{\text{in-the-world}})$;
(7.2) $(\mathfrak{B}_{\text{in}} \models_{\text{as}} \pi_{\text{for}}(\mathfrak{B}_{\text{in-the-world}})$;
(7.3) $((\mathfrak{W}_{\text{world}} \models_{\text{as}} \mathfrak{W}_{\text{world}}) \models_{\text{discl}} \xi_{\text{sign\_poss}}) \models_{\text{but}} (((\alpha \subset \mathfrak{W}_{\text{world}}) \Rightarrow (\alpha \models_{\text{free}} \alpha)) \Rightarrow (\pi(\alpha) \models_{\text{free}} \alpha))$;
(7.4) $(\alpha \models_{\text{as}} \mathfrak{U}_{\text{service}}(\alpha)$; $\alpha \models_{\text{as}} \mathfrak{U}_{\text{use}}(\alpha)$; $\alpha \models_{\text{as}} \mathfrak{U}_{\text{detriment}}(\alpha)) \models_{\text{discover\_as}} (\alpha \models \mathfrak{R}_{\text{to-hand}})$;
(7.5) $(\mathfrak{W}_{\text{categorial}} \models_{\text{reveal\_as}} \tau_{\text{total}}(\iota_{\text{involve}})) \models_{\text{of}} \pi_{\text{interconn}}(\mathfrak{R}_{\text{to-hand}})$;
(7.6) $(\models_{\text{discl}} \pi(\upsilon_{\text{unity}})) \Rightarrow (\models_{\text{discover}} (\upsilon_{\text{unity}}(\mu_{\text{manifold}}(\pi_{\text{at-hand}})), \upsilon_{\text{unity}}(\mathfrak{N})))$;
(7.7) $((\mathfrak{D}_{\text{question}} \models_{\text{about}} (\mathfrak{B}(\mathfrak{N}) \models_{\text{aim\_at}} \Upsilon_{\text{cond}}(\pi(\mathfrak{B}(\mathfrak{N}))))) \models_{\text{accid}}) \models_{\text{quest}}$;
(7.8) $\varphi_{(7.7)} \models_{\text{quest}} \mathfrak{W}_{\text{what}}$;
(7.9) $(\models_{\text{confront}} (\varphi_{(7.7)} \models_{\text{quest}} \alpha)) \Rightarrow ((((\alpha \not\models \Upsilon_{\text{char}}(\mathfrak{D})) \models_{\mathfrak{U}} \mathfrak{B}(\alpha)) \Leftarrow (\Upsilon_{\text{cond}}(\pi(\alpha)) \models_{\text{discl}} \alpha)) \models_{\text{why}})$;
(7.10) [An insignificant comment];
(7.11) [A comment]

The list of operands and operators is as follows:

α: 7.3, 7.3, 7.4, 7.4, 7.9, 7.9;
Υ_{char}: 7.9; $\Upsilon_{\text{char}}(\mathfrak{D})$: 7.9;
Υ_{cond}: 7.7, 7.9; $\Upsilon_{\text{cond}}(\pi(\alpha))$: 7.9;
$\Upsilon_{\text{cond}}(\pi(\mathfrak{B}(\mathfrak{N})))$: 7.7; ι_{involve}: 7.5;
μ_{manifold}: 7.6; $\mu_{\text{manifold}}(\pi_{\text{at-hand}})$: 7.6;
$\xi_{\text{sign\_poss}}$: 7.3; π: 7.3, 7.6, 7.7, 7.9;
$\pi(\alpha)$: 7.3, 7.9; $\pi(\mathfrak{B}(\mathfrak{N}))$: 7.7;
$\pi(\upsilon_{\text{unity}})$: 7.6; $\pi_{\text{at-hand}}$: 7.6;
π_{for}: 7.2; $\pi_{\text{for}}(\mathfrak{B}_{\text{in-the-world}})$: 7.2;
$\pi_{\text{interconn}}$: 7.5; $\pi_{\text{interconn}}(\mathfrak{R}_{\text{to-hand}})$: 7.5;
τ_{total}: 7.5; $\tau_{\text{total}}(\iota_{\text{involve}})$: 7.5;
υ_{unity}: 7.6;
$\upsilon_{\text{unity}}(\mu_{\text{manifold}}(\pi_{\text{at-hand}}))$: 7.6;
$\upsilon_{\text{unity}}(\mathfrak{N})$: 7.6; $\varphi_{(7.7)}$: 7.8, 7.9;

$\mathfrak{U}_{\text{detriment}}$: 7.4; $\mathfrak{U}_{\text{detriment}}(\alpha)$: 7.4;
$\mathfrak{U}_{\text{service}}$: 7.4; $\mathfrak{U}_{\text{service}}(\alpha)$: 7.4;
$\mathfrak{U}_{\text{use}}$: 7.4; $\mathfrak{U}_{\text{use}}(\alpha)$: 7.4;
$\mathfrak{B}$: 7.7, 7.9; $\mathfrak{B}(\alpha)$: 7.9;
$\mathfrak{B}(\mathfrak{N})$: 7.7; $\mathfrak{B}_{\text{in}}$: 7.2;
$\mathfrak{B}_{\text{in-the-world}}$: 7.1, 7.2; $\mathfrak{D}$: 7.9;
$\mathfrak{D}_{\text{discl}}$: 7.1; $\mathfrak{N}$: 7.6, 7.7;
$\mathfrak{D}_{\text{question}}$: 7.7; $\mathfrak{R}_{\text{to-hand}}$: 7.4, 7.5;
$\mathfrak{U}$: 7.1; $\mathfrak{W}_{\text{basic\_state}}$: 7.1;
$\mathfrak{W}_{\text{basic\_state}}(\mathfrak{B}_{\text{in-the-world}})$: 7.1;
$\mathfrak{W}_{\text{categorial}}$: 7.5; $\mathfrak{W}_{\text{what}}$: 7.8;
$\mathfrak{W}_{\text{world}}$: 7.3, 7.3;

$\models$: 7.4; $\models_{\text{about}}$: 7.7;

$\models_{accid}$: 7.7; $\models_{aim\_at}$: 7.7;
$\models_{as}$: 7.1, 7.2, 7.3, 7.4; $\models_{but}$: 7.3;
$\models_{confront}$: 7.9; $\models_{discl}$: 7.3, 7.6, 7.9;
$\models_{discover\_as}$: 7.4, 7.6; $\models_{free}$: 7.3;
$\models_{of}$: 7.5; $\models_{quest}$: 7.7, 7.8, 7.9;
$\models_{reveal\_as}$: 7.5; $\models_{\mathfrak{U}}$: 7.9;
$\models_{why}$: 7.9; $\not\models$: 7.9;
$\subset$: 7.3; $\Rightarrow$: 7.3, 7.6, 7.9;
$\Leftarrow$: 7.9; $\forall_{pertain}$: 7.1

This paragraph is evidently active in an input, output, and processing (understanding) way.

3.8. *A FORMULA SYSTEM OF THE EIGHTH PARAGRAPH OF §31 [BT]*

The eighth paragraph is depicted by the following formula system:

(8.1) $(((\varepsilon_{dim}(\alpha \subset_{discl} \mathfrak{U}) \models_{all\_press\_for} \pi) \models \mathfrak{U}) \models_{quest} \mathfrak{W}_{why};$
$((\alpha \models \varepsilon_{dim}) \models \alpha) \models \varepsilon_{dim}(\alpha);$
$(\mathfrak{U} \subset_{discl} \varepsilon_{dim}(\alpha)) \subset_{discl} \mathfrak{U};$

(8.2) $((\sigma_{exist} \subset \mathfrak{U}) \models \pi_{project}) \Rightarrow \varphi_{(8.1)};$

(8.3) $(\mathfrak{U} \models_{with\_eq\_p} \mathfrak{B}(\mathfrak{D})) \models_{proj\_upon} (\varphi_{sake}(\mathfrak{U}), \xi_{sign} \models_{as} \mathfrak{W}_{worldhood}(\mathfrak{W}_{cur\_world}(\mathfrak{U})));$

(8.4) $((\gamma_{char}(\mathfrak{U}) \models_{as} \pi_{project}) \models_{const} \mathfrak{B}_{in\_the\text{-}world}) \models_{with\_regard} \vartheta_{discl}(\mathfrak{S}_{of\text{-}Being}(\gamma_{char}(\mathfrak{U})) \models_{exist\_const});$
$(\gamma_{char}(\mathfrak{U}) \models_{by} \varphi_{fact}(\pi_{for\text{-}Being})) \models_{get} \lambda_{leeway}(\gamma_{char}(\mathfrak{U}));$

(8.5) $(\mathfrak{D} \models_{as} \tau_{thrown}) \models_{thrown} (\mathfrak{K}(\mathfrak{B}) \models \mathfrak{P}_{project});$

(8.6) $\mathfrak{P}_{project} \not\models (\mathfrak{C}_{comport}(\omega_{oneself}) \models_{towards} ((\models_{think\_out} \pi_{plan}) \dashv_{accord} (\mathfrak{D} \models_{arr} \mathfrak{B}(\mathfrak{D}))));$

(8.7) $\mathfrak{D} \models_{contra} ((\mathfrak{D} \models_{as} \mathfrak{D}) \models_{project} \mathfrak{D});$
$((\mathfrak{D} \models; \models \mathfrak{D}) \models_{as\_long\_as} \mathfrak{D}) \models \mathfrak{P}_{project};$

(8.8) $(((\mathfrak{D} \models; \models \mathfrak{D}) \models_{as\_long\_as} \mathfrak{D}) \models_{al\_underst} \mathfrak{D}) \models_{as} \pi(\mathfrak{D});$

(8.9) $(\gamma_{char}(\mathfrak{U}) \models \pi_{project}) \models \alpha;$
$\mathfrak{U} \not\models_{grasp\_thema} ((\mathfrak{U}(\alpha) \models_{project} \alpha), \pi);$

(8.10) $(\mathfrak{G}_{grasp}(\mathfrak{U}) \models_{in\_manner} \varphi_{(8.9)}) \models_{take\_away} ((\mathfrak{U} \models \mathfrak{P}_{project}(\alpha)) \models_{as} \pi(\gamma_{char}(\mathfrak{U})));$
$\mathfrak{G}_{grasp}(\mathfrak{U}) \models_{reduce} (\mathfrak{U} \models_{to} \gamma_{content}(\mu_{mind}));$
$((\pi_{project} \subset \mathfrak{T}_{throw}) \models_{throw\_before} \pi_{project}) \models$
$((\pi \models_{as} \pi) \models; \models (\pi \models_{as} \pi));$

(8.11) $(\mathfrak{U} \models (\pi(\mathfrak{U}) \models_{as} \pi(\mathfrak{U}))) \subset$
$((\mathfrak{U} \models_{as} \mathfrak{P}_{project}) \models \mathfrak{K}(\mathfrak{B}(\mathfrak{D})))$

In this system the following operands and operators appear:

α: 8.1, 8.1, 8.9, 8.9, 8.9, 8.10;
γ_{char}: 8.4, 8.4, 8.9, 8.10;
$\gamma_{char}(\mathfrak{U})$: 8.4, 8.4, 8.9, 8.10;
$\gamma_{content}$: 8.10; $\gamma_{content}(\mu_{mind})$: 8.10;
ε_{dim}: 8.1, 8.1; $\varepsilon_{dim}(\alpha)$: 8.1;
$\varepsilon_{dim}(\alpha \subset_{discl} \mathfrak{U})$: 8.1; ϑ_{discl}: 8.4;
$\vartheta_{discl}(\mathfrak{S}_{of\text{-}Being}(\gamma_{char}(\mathfrak{U})) \models_{exist\_const})$: 8.4;
λ_{leeway}: 8.4; $\lambda_{leeway}(\gamma_{char}(\mathfrak{U}))$: 8.4;
μ_{mind}: 8.10; ξ_{sign}: 8.3;
π: 8.1, 8.8, 8.9, 8.10, 8.10, 8.11;
$\pi(\gamma_{char}(\mathfrak{U}))$: 8.10; $\pi(\mathfrak{D})$: 8.8;
$\pi(\mathfrak{U})$: 8.11; $\pi_{for\text{-}Being}$: 8.4;
π_{plan}: 8.6;
$\pi_{project}$: 8.2, 8.4, 8.9, 8.10, 8.10;
σ_{exist}: 8.2; τ_{thrown}: 8.5;
φ_{fact}: 8.4; $\varphi_{fact}(\pi_{for\text{-}Being})$: 8.4;
φ_{sake}: 8.3; $\varphi_{sake}(\mathfrak{U})$: 8.3;
$\varphi_{(8.1)}$: 8.2; $\varphi_{(8.9)}$: 8.10;
$\omega_{oneself}$: 8.6;

$\mathfrak{B}$: 8.3, 8.6, 8.11; $\mathfrak{B}(\mathfrak{D})$: 8.3, 8.6, 8.11;
$\mathfrak{B}_{in\text{-}the\text{-}world}$: 8.4; $\mathfrak{C}_{comport}$: 8.6;
$\mathfrak{C}_{comport}(\omega_{oneself})$: 8.6;
$\mathfrak{D}$: 8.3, 8.5, 8.6, 8.7, 8.7, 8.8, 8.8, 8.11;
$\mathfrak{G}_{grasp}$: 8.10, 8.10; $\mathfrak{G}_{grasp}(\mathfrak{U})$: 8.10, 8.10;
$\mathfrak{K}$: 8.5, 8.11; $\mathfrak{K}(\mathfrak{B})$: 8.5;
$\mathfrak{K}(\mathfrak{B}(\mathfrak{D}))$: 8.11;
$\mathfrak{P}_{project}$: 8.5, 8.6, 8.7, 8.10, 8.11;
$\mathfrak{P}_{project}(\alpha)$: 8.10; $\mathfrak{S}_{of\text{-}Being}$: 8.4;
$\mathfrak{S}_{of\text{-}Being}(\gamma_{char}(\mathfrak{U}))$: 8.4; $\mathfrak{T}_{throw}$: 8.10;
$\mathfrak{U}$: 8.1, 8.1, 8.2, 8.3, 8.3, 8.4, 8.4, 8.5, 8.5, 8.9, 8.9; 8.10, 8.10, 8.11, 8.11;
$\mathfrak{U}(\alpha)$: 8.9; $\mathfrak{W}_{cur\_world}$: 8.3;
$\mathfrak{W}_{cur\_world}(\mathfrak{U})$: 8.3; $\mathfrak{W}_{why}$: 8.1;
$\mathfrak{W}_{worldhood}$: 8.3;
$\mathfrak{W}_{worldhood}(\mathfrak{W}_{cur\_world}(\mathfrak{U}))$: 8.3;

$\models$: 8.2, 8.5, 8.7, 8.8, 8.9, 8.10;
$\models_{all\_press\_for}$: 8.1; $\models_{al\_underst}$: 8.8;
$\models_{arr}$: 8.6;
$\models_{as}$: 8.3, 8.4, 8.5, 8.7, 8.8, 8.10, 8.11;
$\models_{as\_long\_as}$: 8.7, 8.8; $\models_{by}$: 8.4;
$\models_{const}$: 8.4; $\models_{contra}$: 8.7;
$\models_{exist\_const}$: 8.4; $\models_{get}$: 8.4;
$\models_{in\_manner}$: 8.10; $\models_{project}$: 8.7, 8.9;

$\models_{proj\_upon}$: 8.3; $\models_{quest}$: 8.1;
$\models_{reduce}$: 8.10; $\models_{take\_away}$: 8.10;
$\models_{think\_out}$: 8.6; $\models_{throw\_before}$: 8.10;
$\models_{thrown}$: 8.5; $\models_{to}$: 8.10;
$\models_{towards}$: 8.6; $\models_{with\_eq\_p}$: 8.3;
$\models_{with\_regard}$: 8.4; $\not\models$: 8.6;
$\not\models_{grasp\_thema}$: 8.9; $\dashv_{accord}$: 8.6;
C: 8.2, 8.10, 8.11; C_{discl}: 8.1;
$\Rightarrow$: 8.2

The eighth paragraph is one of the most complex one. Several informational curiosities can be observed. For instance, α as something which is the object of understanding, that is, a kind of input variable, becomes as object informed within the system of understanding. This is the phenomenon of the so-called understanding internalization of an external, in fact, only observed entity by different ontological entities (for instance, Being, Dasein, projection, projecting, possibilities, significance, etc.) and, certainly, by understanding as the key entity in this context. Various, to the understanding related entities arise as a kind of output variables, including with those, which could be viewed as the internal processing devices. For instance, projection, projecting, significance and understanding, all together arise in the process of Being-there as understanding. And counter-informationally, this is true also for the so-called basic constitutive entities, as there are Being, Dasein, possibilities, etc. as completely open entities in the realm of Being and time.

It is also important to mention the function of informational operators. At this point of understanding, the general informational operator $\models$ (the operational joker of an informational realm) is always an attribute which offers to the involved operands the possibility of the accessibility of the entire informational background. In short, within the formalization of the eighth paragraph, with experience obtained through the informational analysis of the text, new possibilities of understanding a text in an informational sense are coming to the consciousness, opening the perspective of further, yet nontraditional (not merely linguistic or semiotic) kind of informational investigation. Wtthin this perspective, the informational of a text is not only a classical analysis and synthesis of the meaning; it comprises the informational arising.

3.9. *A FORMULA SYSTEM OF THE NINTH PARAGRAPH OF §31 [BT]*

The ninth paragraph constitute the following four formula systems:

(9.1) $(o_{one} \models_{able} ((o_{one} \models_{make} (\iota_{inventory}(\mathfrak{D}) \models_{as} \alpha_{at\_hand})), (o_{one} \models_{list} \Upsilon_{content}(\mathfrak{B}(\mathfrak{D}))))) \Rightarrow ((\varepsilon_{exist}(\pi_{project}) \models_{const} \mathfrak{R}(\mathfrak{B})) \Rightarrow ((\mathfrak{D} \models_{constantly} \mu_{more}(\mathfrak{D})) \models_{than} (\mathfrak{D} \models_{factually}; \models_{factually} \mathfrak{D})))$;

(9.2) $(\mathfrak{D} \not\models_{more\_than} (\mathfrak{D} \models_{factically})) \Leftarrow (\pi_{for\text{-}Being}(\mathfrak{D}) \in_{essen} \varphi_{fact}(\mathfrak{D}))$;

(9.3) $((\mathfrak{D} \models_{as} \mathfrak{B}_{possible}) \not\models_{less} \alpha) \models_{yet} \mathfrak{D}$; $(\mathfrak{D} \models_{exist} \alpha) \models_{in} (\mathfrak{D} \not\models_{yet} \pi_{for\text{-}Being}(\mathfrak{D}))$;

(9.4) $((\mathfrak{B}(\tau_{there}) \models_{say} \mathfrak{B}(\tau_{there})) \models_{say\_with} \mathfrak{U}(\mathfrak{B}(\tau_{there}))) \models_{only\_because} ((\mathfrak{B}(\tau_{there}) \models_{receive} \Upsilon_{const}(\mathfrak{B}(\tau_{there}))) \models_{through} ((\mathfrak{U}; \Upsilon_{char}(\mathfrak{U}) \models_{as} \pi_{project}); (\mathfrak{B}(\tau_{there}) \models; \models \mathfrak{B}(\tau_{there}); \not\dashv \mathfrak{B}(\tau_{there}); \mathfrak{B}(\tau_{there}) \not\dashv)))$

The appearance of operands and operators in the last formula system is the following:

α: 9.3, 9.3; α_{at_hand}: 9.1;
Υ_{char}: 9.4; $\Upsilon_{char}(\mathfrak{U})$: 9.4;
Υ_{const}: 9.4; $\Upsilon_{const}(\mathfrak{B}(\tau_{there}))$: 9.4;
$\Upsilon_{content}$: 9.1; $\Upsilon_{content}(\mathfrak{B}(\mathfrak{D}))$: 9.1;
ε_{exist}: 9.1; $\varepsilon_{exist}(\pi_{project})$: 9.1;
$\iota_{inventory}$: 9.1; $\iota_{inventory}(\mathfrak{D})$: 9.1;
μ_{more}: 9.1; $\mu_{more}(\mathfrak{D})$: 9.1;
o_{one}: 9.1, 9.1; $\pi_{for\text{-}Being}$: 9.2, 9.3;
$\pi_{for\text{-}Being}(\mathfrak{D})$: 9.2, 9.3; $\pi_{project}$: 9.1, 9.4;
τ_{there}: 9.4, 9.4; φ_{fact}: 9.2;
$\varphi_{fact}(\mathfrak{D})$: 9.2;

$\mathfrak{B}$: 9.1, 9.1, 9.4, 9.4; $\mathfrak{B}(\mathfrak{D})$: 9.1;
$\mathfrak{B}(\tau_{there})$: 9.4, 9.4; $\mathfrak{B}_{possible}$: 9.3;
$\mathfrak{D}$: 9.1, 9.1, 9.2, 9.2, 9.3, 9.3;
$\mathfrak{R}$: 9.1; $\mathfrak{R}(\mathfrak{B})$: 9.1;
$\mathfrak{U}$: 9.4, 9.4; $\mathfrak{U}(\mathfrak{B}(\tau_{there}))$: 9.4;

$\models$: 9.4; $\models_{able}$: 9.1;
$\models_{as}$: 9.1, 9.3, 9.4; $\models_{const}$: 9.1;
$\models_{constantly}$: 9.1; $\models_{exist}$: 9.3;
$\models_{factually}$: 9.1, 9.2; $\models_{in}$: 9.3;
$\models_{list}$: 9.1; $\models_{make}$: 9.1;

$\models_{only\_because}$: 9.4; $\models_{receive}$: 9.4;
$\models_{say}$: 9.4; $\models_{say\_with}$: 9.4;
$\models_{than}$: 9.1; $\models_{through}$: 9.4;
$\models_{yet}$: 9.3; $\not\models_{less}$: 9.3;
$\not\models_{more\_than}$: 9.2; $\not\models_{yet}$: 9.3;
$\not\exists$: 9.4; $\in_{essen}$: 9.2;
$\Rightarrow$: 9.1; $\Leftarrow$: 9.2

By this the list of operands and operators is completed. Similar comments as those in case of the previous paragraph can be made.

3.10. *A FORMULA SYSTEM OF THE TENTH PARAGRAPH OF §31 [BT]*

The formal depiction of the tenth paragraph is as follows:

(10.1) $\pi_{project} \models_{always} \circ \models_{pertain} \varphi_{full}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world}))$;
$\pi(\mathfrak{U}) \subset (\mathfrak{U} \models_{as} \pi_{for\text{-}Being})$;
$\pi(\mathfrak{U}) \models_{sketch\_out\_beforehand} (\pi(\mathfrak{U}) \subset ((\rho_{range} \models_{essen} \circ \models_{discl} \mathfrak{W}_{what}) \models_{essen} \circ \models_{discl} \rho_{range}))$;

(10.2) $(\mathfrak{U} \models_{devote} \mathfrak{U}) \models_{prim\_to} \vartheta_{disclose}(\mathfrak{W}_{world})$;
$(\mathfrak{D} \models_{\mathfrak{U}} \mathfrak{D}) \models_{prox} \circ \models_{most\_part} \mathfrak{W}_{world}(\mathfrak{D})$;

(10.3) $(\mathfrak{U} \models_{throw} \mathfrak{U}) \models_{prim\_into} \varphi_{sake}$;
$\mathfrak{D} \models_{exist\_as} \mathfrak{D}$;

(10.4) $(\mathfrak{U} \models_{either} (\mathfrak{U}_{auth} \models_{arise\_out} (\sigma_{self}(o_{one}) \models_{as} \sigma_{self}))) \models_{or} (\mathfrak{U} \models \mathfrak{U}_{inauth})$;

(10.5) $(\iota_{in} \models_{of} \iota_{inauth}) \not\models_{mean} ((\mathfrak{D} \models_{cut\_off} \mathfrak{D}) \models_{from} \sigma_{self}(\mathfrak{D}))$;
$\iota_{inauth} \not\models_{mean} (\mathfrak{D} \models_{\mathfrak{U}} \circ \models_{only} \mathfrak{W}_{world})$;

(10.6) $(\mathfrak{W}_{world} \models_{as} \mathfrak{B}_{in\text{-}the\text{-}world}) \in \mathfrak{B}_{one's\text{-}Self}$;

(10.7) $(\mathfrak{U}_{auth}, \mathfrak{U}_{inauth} \models_{either} \Upsilon_{genuine}) \models_{or} (\mathfrak{U}_{auth}, \mathfrak{U}_{inauth} \not\models \Upsilon_{genuine})$;

(10.8) $\pi \models_{permeate} (\mathfrak{U} \models_{as} \pi_{for\text{-}Being})$;

(10.9) $(\pi_{basic\_1}(\mathfrak{U}) \models_{divert} o_{one}) \Rightarrow (\not\models_{lay\_aside} \pi_{basic\_2}(\mathfrak{U}))$;

(10.10) $((\mathfrak{U} \models_{as} \mathfrak{B}_{in\text{-}the\text{-}world}) \models_{always} \circ \models_{pertain} \varphi_{full}(\vartheta_{disclose}(\mathfrak{D}))) \Rightarrow ((\vartheta_{diversion}(\mathfrak{U}) \models_{as} \mathfrak{W}_{whole}) \models_{exist} \mu_{modif}(\pi_{project}))$;

(10.11) $\mathfrak{U}(\mathfrak{W}_{world}) \models_{always} \circ \models_{\mathfrak{U}} \mathfrak{B}_{in}(\mathfrak{U}(\mathfrak{W}_{world}))$;
$(\mathfrak{U}(\varepsilon_{ex}) \models_{as} \mathfrak{U}(\varepsilon_{ex})) \models_{always} \mathfrak{U}(\mathfrak{W}_{world})$

The survey of the operands and operators of formula system (10) and their appearance in formula subsystems (10.i), i = 1, ... , 11 is given by the following list:

$\Upsilon_{genuine}$: 10.7, 10.7; ε_{ex}: 10.11, 10.11;
$\vartheta_{disclose}$: 10.1, 10.2, 10.10;
$\vartheta_{disclose}(\mathfrak{D})$: 10.10;
$\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world})$: 10.1;
$\vartheta_{disclose}(\mathfrak{W}_{world})$: 10.2; $\vartheta_{diversion}$: 10.10;
$\vartheta_{diversion}(\mathfrak{U})$: 10.10; ι_{in}: 10.5;
ι_{inauth}: 10.5, 10.5; μ_{modif}: 10.10;
$\mu_{modif}(\pi_{project})$: 10.10; o_{one}: 10.4, 10.9;
π: 10.1, 10.8; $\pi(\mathfrak{U})$: 10.1, 10.1;
π_{basic_1}: 10.9; $\pi_{basic\_1}(\mathfrak{U})$: 10.9;
π_{basic_2}: 10.9; $\pi_{basic\_2}(\mathfrak{U})$: 10.9;
$\pi_{for\text{-}Being}$: 10.1, 10.8; $\pi_{project}$: 10.1, 10.10;
ρ_{range}: 10.1, 10.1; σ_{self}: 10.4, 10.4, 10.5;
$\sigma_{self}(\mathfrak{D})$: 10.5; $\sigma_{self}(o_{one})$: 10.4
φ_{full}: 10.1, 10.10; $\varphi_{full}(\vartheta_{disclose}(\mathfrak{D}))$: 10.10;
$\varphi_{full}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world}))$: 10.1;
φ_{sake}: 10.2;

$\mathfrak{B}_{in}$: 10.11;
$\mathfrak{B}_{in}(\mathfrak{U}(\mathfrak{W}_{world}))$: 10.11;
$\mathfrak{B}_{in\text{-}the\text{-}world}$: 10.1, 10.6, 10.10;
$\mathfrak{B}_{one's\text{-}Self}$: 10.6;
$\mathfrak{D}$: 10.2, 10.2, 10.3, 10.3, 10.5, 10.5;
$\mathfrak{U}$: 10.1, 10.1, 10.2, 10.2, 10.3, 10.3, 10.4, 10.4, 10.8, 10.9, 10.10, 10.10, 10.11, 10.11;
$\mathfrak{U}(\varepsilon_{ex})$: 10.11, 10.11; $\mathfrak{U}(\mathfrak{W}_{world})$: 10.11, 10.11;
$\mathfrak{U}_{auth}$: 10.4, 10.7, 10.7;
$\mathfrak{U}_{inauth}$: 10.4, 10.7, 10.7;
$\mathfrak{W}_{what}$: 10.1; $\mathfrak{W}_{whole}$: 10.10;
$\mathfrak{W}_{world}$: 10.2, 10.2, 10.5, 10.6, 10.11, 10.11;
$\mathfrak{W}_{world}(\mathfrak{D})$: 10.2

$\models_{always}$: 10.1, 10.10, 10.11;
$\models_{always} \circ \models_{pertain}$: 10.1, 10.10;
$\models_{always} \circ \models_{\mathfrak{U}}$: 10.11; $\models_{arise\_out}$: 10.4;
$\models_{as}$: 10.1, 10.4, 10.6, 10.8, 10.10, 10.11;
$\models_{cut\_off}$: 10.5;
$\models_{devote}$: 10.2; $\models_{discl}$: 10.1;
$\models_{divert}$: 10.9; $\models_{either}$: 10.4, 10.7;
$\models_{essen}$: 10.1; $\models_{essen} \circ \models_{discl}$: 10.1;
$\models_{exist}$: 10.10; $\models_{exist\_as}$: 10.3;
$\models_{from}$: 10.5; $\models_{most\_part}$: 10.2;
$\models_{of}$: 10.5; $\models_{only}$: 10.5;
$\models_{or}$: 10.4, 10.7; $\models_{permeate}$: 10.8;
$\models_{pertain}$: 10.1, 10.10; $\models_{prim\_to}$: 10.2;
$\models_{prox}$: 10.2; $\models_{prox} \circ \models_{most\_part}$: 10.2;
$\models_{sketch\_out\_beforehand}$: 10.1;

$\models_{throw}$: 10.3; $\models_{\mathfrak{U}}$: 10.2, 10.5, 10.11;
$\models_{\mathfrak{U}} \circ \models_{only}$: 10.5; $\not\models$: 10.7;
$\not\models_{lay\_aside}$: 10.9; $\not\models_{mean}$: 10.5;
$\subset$: 10.1; $\in$: 10.5;
$\Rightarrow$: 10.9, 10.10;
$\circ$: 10.1, 10.2, 10.5, 10.10, 10.11

The analysis of input, output, and processing operands of this formula segment is left to the reader.

3.11. *A FORMULA SYSTEM OF THE ELEVENTH PARAGRAPH OF §31 [BT]*

The eleventh paragraph has only one formula, that is,

(11.1) $(\mathfrak{D} \models_{as} \mathfrak{D}_{fact}) \models_{divert} (\pi_{for\text{-}Being}(\mathfrak{D}) \models_{into} \pi(\mathfrak{U}))$

In this formula only the following operands and operators occur:

π: 11.1; $\pi(\mathfrak{U})$: 11.1;
$\pi_{for\text{-}Being}$: 11.1; $\pi_{for\text{-}Being}(\mathfrak{D}$: 11.1;

$\mathfrak{D}$: 11.1, 11.1; $\mathfrak{D}_{fact}$: 11.1;
$\mathfrak{U}$: 11.1;

$\models_{as}$: 11.1; $\models_{divert}$: 11.1;
$\models_{into}$: 11.1

This completes the operand and operator list of formula system (11). One recognizes how this system is without a particular significance if it is asserted outside of the understanding system as a whole.

3.12. *A FORMULA SYSTEM OF THE TWELFTH PARAGRAPH OF §31 [BT]*

The twelfth paragraph is informationally characterized by the following formula system:

(12.1) $\Upsilon_{char}(\pi_{project}(\mathfrak{U})) \models_{in} (\mathfrak{U} \models_{make\_up} \circ \models_{exist} \sigma_{sight}(\mathfrak{D}))$;
(12.2) $\sigma_{sight}(\mathfrak{D}) \models_{with} (\vartheta_{disclose}(\tau_{there}) \models_{exist})$;
$((\mathfrak{D} \models \sigma_{sight}(\mathfrak{D})) \models_{equi\_p} \circ \models_{always} \mathfrak{B}(\mathfrak{D})) \models_{as}$
$(\Upsilon_{circumsp}(\Upsilon_{concern}), \Upsilon_{consider}(\pi_{solicitude}), \sigma_{sight}((\mathfrak{B} \models_{as} \mathfrak{B}) \models_{sake} (\mathfrak{D} \models; \models \mathfrak{D})))$;
(12.3) $(\varepsilon_{ex} \models_{rel\_prim\_whole} \sigma_{sight}(\mathfrak{D})) \models \tau_{transpar}$;
(12.4) $\tau_{transpar} \models$
$(\mathfrak{K}_{know}(\sigma_{self}) \models \sigma_{sense}(\mathfrak{U}_{well}))$;
$(\tau_{transpar} \not\models \mathfrak{P}_{percept}(\sigma_{self}) \models_{but}$
$(\mathfrak{S}_{seize}(\varphi_{full}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world})))$
$\models_{throughout} \circ \models_{all}$
$(\iota_{item}(\Upsilon_{const}) \models_{essen} (\tau_{transpar}, \mathfrak{U})))$;
(12.5) $(\alpha \models_{sight} \mathfrak{S}_{exist}(\alpha)) \models_{so\_far}$
$(\alpha \models (\tau_{transpar}(\alpha) \models_{with} \circ \models_{equal}$
$\pi_{prim}(\iota_{item} \models \mathfrak{S}_{exist}(\alpha)))$;
$\mathfrak{S}_{exist}(\alpha) \models \mathfrak{B}_{alongside}(\mathfrak{W}_{world}, \mathfrak{B}_{with}(\omega))$

In this system the following operands and operator ar involved:

α: 12.5, 12.5; Υ_{char}: 12.1;
$\Upsilon_{char}(\pi_{project}(\mathfrak{U}))$: 12.1; $\Upsilon_{circumsp}$: 12.2;
$\Upsilon_{circumsp}(\Upsilon_{concern})$: 12.2;
$\Upsilon_{concern}$: 12.2; $\Upsilon_{consider}$: 12.2;
$\Upsilon_{consider}(\pi_{solicitude})$: 12.2;
Υ_{const}: 12.4; ε_{ex}: 12.3;
$\vartheta_{disclose}$: 12.2, 12.4; $\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world})$: 12.4; $\vartheta_{disclose}(\tau_{there})$: 12.2;
ι_{item}: 12.4, 12.5; $\iota_{item}(\Upsilon_{const})$: 12.4;
π_{prim}: 12.5;
$\pi_{prim}(\iota_{item} \models \mathfrak{S}_{exist}(\alpha))$: 12.5;
$\pi_{project}$: 12.1; $\pi_{project}(\mathfrak{U})$: 12.1;
$\pi_{solicitude}$: 12.2; σ_{self}: 12.4, 12.4;
σ_{sense}: 12.4; $\sigma_{sense}(\mathfrak{U}_{well})$: 12.4;
σ_{sight}: 12.1, 12.2, 12.2, 12.3;
$\sigma_{sight}(\mathfrak{D})$: 12.1, 12.2, 12.2, 12.3;
$\sigma_{sight}((\mathfrak{B} \models_{as} \mathfrak{B}) \models_{sake} (\mathfrak{D} \models; \models \mathfrak{D}))$: 12.2;
τ_{there}: 12.2;
$\tau_{transpar}$: 12.3, 12.4, 12.4, 12.5;
$\tau_{transpar}(\alpha)$: 12.5; φ_{full}: 12.4;
$\varphi_{full}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world}))$: 12.4;
ω: 12.5;

$\mathfrak{B}$: 12.2, 12.2; $\mathfrak{B}(\mathfrak{D})$: 12.2;
$\mathfrak{B}_{alongside}$: 12.5;
$\mathfrak{B}_{alongside}(\mathfrak{W}_{world}, \mathfrak{B}_{with}(\omega))$: 12.5;
$\mathfrak{B}_{in\text{-}the\text{-}world}$: 12.4; $\mathfrak{B}_{with}$: 12.5;
$\mathfrak{B}_{with}(\omega)$: 12.5;
$\mathfrak{D}$: 12.1, 12.2, 12.2, 12.3;
$\mathfrak{S}_{exist}$: 12.5; $\mathfrak{S}_{exist}(\alpha)$: 12.5, 12.5;
$\mathfrak{K}_{know}$: 12.4; $\mathfrak{K}_{know}(\sigma_{self})$: 12.4;
$\mathfrak{P}_{percept}$: 12.4; $\mathfrak{P}_{percept}(\sigma_{self})$: 12.4;
$\mathfrak{S}_{seize}$: 12.4;
$\mathfrak{S}_{seize}(\varphi_{full}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world})))$: 12.4;

$\mathfrak{U}$: 12.1, 12.1, 12.4; $\mathfrak{U}_{well}$: 12.4;
$\mathfrak{W}_{world}$: 12.5;

$\models$: 12.2, 12.3, 12.4, 12.5;
$\models_{all}$: 12.4; $\models_{always}$: 12.2;
$\models_{as}$: 12.2; $\models_{but}$: 12.4;
$\models_{in}$: 12.1; $\models_{equal}$: 12.5;
$\models_{equi\_p}$: 12.2; $\models_{equi\_p} \circ \models_{always}$: 12.2;
$\models_{essen}$: 12.4; $\models_{exist}$: 12.1, 12.2;
$\models_{make\_up}$: 12.1; $\models_{make\_up} \circ \models_{exist}$: 12.1;
$\models_{rel\_prim\_whole}$: 12.3; $\models_{sake}$: 12.2;
$\models_{sight}$: 12.5; $\models_{so\_far}$: 12.5;
$\models_{throughout}$: 12.4; $\models_{throughout} \circ \models_{all}$: 12.4
$\models_{with}$: 12.2, 12.5; $\models_{with} \circ \models_{equal}$: 12.5
$\not\models$: 12.4; $\circ$: 12.1, 12.2, 12.4, 12.5

Thus, the operand and operator list for the twelfth paragraph is completed.

3.13. *A FORMULA SYSTEM OF THE THIRTEENTH PARAGRAPH OF §31 [BT]*

The text of the thirteenth paragraph, which includes only one sentence, delivers the following formula system:

(13.1) $\mathfrak{D}_{opaque}(\mathfrak{D}) \not\models_{root} \circ \models_{prim\_sol} \varepsilon_{ego}(\mathfrak{D} \models_{decept} \mathfrak{D})$;
$\mathfrak{D}_{opaque} \models_{root} \circ \models_{just\_as\_much} \lambda_{lack}(\mathfrak{A}_{acquaint} \models_{with} \mathfrak{W}_{world})$

The operand and operator list of the last two formulas is:

ε_{ego}: 13.1; $\varepsilon_{ego}(\mathfrak{D} \models_{decept} \mathfrak{D})$: 13.1;
μ_{lack}: 13.1;
$\mu_{lack}(\mathfrak{A}_{acquaint} \models_{with} \mathfrak{W}_{world})$: 13.1;

$\mathfrak{A}_{acquaint}$: 13.1; $\mathfrak{D}$: 13.1, 13.1;
$\mathfrak{D}_{opaque}$: 13.1, 13.1; $\mathfrak{D}_{opaque}(\mathfrak{D})$: 13.1;
$\mathfrak{W}_{world}$: 13.1;

$\models_{decept}$: 13.1; $\models_{just\_as\_much}$: 13.1;
$\models_{prim\_sol}$: 13.1; $\models_{root}$: 13.1;
$\models_{root} \circ \models_{just\_as\_much}$: 13.1;
$\models_{with}$: 13.1; $\not\models_{root}$: 13.1;
$\not\models_{root} \circ \models_{prim\_sol}$: 13.1; $\circ$: 13.1

This completes the list of operands and operators of the thirteenth paragraph.

3.14. *A FORMULA SYSTEM OF THE FOURTEENTH PARAGRAPH OF §31 [BT]*

The formula system for the fourteenth paragraph is, for instance,

(14.1) $(o_{one} \models_{guard\_against} \mathfrak{U}_{mis}(\varepsilon_{express}(\sigma_{sight}))) \Rightarrow (o_{one} \models_{sure})$;
(14.2) $\varepsilon_{express}(\sigma_{sight}) \models_{corresp} (\Upsilon_{cleared} \models_{char} \vartheta_{disclose}(\tau_{there}))$;
(14.3) $\mathfrak{S}_{see} \not\models_{mean} \circ \models_{just} (\mathfrak{P}_{perceive} \models_{with} \varepsilon_{eyes}(\beta_{body}))$;
$\mathfrak{S}_{see} \not\models_{mean} (\mathfrak{A}_{aware}(\alpha) \models_{pure} \circ \not\models_{sense} ((\alpha \models \alpha_{present\_at\_hand}) \subset \pi_{at\text{-}hand}(\alpha)))$;
(14.4) $(o_{one} \models_{give} \sigma_{sign\_exist}(\sigma_{sight})) \models_{draw\_upon} \circ \models_{merely} \varphi_{feature}(\pi_{peculiar}(\mathfrak{S}_{see}))$;
$\alpha \models_{encounter} \circ \not\models_{conceal} (\alpha \models_{access} \varphi_{feature}(\pi_{peculiar}(\mathfrak{S}_{see})))$;
(14.5) $(\sigma_{sense} \models_{do} \varphi_{(14.4)} \subset \vartheta_{domain}(\vartheta_{disclose}(\sigma_{sense}))$;
(14.6) $\beta_{begin}(\tau_{trad}(\pi_{philo})) \models_{orient} \circ (\models_{prim} \circ \models_{towards}) (\mathfrak{S}_{see} \models_{as} (\mathfrak{W}_{way}(\mathfrak{A}_{access}) \models_{to} \alpha, \mathfrak{B}))$;
(14.7) $\tau_{trad} \models_{connect} (o_{one} \models_{formalize} (\sigma_{sight}, \mathfrak{S}_{see} \models_{enough} \circ \models_{obtain} (\tau_{term\_uni} \models_{char} (\mathfrak{A}_{access} \models_{to} (\alpha, \mathfrak{B} \models_{as} (\mathfrak{A}_{access} \subset \mathfrak{G}_{general})))))$

In this formula system the following operands and operators occur:

α: 14.3, 14.3, 14.4, 14.4, 14.6, 14.7;
$\alpha_{present\_at\_hand}$: 14.3; β_{begin}: 14.6;
$\beta_{begin}(\tau_{trad}(\pi_{philo}))$: 14.6;
β_{body}: 14.3; $\Upsilon_{cleared}$: 14.2;
$\varepsilon_{express}$: 14.1, 14.2;
$\varepsilon_{express}(\sigma_{sight})$: 14.1, 14.2;
ε_{eyes}: 14.3; $\varepsilon_{eyes}(\beta_{body})$: 14.3;
$\vartheta_{disclose}$: 14.2, 14.5; $\vartheta_{disclose}(\tau_{there})$: 14.2;
ϑ_{domain}: 14.5;
$\vartheta_{domain}(\vartheta_{disclose}(\sigma_{sense}))$: 14.5;
o_{one}: 14.1, 14,1, 14.4, 14.7;
$\pi_{at\text{-}hand}$: 14.3; $\pi_{at\text{-}hand}(\alpha)$: 14.3;
$\pi_{peculiar}$: 14.4, 14.4;
$\pi_{peculiar}(\mathfrak{S}_{see})$: 14.4, 14.4;

π_{philo}: 14.6; σ_{sign_exist}: 14.4;
$\sigma_{sign\_exist}(\sigma_{sight})$: 14.4; σ_{sense}: 14.5;
σ_{sight}: 14.1, 14.4, 14.7; τ_{term_uni}: 14.7; τ_{there}: 14.2;
τ_{trad}: 14.6, 14.7; $\tau_{trad}(\pi_{philo})$: 14.6;
$\varphi_{feature}$: 14.4, 14.4;
$\varphi_{feature}(\pi_{peculiar}(\mathfrak{S}_{see}))$: 14.4, 14.4;
$\varphi_{(14.4)}$: 14.5;

$\mathfrak{A}_{access}$: 14.6, 14.7, 14.7;
$\mathfrak{A}_{aware}$: 14.3; $\mathfrak{A}_{aware}(\alpha)$: 14.3;
$\mathfrak{B}$: 14.6, 14.7; $\mathfrak{G}_{general}$: 14.7;
$\mathfrak{P}_{perceive}$: 14.3;
$\mathfrak{S}_{see}$: 14.3, 14.3, 14.4, 14.4, 14.6, 14.7;
$\mathfrak{U}_{mis}$: 14.1;
$\mathfrak{U}_{mis}(\varepsilon_{express}(\sigma_{sight}))$: 14.1;
$\mathfrak{W}_{way}$: 14.6; $\mathfrak{W}_{way}(\mathfrak{A}_{access})$: 14.6;

$\models$: 14.3; $\models_{access}$: 14.4;
$\models_{as}$: 14.6, 14.7; $\models_{char}$: 14.2, 14.7;
$\models_{connect}$: 14.7; $\models_{corresp}$: 14.2;
$\models_{do}$: 14.5; $\models_{draw\_upon}$: 14.4;
$\models_{draw\_upon} \circ \models_{merely}$: 14.4;
$\models_{encounter}$: 14.4;
$\models_{encounter} \circ \not\models_{conceal}$: 14.4;
$\models_{enough}$: 14.7; $\models_{enough} \circ \models_{obtain}$: 14.7;
$\models_{formalize}$: 14.7; $\models_{give}$: 14.4;
$\models_{guard\_against}$: 14.1; $\models_{just}$: 14.3;
$\models_{merely}$: 14.4; $\models_{obtain}$: 14.7;
$\models_{orient}$: 14.6;
$\models_{orient} \circ (\models_{prim} \circ \models_{towards})$: 14.6;
$\models_{prim}$: 14.6; $\models_{prim} \circ \models_{towards}$: 14.6;
$\models_{pure}$: 14.3; $\models_{pure} \circ \not\models_{sense}$: 14.3;
$\models_{sure}$: 14.1; $\models_{to}$: 14.6, 14.7;
$\models_{towards}$: 14.6; $\models_{with}$: 14.3;
$\not\models_{conceal}$: 14.4; $\not\models_{mean}$: 14.3;
$\not\models_{mean} \circ \models_{just}$: 14.3; $\not\models_{sense}$: 14.3;
$\subset$: 14.3, 14.5, 14.7; $\Rightarrow$: 14.1;
$\circ$: 14.3, 14.4, 14.6, 14.7;

This formula system shows an intensive informing in understanding something α by different aspects (for instance, by misunderstanding, present-at-hand, awareness, seeing, signification, etc.). For a further analysis and informational extension of the system, the list of operands and operators may be instructive showing simultaneously the perplexity of operands in different formulas and the operands' operational attributing, that is, occurrences of operators as attributes of operands to which the operators pertain.

3.15. *A FORMULA SYSTEM OF THE FIFTEENTH PARAGRAPH OF §31 [BT]*

The fifteenth paragraph has the following formula system:

(15.1) $(o_{one} \models_{show} ((\sigma_{sight} \subset_{ground} \circ \subset_{prim} \mathfrak{U}), (\gamma_{circumsp}(\gamma_{concern}) \models (\mathfrak{U} \models_{as} \gamma_{comm}(\sigma_{sense})))) \Rightarrow ((o_{one} \models_{deprive} (\pi_{pure}(\iota_{intuition} \models_{of} \pi_{prior}(\pi_{pure}(\iota_{intuition})))) \models_{corresp} \circ \models_{noe} \pi_{prior}(\pi_{at\text{-}hand} \subset \tau_{trad}(o_{ontology}))$;

(15.2) $\iota_{intuition}, \mathfrak{T}_{think} \models_{already} \vartheta_{derive\_remote}(\mathfrak{U})$;

(15.3) $\varphi_{phenomenal}(\iota_{intuition}(\varepsilon_{essence})) \subset_{ground} \mathfrak{U}_{exist}$;

(15.4) $(\gamma_{concept\_ex}(\mathfrak{B}), \sigma_{structure}(\mathfrak{B}) \models_{as} \lambda_{logic}(\varphi_{phenomenal})) \Rightarrow (o_{one} \models_{decide} \mathfrak{K}(\mathfrak{S}_{see}(\mathfrak{B})))$

The list of operands and operators is the following:

$\gamma_{circumsp}$: 15.1; $\gamma_{circumsp}(\gamma_{concern})$: 15.1;
γ_{comm}: 15.1; $\gamma_{comm}(\sigma_{sense})$: 15.1;
$\gamma_{concept\_ex}$: 15.4; $\gamma_{concept\_ex}(\mathfrak{B})$: 15.4;
$\gamma_{concern}$: 15.1; $\varepsilon_{essence}$: 15.3;
$\vartheta_{derive\_remote}$: 15.2; $\vartheta_{derive\_remote}(\mathfrak{U})$: 15.2;
$\iota_{intuition}$: 15.1, 15.1, 15.2, 15.3;
$\iota_{intuition}(\varepsilon_{essence})$: 15.3;
λ_{logic}: 15.4; $\lambda_{logic}(\varphi_{phenomenal})$: 15.4;
o_{one}: 15.1, 15.1, 15.4; $o_{ontology}$: 15.1;
π_{at_hand}: 15.1; π_{prior}: 15.1, 15.1;
$\pi_{prior}(\pi_{pure}(\iota_{intuition}))$: 15.1;
$\pi_{prior}(\pi_{at\text{-}hand} \subset \tau_{trad}(o_{ontology}))$: 15.1;
π_{pure}: 15.1, 15.1; $\pi_{pure}(\iota_{intuition})$: 15.1;
$\pi_{pure}(\iota_{intuition} \models_{of} \pi_{prior}(\pi_{pure}(\iota_{intuition})))$: 15.1;
σ_{sense}: 15.1; σ_{sight}: 15.1;
$\sigma_{structure}$: 15.4; $\sigma_{structure}(\mathfrak{B})$: 15.4;
τ_{trad}: 15.1; $\tau_{trad}(o_{ontology})$: 15.1;
$\varphi_{phenomenal}$: 15.3, 15.4;
$\varphi_{phenomenal}(\iota_{intuition}(\varepsilon_{essence}))$: 15.3;

$\mathfrak{B}$: 15.4, 15.4; $\mathfrak{K}$: 15.4;
$\mathfrak{K}(\mathfrak{S}_{see}(\mathfrak{B}))$: 15.4; $\mathfrak{S}_{see}$: 15.4;
$\mathfrak{S}_{see}(\mathfrak{B})$: 15.4; $\mathfrak{T}_{think}$: 15.2;
$\mathfrak{U}$: 15.1, 15.1, 15.2; $\mathfrak{U}_{exist}$: 15.3;

$\models$: 15.1; $\models_{already}$: 15.1;
$\models_{as}$: 15.1, 15.4; $\models_{corresp}$: 15.1;
$\models_{corresp} \circ \models_{noe}$: 15.1; $\models_{decide}$: 15.4;
$\models_{deprive}$: 15.1; $\models_{noe}$: 15.1;
$\models_{of}$: 15.1; $\models_{show}$: 15.1;
$\subset_{ground}$: 15.3; $\subset_{prim}$: 15.1;
$\Rightarrow$: 15.1, 15.4; $\circ$: 15.1

Thus, the operand and operator list for this paragraph is completed.

3.16. *A FORMULA SYSTEM OF THE SIXTEENTH PARAGRAPH OF §31 [BT]*

The collection of formulas for the sixteenth paragraph is as follows:

(16.1) $(\vartheta_{disclose}(\tau_{there}) \subset \mathfrak{U}) \models$
$(\vartheta_{disclose}(\tau_{there}) \models \mathfrak{W}_{way}(\pi_{for\text{-}Being}(\mathfrak{D})))$;

(16.2) $(\vartheta_{disclose}(\mathfrak{B}) \models_{general}) \subset_{lie}$
$((\mathfrak{B}(\vartheta_{disclose}) \models_{project} \circ \models_{upon}$
$(\varphi_{sake}, \sigma_{sign}(\mathfrak{W}_{world}))) \subset \mathfrak{W}_{way})$;

(16.3) $\mathfrak{U}(\mathfrak{B}) \models_{have} \circ ((\models_{already} \circ \models_{been}) \circ \models_{take})$
$((\mathfrak{G}_{grant} \subset \mathfrak{P}_{project}) \models_{upon} \pi)$;

(16.4) $((\models_{\mathfrak{U}} \mathfrak{B}) \subset \pi_{project}) \models_{though}$
$(\not\models_{onto} \circ \models_{conceive} \mathfrak{B})$;

(16.5) $(\mathfrak{K}(\mathfrak{B}(\alpha)) \models_{essen} \pi_{project}(\mathfrak{B}_{in\text{-}the\text{-}world}))$
$\Rightarrow (\alpha \models_{have} (\mathfrak{U}(\mathfrak{B}) \models_{const} \mathfrak{B}(\alpha)))$;

(16.6) $(\sigma_{stage}(\varepsilon_{early}) \models_{posit} \circ \models_{dogma} \mathfrak{W}_{what})$
$\models_{now} \circ \models_{get} \varepsilon_{exhibit}(\mathfrak{C}_{const}(\mathfrak{B}))$;
$((\mathfrak{D} \subset \mathfrak{C}_{const}(\mathfrak{B})) \models_{as} \mathfrak{U}) \models$
$\tau_{there}(\mathfrak{C}_{const}(\mathfrak{B}))$;

(16.7) $\mathfrak{I}_{inter}(\mathfrak{T}_{temporal}(\mathfrak{B})) \models_{clarify}$
$(\mu_{exist} \models_{of} \mathfrak{U}(\mathfrak{B}))$;
$(\mu_{exist} \models_{of} \mathfrak{U}(\mathfrak{B})) \not\models_{satisfy} \circ \subset_{clarify}$
$\lambda_{limit}(\iota_{investigate})$;
$\varphi_{(1.1)}, \varphi_{(1.2)}, \cdots, \varphi_{(18.1)} \models \iota_{investigate}$

Operands and operators of this formula collection are:

α: 16.5, 16.5; ε_{early}: 16.6;
$\varepsilon_{exhibit}$: 16.6; $\varepsilon_{exhibit}(\mathfrak{C}_{const}(\mathfrak{B}))$: 16.6;
$\vartheta_{disclose}$: 16.1, 16.1, 16.2, 16.2;
$\vartheta_{disclose}(\mathfrak{B})$: 16.2;
$\vartheta_{disclose}(\tau_{there})$: 16.1, 16.1;
$\iota_{investigate}$: 16.7, 16.7; λ_{limit}: 16.7;
$\lambda_{limit}(\iota_{investigate})$: 16.7; μ_{exist}: 16.7, 16.7;
π: 16.3; $\pi_{for\text{-}Being}$: 16.1;
$\pi_{for\text{-}Being}(\mathfrak{D})$: 16.1; $\pi_{project}$: 16.4, 16.5;
$\pi_{project}(\mathfrak{B}_{in\text{-}the\text{-}world})$: 16.5;
σ_{sign}: 16.2; $\sigma_{sign}(\mathfrak{W}_{world})$: 16.2;
σ_{stage}: 16.6; $\sigma_{stage}(\varepsilon_{early})$: 16.6;
τ_{there}: 16.1, 16.1, 16.6;
φ_{sake}: 16.2;
$\varphi_{(1.1)}, \varphi_{(1.2)}, \cdots, \varphi_{(18.1)}$: 16.7;

$\mathfrak{B}$: 16.2, 16.2, 16.3, 16.4, 16.4, 16.5, 16.5, 16.6, 16.7, 16.7;
$\mathfrak{B}(\alpha)$: 16.5, 16.5; $\mathfrak{B}(\vartheta_{disclose})$: 16.2;
$\mathfrak{B}_{in\text{-}the\text{-}world}$: 16.5; $\mathfrak{C}_{const}$: 16.6, 16.6;
$\mathfrak{C}_{const}(\mathfrak{B})$: 16.6, 16.6; $\mathfrak{D}$: 16.1, 16.6;
$\mathfrak{G}_{grant}$: 16.3; $\mathfrak{I}_{inter}$: 16.7;
$\mathfrak{I}_{inter}(\mathfrak{T}_{temporal}(\mathfrak{B}))$: 16.7;
$\mathfrak{K}$: 16.5; $\mathfrak{K}(\mathfrak{B}(\alpha))$: 16.5;
$\mathfrak{P}_{project}$: 16.3; $\mathfrak{T}_{temporal}$: 16.7;
$\mathfrak{T}_{temporal}(\mathfrak{B})$: 16.7
$\mathfrak{U}$: 16.1, 16.3, 16.5, 16.6, 16.7, 16.7;
$\mathfrak{U}(\mathfrak{B})$: 16.3, 16.5, 16.7, 16.7;
$\mathfrak{W}_{way}$: 16.1, 16.2;
$\mathfrak{W}_{way}(\pi_{for\text{-}Being}(\mathfrak{D}))$: 16.1;
$\mathfrak{W}_{what}$: 16.6; $\mathfrak{W}_{world}$: 16.2;

$\models$: 16.1, 16.6, 16.7; $\models_{already}$: 16.3;
$\models_{already} \circ \models_{been}$: 16.3;
$(\models_{already} \circ \models_{been}) \circ \models_{take}$: 16.3;
$\models_{as}$: 16.6; $\models_{been}$: 16.3;
$\models_{clarify}$: 16.7; $\models_{conceive}$: 16.4;
$\models_{dogma}$: 16.6; $\models_{essen}$: 16.5;
$\models_{general}$: 16.2; $\models_{get}$: 16.6;
$\models_{have}$: 16.3, 16.5;
$\models_{have} \circ ((\models_{already} \circ \models_{been}) \circ \models_{take})$: 16.3;
$\models_{now}$: 16.6; $\models_{now} \circ \models_{get}$: 16.6;
$\models_{of}$: 16.7; $\models_{posit}$: 16.6;
$\models_{posit} \circ \models_{dogma}$: 16.6; $\models_{project}$: 16.2;
$\models_{project} \circ \models_{upon}$: 16.2; $\models_{take}$: 16.3;
$\models_{though}$: 16.4; $\models_{\mathfrak{U}}$: 16.4;
$\models_{upon}$: 16.2, 16.3; $\not\models_{onto}$: 16.4;
$\not\models_{onto} \circ \models_{conceive}$: 16.4; $\not\models_{satisfy}$: 16.7;
$\not\models_{satisfy} \circ \subset_{clarify}$: 16.7;
$\subset$: 16.1, 16.2, 16.3, 16.4, 16.6;
$\subset_{clarify}$: 16.7; $\subset_{lie}$: 16.2;
$\Rightarrow$: 16.5;
$\circ$: 16.2, 16.3, 16.4, 16.6, 16.7

This completes the list of operands and operator of informational system (16).

.3.17. *A FORMULA SYSTEM OF THE SEVENTEENTH PARAGRAPH OF §31 [BT]*

The seventeenth paragraph is characterized by the following informational formula system:

(17.1) $((\mu_{\text{multi}}(\mathfrak{S}_{\text{mind}}), \mathfrak{U} \models_{\text{as}} \mu_{\text{multi}}(\varepsilon_{\text{exist}})) \models_{\text{char}} \pi_{\text{prim}}(\vartheta_{\text{disclose}}(\mathfrak{B}_{\text{in-the-world}}))$;

(17.2) $(\mathfrak{W}_{\text{way}}(\mathfrak{M}) \models \mathfrak{D}) \models_{\text{see}} \pi(\mathfrak{D} \models ; \models \mathfrak{D})$;

(17.3) $(\mathfrak{D} \subset \pi_{\text{project}}(\vartheta_{\text{disclose}}(\pi(\mathfrak{D})))) \models_{\text{have}} \circ \models_{\text{always}} \mathfrak{M}$;

(17.4) $\pi_{\text{project}}(\pi_{\text{for-Being}}(\mathfrak{D})) \models_{\text{deliver}} (\varphi_{\text{fact}}(\tau_{\text{throw}}(\mathfrak{D})) \subset \tau_{\text{there}})$;

(17.5) $((\mathfrak{B}(\mathfrak{D}) \not\models_{\text{more}} \circ \models_{\text{now}} \varepsilon_{\text{enigma}}) \models_{\text{that}} ((o_{\text{one}} \models_{\text{explicate}} \varepsilon_{\text{exist}}(\gamma_{\text{const}}(\mathfrak{B}(\tau_{\text{there}})))) \subset \sigma_{\text{sense}}(\tau_{\text{throw}}(\pi_{\text{project}})))) \models \mathfrak{Q}_{\text{quest}}(\mathfrak{B}(\mathfrak{D}))$;

(17.6) $(\mathfrak{U}_{\text{answer}} \models_{\text{to}} \mathfrak{Q}_{\text{quest}}(\mathfrak{B}(\mathfrak{D}))) \models \iota_{\text{indeed}}$;

(17.7) $(\varphi_{\text{full}}(\gamma_{\text{char}}(\varepsilon_{\text{enigma}}(\mathfrak{B}(\mathfrak{D})))) \models_{\text{first}} \circ (\models_{\text{let}} \circ \models_{\text{emerge}})) \models_{\text{even}} (((o_{\text{one}} \models_{\text{come}} \gamma_{\text{genuine}}(\beta_{\text{breakdown}}) \models_{\text{over}} \sigma_{\text{solution}}(\mathfrak{B})) \Rightarrow (o_{\text{one}} \models_{\text{formulate}} \nu_{\text{anew}}(\mathfrak{Q}_{\text{quest}} \models_{\text{about}} (\mathfrak{B} \models_{\text{of}} \tau_{\text{throw}}(\pi_{\text{project}}(\mathfrak{B}_{\text{in-the-world}}))))))$

The operand and operator list is the following:

$\beta_{\text{breakdown}}$: 17.7; γ_{char}: 17.7;
$\gamma_{\text{char}}(\varepsilon_{\text{enigma}}(\mathfrak{B}(\mathfrak{D})))$: 17.7;
γ_{const}: 17.5; $\gamma_{\text{const}}(\mathfrak{B}(\tau_{\text{there}}))$: 17.5;
γ_{genuine}: 17.7;
$\gamma_{\text{genuine}}(\beta_{\text{breakdown}})$: 17.7;
$\varepsilon_{\text{enigma}}$: 17.5, 17.7; $\varepsilon_{\text{enigma}}(\mathfrak{B}(\mathfrak{D}))$: 17.7;
$\varepsilon_{\text{exist}}$: 17.1, 17.5;
$\varepsilon_{\text{exist}}(\gamma_{\text{const}}(\mathfrak{B}(\tau_{\text{there}})))$: 17.5;
$\vartheta_{\text{disclose}}$: 17.1, 17.3; $\vartheta_{\text{disclose}}(\pi(\mathfrak{D}))$: 17.3;
$\vartheta_{\text{disclose}}(\mathfrak{B}_{\text{in-the-world}})$: 17.1;
ι_{indeed}: 17.6; μ_{multi}: 17.1, 17.1;
$\mu_{\text{multi}}(\varepsilon_{\text{exist}})$: 17.1; $\mu_{\text{multi}}(\mathfrak{S}_{\text{mind}})$: 17.1;
ν_{anew}: 17.7; $\nu_{\text{anew}}(\mathfrak{Q}_{\text{quest}} \models_{\text{about}} (\mathfrak{B} \models_{\text{of}} \tau_{\text{throw}}(\pi_{\text{project}}(\mathfrak{B}_{\text{in-the-world}}))))$: 17.7;
o_{one}: 17.5, 17.7, 17.7; π: 17.2, 17.3;
$\pi(\mathfrak{D})$: 17.3; $\pi(\mathfrak{D} \models ; \models \mathfrak{D})$: 17.2;
$\pi_{\text{for-Being}}$: 17.4; $\pi_{\text{for-Being}}(\mathfrak{D})$: 17.4;
π_{prim}: 17.1;
$\pi_{\text{prim}}(\vartheta_{\text{disclose}}(\mathfrak{B}_{\text{in-the-world}}))$: 17.1;
π_{project}: 17.3, 17.4, 17.5, 17.7;
$\pi_{\text{project}}(\vartheta_{\text{disclose}}(\pi(\mathfrak{D})))$: 17.3;
$\pi_{\text{project}}(\pi_{\text{for-Being}}(\mathfrak{D}))$: 17.4;
$\pi_{\text{project}}(\mathfrak{B}_{\text{in-the-world}})$: 17.7;
σ_{sense}: 17.5;
$\sigma_{\text{sense}}(\tau_{\text{throw}}(\pi_{\text{project}}))$: 17.5;
σ_{solution}: 17.7; $\sigma_{\text{solution}}(\mathfrak{B})$: 17.7;
τ_{there}: 17.4, 17.5; τ_{throw}: 17.4, 17.7;
$\tau_{\text{throw}}(\pi_{\text{project}}(\mathfrak{B}_{\text{in-the-world}}))$: 17.7;
$\tau_{\text{throw}}(\mathfrak{D})$: 17.4; $\tau_{\text{throw}}(\pi_{\text{project}})$: 17.5;
φ_{fact}: 17.4; $\varphi_{\text{fact}}(\tau_{\text{throw}}(\mathfrak{D}))$: 17.4;
φ_{full}: 17.7;
$\varphi_{\text{full}}(\gamma_{\text{char}}(\varepsilon_{\text{enigma}}(\mathfrak{B}(\mathfrak{D}))))$: 17.7;

$\mathfrak{U}_{\text{answer}}$: 17.6;
$\mathfrak{B}$: 17.5, 17.5, 17.6, 17.7, 17.7;
$\mathfrak{B}(\tau_{\text{there}})$: 17.5;
$\mathfrak{B}(\mathfrak{D})$: 17.5, 17.5, 17.6, 17.7;
$\mathfrak{B}_{\text{in-the-world}}$: 17.1, 17.7;
$\mathfrak{D}$: 17.2, 17.2, 17.3, 17.3, 17.4, 17.4, 17.5, 17.5, 17.6, 17.7; $\mathfrak{M}$: 17.2, 17.3;
$\mathfrak{Q}_{\text{quest}}$: 17.5, 17.6, 17.7;
$\mathfrak{Q}_{\text{quest}}(\mathfrak{B}(\mathfrak{D}))$: 17.5, 17.6;
$\mathfrak{S}_{\text{mind}}$: 17.1; $\mathfrak{U}$: 17.1;
$\mathfrak{W}_{\text{way}}$: 17.2; $\mathfrak{W}_{\text{way}}(\mathfrak{M})$: 17.2;

$\models$: 17.2, 17.5, 17.6; $\models_{\text{about}}$: 17.7;
$\models_{\text{always}}$: 17.3; $\models_{\text{as}}$: 17.1;
$\models_{\text{char}}$: 17.1; $\models_{\text{come}}$: 17.7;
$\models_{\text{deliver}}$: 17.4; $\models_{\text{emerge}}$: 17.7;
$\models_{\text{even}}$: 17.7; $\models_{\text{explicate}}$: 17.5;
$\models_{\text{first}}$: 17.7;
$\models_{\text{first}} \circ (\models_{\text{let}} \circ \models_{\text{emerge}})$: 17.7;
$\models_{\text{formulate}}$: 17.7; $\models_{\text{have}}$: 17.3;
$\models_{\text{have}} \circ \models_{\text{always}}$: 17.3; $\models_{\text{let}}$: 17.7;
$\models_{\text{let}} \circ \models_{\text{emerge}}$: 17.7; $\models_{\text{now}}$: 17.5;
$\models_{\text{of}}$: 17.7; $\models_{\text{over}}$: 17.7;
$\models_{\text{see}}$: 17.2; $\models_{\text{that}}$: 17.5;
$\models_{\text{to}}$: 17.6; $\not\models_{\text{more}}$: 17.5
$\not\models_{\text{more}} \circ \models_{\text{now}}$: 17.5; $\subset$: 17.3, 17.4, 17.5;
$\Rightarrow$: 17.7; $\circ$: 17.3; 17.5

Thus, the operand and operator list for the seventeenth paragraph is completed.

3.18. *A FORMULA SYSTEM OF THE EIGHTEENTH PARAGRAPH OF §31 [BT]*

For the last, eighteenth paragraph the following formula system is obtained:

(18.1) $((\models_{\text{but}} \circ (\models_{\text{first}} \circ \models_{\text{even}}) (o_{\text{one}} \models_{\text{just}} \circ \models_{\text{bring}} \varepsilon_{\text{everyday}}(\mathfrak{R}(\mathfrak{B}))$

$\models_{into}\ \nu_{view})$;
$(\Re(\mathfrak{B}) \models (\mathfrak{U} \models_{with} \Theta_{mind}))$;
$(((o_{one} \models_{do} \sigma_{so}) \models_{in}$
$\varphi_{phenomenal}(\alpha_{adequate}(\mathfrak{W}_{way})))$
$\models_{to} \varphi_{full}(\vartheta_{disclose}(\tau_{there}))) \Rightarrow$
$(o_{one} \models_{must} \circ \models_{work\_out}$
$\Upsilon_{concrete}(\tau_{this}(\mu_{multi}(\varepsilon_{exist}))))$

The operands and operators occurring in this system are:

$\alpha_{adequate}$: 18.1; $\alpha_{adequate}(\mathfrak{W}_{way})$: 18.1;
$\Upsilon_{concrete}$: 18.1;
$\Upsilon_{concrete}(\tau_{this}(\mu_{multi}(\varepsilon_{exist}))$: 18.1;
$\varepsilon_{everyday}$: 18.1; $\varepsilon_{everyday}(\Re(\mathfrak{B}))$: 18.1;
ε_{exist}: 18.1; μ_{multi}: 18.1;
$\vartheta_{disclose}$: 18.1; $\vartheta_{disclose}(\tau_{there})$: 18.1;
$\mu_{multi}(\varepsilon_{exist})$: 18.1; ν_{view}: 18.1;
o_{one}: 18.1, 18.1; σ_{so}: 18.1;
τ_{there}: 18.1; τ_{this}: 18.1;
$\tau_{this}(\mu_{multi}(\varepsilon_{exist}))$: 18.1;
φ_{full}: 18.1;
$\varphi_{full}(\vartheta_{disclose}(\tau_{there}))$: 18.1;
$\varphi_{phenomenal}$: 18.1;
$\varphi_{phenomenal}(\alpha_{adequate}(\mathfrak{W}_{way}))$: 18.1;

$\mathfrak{B}$: 18.1, 18.1; $\Re$: 18.1, 18.1;
$\Re(\mathfrak{B})$: 18.1, 18.1; Θ_{mind}: 18.1;
$\mathfrak{U}$: 18.1; $\mathfrak{W}_{way}$: 18.1;

$\models$: 18.1; $\models_{bring}$: 18.1;
$\models_{but}$: 18.1;
$\models_{but} \circ (\models_{first} \circ \models_{even})$: 18.1;
$\models_{do}$: 18.1; $\models_{even}$: 18.1;
$\models_{first}$: 18.1; $\models_{first} \circ \models_{even}$: 18.1;
$\models_{in}$: 18.1; $\models_{into}$: 18.1;
$\models_{just}$: 18.1; $\models_{just} \circ \models_{bring}$: 18.1;
$\models_{must}$: 18.1; $\models_{must} \circ \models_{work\_out}$: 18.1;
$\models_{to}$: 18.1; $\models_{with}$: 18.1;
$\models_{work\_out}$: 18.1; $\Rightarrow$: 18.1;
$\circ$: 18.1

Thus, we arrived to the end of the symbolical survey of the paragraphs taken from the section §31 of the Heidegger's *Being-there as Understanding* in his epochal work *Being and Time.*

The task which remains is to give a final formal picture of the system which models Section 31 as a whole. Further, we are obliged to comment the significance of the transition from a text to an informational formula system and point out possibilities of the initial formal system obtained from verbal text and system's informational development.

4. SECTION 31 (§31 OF [BT]) AS AN INFORMING INFORMATIONAL SYSTEM

4.1. *INFORMATIONAL FORMALIZATION OF TEXT*

What could be the difference between a written text (a sequence of sentences expressed in a natural language) and to this text corresponding formula system expressed in the proposed informational language? As we usually say, the written text has its meaning which can be differently understood by different observers. An observant text understanding (belonging to someone or to something) informs its specific text meaning in the so-called intelligent metaphysical informational cycle where informing, counter-informing, and embedding of text information take place. This meaning is constituted in several, serial and parallel, traditionally atomized (semiotic) ways, for instance, through semantics, syntax, and pragmatics of the given text. Of these three, pragmatics is that atomized semiotic component which could reach farthest in its specific informational understanding, informing the meaning of text, by its own, individual semantics and syntax, within its own, specifically oriented, intended, and significantly structured informational system. What does such position mean in concern to an informational formula system which in its initial state models the originally written text?

A formula is a sequence of operands and operators figuring as informationally active and passive informational entities simultaneously. Through formalization, a word or a word group in a sentence becomes the informing (arising, changing, variable) entity within the corresponding formula or formula system. It is no more an informationally static (meaningly constant), for all times written and linguistically determined entity, for which one can compose, decompose, and construct meaning solely by means of a dictionary,

however, in his or her own mind. Through transformation of words or word groups of a text into informing entities, the introduced operands and operators begin to inform in their own (spontaneous and circular) way. And that situation only confirms the attitude of a text observer when the text is read, studied, and/or thought through an observing perspective. Through formalization, words and word groups begin to act, perform, interact, observe, and mutually observe each other informationally in the full sense of informing of information. Through informational conceptualization, the previously static text components now inform and are informed, impact and are impacted within an informational formula system.

For instance, by the introduction of an informational formula system, a new happening arises which never take place in case of a computer program. An informational formula arises, changes, is variable, while the computer program performs as a text (recipe, fixed procedure). The informational formula is simultaneously a written and an informationally executing program, while the computer program has two different and independent states (natures): its record as a machine program on the hard disc and its record as an executing procedure within the high-speed memory; both records must not be changed. If the first record is changed, the program is modified by the user; if the second one is changed during the program execution, this change is irregular, illegal, malfunctioning, and unforeseeable, for instance, virological.

After this discussion it is possible to feel the difference which exists between a written text and to the text corresponding formula system, which is an initial situation, before this system with its operand and operator entities begins to inform in its own way, that is, spontaneously and circularly. This informing models the way of happening in the mind of observer when a text is read, studied, and envisioned repeatedly into details, delivering informational interaction, relation, and interdependence through arising structuring and organization. A formula system which initially modeled a text becomes an informing system and entities—originally words and word groups—become informing entities (operands and operators) which change their meaning (semantics, syntax, and pragmatics) and let it arise in an informationally structured and organized manner.

One sees how reading a text is not only a semiotic affair in the sense of a traditional (scientific) semantic, syntactic, and pragmatic atomization; reading a text has its own spontaneous and circular nature of an open informational system in which new, different, and contradictory semantics, syntax, and pragmatics can arise. »Words« begin to impact »other words« and »themselves«, they become informational relatedness impacting the arising of the instantaneous formula system. However, through informing, this relatedness grows, changes and vanishes, becomes influenced by the »affectedness«, that is, »informingness« of the system in question. Thus, modeling a text by a formula system is not only a traditionally semiotic problem; the development of the formula system hides its own informational potentiality of arising, that is, its own, unforeseeable informational nature—of understanding.

4.2. *SECTION 31 AS A WHOLE*

At last, we have to integrate the paragraph informational systems (1) to (18) into the unique section system. In this way we obtain a unique informational program for which operand and operator entities must be informationally determined. At the end of this essay a dictionary of operands and operators is listed and in this dictionary operands can be additionally formalized (in fact, informationally determined in an entity metaphysical-parallel way) to the necessary extent. The verbal explanations in three different natural languages can be extended from single words or idioms to informational structures corresponding to the meanings of words found in different dictionaries of natural languages (and, maybe, also to someone's attitude or taste). By this procedure, additional metaphysical and parallel formulas for particular operand entities are obtained and can be added to the integral informational system of Section 31.

Certainly, later on, in the formula development cycle, further inadequacies can be observed. This state of inappropriateness can call for supplementary modification, correction, and extension of the existing formula system with the goal to reach the adequate state of informational excellence.

This is the main strategy of a text understanding and its perfection through an informational formula system. Improvements in this sense can always be added, broadening the realm of informing of information which concerns a verbal text.

At the end of the paragraph formula system integration into the section system, the following must be stressed repeatedly. In depicting a text into a system of informational formulas, words and word groups are transformed into informational operands and operators. However, operands and operators are not words and word groups anymore; they perform (inform) as informing entities and not as statically, within a given dictionary determined items. Operands are entities which develop in an informational way. This kind of informing is informational arising not only in the way of meaning of entities, but also in their formal structure and organization, which is interaction, interweaving, impacting, observation of the appearing entities of the system.

Thus, formalizing a text means giving to the text words and their grammatically organized groups an informationally arising character, which spontaneously and circularly, also intentionally, projects the informational system, structuring and organizing it dynamically. Through formalization, a text becomes informationally dynamic and keeps the nature of the original text within a certain intention of informational arising, however, flexibly letting the spontaneity and circularity to *be* and perform in an investigational and developmental manner.

5. AN INFORMATIONAL FORMULA SYSTEM IMAGING A WRITTEN TEXT

5.1. *THE IMPACT OF OPERANDS APPEARING SEVERAL TIMES IN INFORMATIONAL FORMULAS*

In a formula or formula system, an informational operand or an subscripted operator marker can appear several times, on different places, in a single or in different formulas, and in different forms. Such operands or markers can be free standing entities or they can be formally nested in complex formal structures, for instance, as operands of the form

(V.1) $\alpha, \varphi(\alpha), \varphi(\psi(\alpha)), \ldots$

or as subscripted operators, for instance,

(V.2) $\models_{\alpha}, \models_{\varphi(\alpha)}, \models_{\varphi(\psi(\alpha))}, \ldots$

The basic question is, how do the informational connections between entities, possessing α, perform when α appears in operand and operator entities, several times in a single formula or in different formulas of a formula system?

In a system of mathematical formulas, the connection between different appearances of a variable α follows the metaphysical scheme $\alpha = \alpha$, which in such cases is the consequence of the mathematical axiom of equivalence of the appearing variable. Mathematics postulates the identity of a variable α in the form of equality (equality symbol =) between several appearances of one and the same symbol α, that is, $\alpha \models \alpha$. How does the informational theory understand this question in case of operand appearances in an informational formula system?

Every informational formula is an autonomous informational entity in which formula's constitutive entities, that is, operands and operators, are involved. Informational formulas arise, that is, develop (grow, change, reduce, vanish) in their own way and by the connective (observing, operand linked, etc.) impact of other formulas. And this happens also to operands and operators which occur in formulas of a formula system. Informationally linked entities can be always metaphysically decomposed, enriching a system of formulas in a new, linking way.

Let in a formula system some »equally« marked entities appear, for instance, α, $\varphi(\alpha)$, $\varphi(\psi(\alpha))$, ..., $\models_{\alpha}$, $\models_{\varphi(\alpha)}$, $\models_{\varphi(\psi(\alpha))}$, ... Metaphysical extension of the basic linking between these entities can start from the following formulas:

(V.3) $\alpha \models \alpha;\ = \models \varphi(\alpha);\ \alpha \models \varphi(\psi(\alpha));\ \ldots$
$\alpha \dashv \alpha;\ \alpha \dashv \varphi(\alpha);\ \alpha \dashv \varphi(\psi(\alpha)),\ \ldots$
$\alpha \models (\beta \models_{\alpha} \gamma);\ \alpha \models (\beta \models_{\varphi(\alpha)} \gamma;$
$\alpha \models (\beta \models_{\varphi(\psi(\alpha))} \gamma;\ \ldots$
$\alpha \dashv (\beta \models_{\alpha} \gamma);\ \alpha \dashv (\beta \models_{\varphi(\alpha)} \gamma;$
$\alpha \dashv (\beta \models_{\varphi(\psi(\alpha))} \gamma;\ \ldots$

A further expansion of this principle is that the entire formulas, and not only equally marked operands and operators and their transformations, inform mutually each other, that is, are informationally involved.

Let us analyze the particular case of formulas $\alpha \models \beta$ and $\gamma \models \alpha$, where entity α appears in both formulas. The extended principle of informational involvement would deliver, for instance,

(V.4) $(\alpha \models \beta) \models (\gamma \models \alpha)$;
$(\gamma \models \alpha) \models (\alpha \models \beta)$

Now, let us mark the occurrences of α in formulas

(V.5) $\alpha \models \beta; \gamma \models \alpha$ by
$\alpha_1 \models \beta; \gamma \models \alpha_2$

where a formal distinction between α_1 and α_2 is introduced. Thus, the metaphysical principle $\alpha \models \alpha$ and $\alpha \dashv\models \alpha$ for both formulas is not violated. However, we can observe and decompose the both metaphysical situations, that is,

(V.6) $\alpha_1 \models \alpha_2; \alpha_1 \dashv\models \alpha_2;$
$\alpha_1 \subset (\alpha \models \beta); \alpha_2 \subset (\gamma \models \alpha)$

But, because $\alpha \models \beta$ and $\gamma \models \alpha$ can be understood as holistic entities, the question of α's interaction in both formulas can be transposed to the whole formulas in which α appears. This conclusion holds also for single formulas in which α as an operand or a marker appears several times. On the basis of such reflection a set of dedicated implications can be constructed pertaining to the problem of communication (informational cooperation) between equally marked entities, that is, to α, $\varphi(\alpha)$, $\varphi(\psi(\alpha))$, ..., $\models_\alpha$, $\models_{\varphi(\alpha)}$, $\models_{\varphi(\psi(\alpha))}$, ... occurring in different formulas, at different places of a formula system. Thus, for instance,

(V.7) $(\alpha \models \beta; \gamma \models \alpha) \Rightarrow$
$((\alpha \models \alpha; \alpha \dashv\models \alpha;$
$(\alpha \models \beta) \models (\gamma \models \alpha);$
$(\alpha \models \beta) \dashv\models (\gamma \models \beta))$

etc. In this way, formulas are informationally connected, interweaved, mutually dependent, communicating (informing, cooperating) through equally marked operand entities.

5.2. *A FORMULA SYSTEM DEVELOPING ON THE BASIS OF THE WRITTEN TEXT IMAGING*

In this section we have to give short answers to the following questions: What does the formalization of a written text mean and what is the purpose of such formalization? How can the initial formalization of a text develop afterwards and what can this development offer as an arising informational system? How do these questions touch the problem of the so-called informational machine which is a tool for informing of information?

What does the formalization of a written text mean and what is the purpose of such formalization? Formalization is nothing else than a transcription of the literal text from a natural into the informational language. This language constitute operands, operators, and pairs of parentheses. The language presupposes that everything informational can be expressed in a mixed active and passive way, by informing, within which operand and operator entities act, interact, impact each other according to the basic informational principles. In this way, a text formalization is always possible. Even more: the process of formalization possesses its own analysis and synthesis, that is, informational composition and decomposition, and it is possible to transcribe a text sufficiently precisely, supplementary, complementary, and also additionally. The result of a text formalization can be a perfected, also improved, informationally well-determined formula system, which from now up performs as an informational program (IM). An IM by itself is informational and behaves as an informational operand. Thus, the purpose of formalization is to make a text informing as to the text corresponding formula system. This system develops, arises with the intention towards a more perfect or final state.

How can the initial formalization of text develop afterwards and what can this development offer as an arising informational system? The initial formula system is as precise as possible informational interpretation of the written text. It is not always sufficiently exact »copy« of the original text, so, supplementary formulas, additional inter-

pretation, and extension of the initial system are possible. One can experience substantial differences in trials to formalize, for instance, the English and the German version of one and the same text. The well-known problem of interpretation occurs in every attempt to translate a text into a natural or formal language. In the next development cycle, the initial or any later formula system can be corrected, made more precise or satisfactory for someone's taste or for an informational machine understanding. By this process of »improvement«, the formula system arises intentionally, where intentional spontaneity and circularity come to one's or a system's constructive and de-constructive consciousness.

How do the previous questions touch the problem of the so-called informational machine, which is a tool for informing of informational entities? Within an IM, operands and operators must keep the property to inform and to be informed, that is, counter- inform and embed the arriving and arisen information in a spontaneous and circular way. Within its architecture and operating system, the IM must offer possibilities and necessities of spontaneous and circular informing of informational entities and enable the informational interaction among entities. In this context, the spontaneous means an automatic, impulsive, unpremediated, arising action which is simultaneously unconstrained, unreasoned, natural, etc. in an entity-intentional and phenomenal way, keeping the sense of an entity's phenomenalism, externalism, internalism, and metaphysics. The circular has the meaning of a circulating way of informing not only in an indirect and cyclic way, but al;so in a spread, exchange, mutually impacting, rotating manner, etc. An IM is programmed by informational formula systems which are nothing other than informational programs, describing the involved informational entities informationally.

6. CONCLUSION

We see how a linguistic, for instance, a traditional grammatical and semantical approach to the analysis (and synthesis) of sentences, paragraphs, and texts is an artificially reduced, atomically broken, grammatically structured, etc., and by this not in the possible entirety informing system. What could be said to this observation is that informational entities of a written text, that is words, idioms, word groups, sentences, paragraphs, etc. inform among each other and are informed in such a way. Informational approach of text interpretation surpasses the conventional styles of linguistic understanding, which in its atomic structure is not a dynamically structured and arising system of understanding. Certainly, the informational approach can consider traditional and scientifically posited structures and methods of a text recognition; however, this may be regularly not sufficient for the dynamically understood written text within an informational environment, in the world where information and its understanding arise in every moment.

REFERENCES

[BIW] Dreyfus, H.L., Being-in-the-World, The MIT Press, Cambridge, Ma (1991).

[SZ] Heidegger, M., Sein und Zeit, Max Niemeyer Verlag, Tübingen (1986).

[BT] Heidegger, M., Being and Time, Harper & Row, New York (1962).

[BV] Heidegger, M., Bitak i vrijeme (in Croatian), Naprijed, Zagreb (1985).

[WIT] Heidegger, M., What Is a Thing, Regnery/Gateway Inc., South Bend, In (1967).

[OK] Popper, K., Objective Knowledge, An Evolutionary Approach, Clarendon Press, Oxford (1972).

[OWI] Železnikar, A.P., On the Way to Information, Informatica **11** (1987) 1, 4-18.

[UAI2] Železnikar, A.P., Understanding as Information II, Informatica **14** (1990) 4, 5-30.

[FIP] Železnikar, A.P., Formal Informational Principles, Cybernetica **35** (1992) (in press).

Appendix: *Index of Operands and Operators*

pertaining to Being-there in Heidegger's Being and Time (§31)

A DICTIONARY OF INFORMATIONAL OPERANDS

(Operand symbols with English, German, and Slovene explanation)

Greek Operands

α something; thing | etwas; Ding, das | nekaj; stvar | (3.1), (3.2), (3.4) (3.8), (5.1), (7.3), (7.4), (7.9), (8.1), (8.9),(8.10), (9.3), (12.5), (14.3), (14.7), (16.5) □

α_{act} actuality | Wirklichkeit, die | aktualnost | (3.10) □

$\alpha_{adequate}$ adequateness; adequate | hinreichend | zadostnost; zadosten | (18.1) □

$\alpha_{adequate}$ adequate way | hinreichend in den Blick | zadosten v pogledu | (18.1) □

α_{at_hand} something-at-hand | Vorhandene, das| nekaj-priročnega | (9.1) □

α_{merely_poss} something merely possible | nur Mögliche, das | edino mogoče | (3.9) □

α_{poss_ways} different possible ways and degrees | verschiedene mögliche Weişen und Grade, die | različni možni načini in stopnje | (4.6) □

$\alpha_{present\text{-}at\text{-}hand}$ something present-at-hand | ein Vorhandenes | nekaj priročnega | (3.4), (3.7), (14.3) □

α_{still_out} something still outstanding | Noch-nicht-vorhandenes, das | še-ne-razpoložljivo | (5.1) □

β_{begin} beginning, the | Anfang, der| začetek | (14.6) □

$\beta_{begin}(\tau_{trad}(\pi_{philo}))$ beginning onwards the tradition of philosophy, the | Tradition der Philosophie von Anfang an, die | tradicija filozofije od začetka | (14.6) □

β_{body} body | Leib, der | telo | (14.3) □

$\beta_{breakdown}$ breakdown | Weise, die; Bruch, der | način; prelom | (17.7) □

Υ_{char} character | Mäßige, das; '-mäßig'; Charakter, Entwurfcharakter, der | karakter; primerno | (7.9), (8.4), (8.9), (8.10), (9.4), (12.1), (17.7) □

$\Upsilon_{char}(\varepsilon_{enigma}(\mathfrak{B}(\mathfrak{D})))$ full enigmatical character of Dasein's Being, the | volle Rätselhaftigkeit des Seins des Daseins, die | polna skrivnostnost biti tubiti | (17.7) □

$\Upsilon_{char}(\pi_{project}(\mathfrak{U}))$ understanding's projective character, the | Entwurfcharakter des Verstehens, der | projektivni karakter razumevanja | (12.1) □

$\Upsilon_{char}(\mathfrak{D})$ character of Dasein | Mäßige des Daseins, das | tubiti primerno | (7.9) □

$\Upsilon_{char}(\mathfrak{U})$ character of understanding | Entwurfcharakter des Verstehens, der | karakter razumevanja; razumevajoče | (8.4), (8.9), (8.10), (9.4) □

$\Upsilon_{circumsp}$ circumspection | Umsicht, die| preudarnost | (12.2), (15.1) □

$\Upsilon_{circumsp}(\Upsilon_{concern})$ circumspection of concern | Umsicht des Besorgens, die | preudarnost preskrbe | (12.2), (15.1) □

$\Upsilon_{cleared}$ claredness | Gelichtetheit, die | osvetljenost | (14.2) □

Υ_{comm} common | gemeinsam | skupen | (15.1) □

$\Upsilon_{comm}(\sigma_{sense})$ common sense | Verständigkeit, die | razumnost | (15.1) □

$\Upsilon_{concept\_ex}$ explicit conceptions | expliziten Begriffe, die | eksplicitni pojmi | (15.4) □

$\Upsilon_{concept\_ex}(\mathfrak{B})$ explicit conceptions of Being |

expliziten Begriffe von Sein, die | eksplicitni pojmi biti | (15.4) □

$\Upsilon_{concern}$ concern | Besorgen, das | preskrba | (12.2), (15.1) □

$\Upsilon_{concrete}$ concreteness; concretely | konkret | konkreten | (18.1) □

$\Upsilon_{concrete}(\tau_{this}(\mu_{multi}(\varepsilon_{exist}))$ these existentialia concretely | konkrete diese Existenzialien, die | ti konkretni eksistenciali | (18.1) □

Υ_{cond} condition | Bedingung, die | pogoj | (7.7), (7.9) □

$\Upsilon_{cond}(\pi(\alpha))$ conditions of the possibility of entities | Bedingungen der Möglichkeit des Seiendes, die | pogoji možnosti bivajočega | (7.9) □

$\Upsilon_{cond}(\pi(\mathfrak{B}(\mathfrak{N})))$ conditions of possibilities of the Being of Nature | Bedingungen der Möglichkeit nach der Frage des Seins der Natur, die | pogoji možnosti biti narave | (7.7) □

$\Upsilon_{consider}$ considerateness | Rücksicht, die | ozir | (12.2) □

$\Upsilon_{consider}(\pi_{solicitude})$ considerateness of solicitude, the | Rücksicht der Fürsorge, die | ozir skrbi | (12.2) □

Υ_{const} Constitution; constitution| Konstitution, die | konstitucija | (9.4), (17.5) □

$\Upsilon_{const}(\mathfrak{B}(\tau_{there}))$ Constitution of the Being of the "there" | Konstitution des Seins des Da, die| konstitutcija Tu-biti | (9.4), (12.4), (17.5) □

Υ_{cont} contingency | Kontingenz, die | kontingenca; naključnost | (3.7) □

$\Upsilon_{cont}(\alpha_{present\text{-}at\text{-}hand})$ contingency of something present-at-hand | Kontingenz eines Vorhandenen, die | naključnost priročnega | (3.7) □

$\Upsilon_{content}$ contents, the | Bestand, der | vsebina; stanje | (8.10), (9.1) □

$\Upsilon_{content}(\mu_{mind})$ contents of mind; contents one has in mind | gemeinte Bestand, der | mišljena vsebina; vsebina uma | (8.10) □

$\Upsilon_{content}(\mathfrak{B}(\mathfrak{D}))$ contents of Dasein's Being, the | Seinsbestand des Daseins, der | stanje biti tubiti | (9.1) □

$\Upsilon_{genuine}$ genuine | durchsetzt | uveljavljen; resničen | (10.7), (17.7) □

$\Upsilon_{genuine}(\beta_{breakdown})$ genuine breakdown | echte Weise, die | pravi način | (17.7) □

Υ_{modal} category, modal | modale Kategorie, die | modalna kategorija | (3.8) □

$\Upsilon_{modal}(\pi_{at\text{-}hand})$ category, modal of presence-at-hand | modale Kategorie der Vorhandenheit, die | modalna kategorija priročnosti | (3.8) □

δ_{exist} derivative, existential | existenziale Derivat, das | eksistencialni derivat | (1.6) □

ε_{dim} essential dimension | wesenhaften Dimensionen, die | bistvene dimenzije | (8.1) □

$\varepsilon_{dim}(\alpha)$ essential dimension of that which (of something which) | wesenhaften Dimensionen des/der ... | bistvene dimenzije nečésa | (8.1) □

$\varepsilon_{dim}(\alpha \subset_{discl} \mathfrak{U})$ essential dimension of that which can be disclosed in understanding | die wesenhaften Dimensionen des im Verstehen Erschließbaren | bistvene dimenzije tega, kar se v razumevanju razkriva | (8.1) □

ε_{early} earlier | früher | prej | (16.6) □

ε_{ego} egocentric | egozentrisch | egocentričen | (13.1) □

$\varepsilon_{ego}(\mathfrak{D} \models_{decept} \mathfrak{D})$ egocentric self-deceptions of Dasein | egozentrischen Selbsttäuschungen des Daseins, die | egocentrične samoprevare tubiti | (13.1) □

ε_{enigma} enigmatical | rätselhaft | skrivnosten | (17.5), (17.7) □

$\varepsilon_{enigma}(\mathfrak{B}(\mathfrak{D}))$ enigmatical Being of Dasein | Rätselhaftigkeit des Seins von Dasein, die | skrivnostnost biti tubiti | (17.7) □

$\varepsilon_{essence}$ essences | Wesen, das | bistvo |

(15.3) □

$\varepsilon_{everyday}$ everyday | alltäglich | vsakdanji | (18.1) □

$\varepsilon_{everyday(\mathfrak{R}(\mathfrak{B}))}$
everyday kind of Being | alltägliche Seinsart, die | vsakdanji način biti | (18.1) □

ε_{ex} existence | Existenz, die | eksistenca | (10.11), (12.3) □

$\varepsilon_{exhibit}$ exhibited | Aufweisung, die | izkazanost | (16.6) □

$\varepsilon_{exhibit(\mathbb{C}_{const}(\mathfrak{B}))}$
exhibited in terms of the Constitution of the being | Aufweisung aus der Konstitution des Seins, die | izkazanost konstitucije biti | (16.6) □

ε_{exist} existentiale | Existenzial, das | eksistencial | (3.2), (3.11), (4.1), (9.1), (17.1), (17.5), (18.1) □

$\varepsilon_{exist(\Upsilon\ const(\mathfrak{B}(\tau_{there})))}$
existential constitution of the Being of the "there" | existenziale Verfassung des Seins des Da, die | eksistencialno stanje biti-tu | (17.5) □

$\varepsilon_{exist(\pi_{project})}$
existentiale of projection | Existenzial des Entwurfs, das |eksistencial projekta | (9.1) □

$\varepsilon_{express}$ expression | Ausdruch, der | izraz | (14.1), 14.2) □

$\varepsilon_{express(\sigma_{sight})}$
expression 'sight', the | Ausdruck »Sicht«, der | izraz »vid« | (14.1), (14.2) □

ε_{eyes} eyes | Augen, die | oči | (14.3) □

$\varepsilon_{eyes(\beta_{body})}$
bodily eyes, the | leibliche Augen, die | telesne oči | (14.3) □

ε_{fund} existentiale, fundamental | fundamentales Existenzial | fundamentalni eksistencial | (1.5) □

$\vartheta_{derive\_remote}$
derivatives, remote | entfernte Derivate, die | oddaljeni derivati | (15.2) □

$\vartheta_{derive\_remote}$
remote derivatives of understanding | entfernte Derivate des Verstehens, die | oddaljeni derivati razumevanja | (15.2) □

ϑ_{discl} or $\vartheta_{disclose}$
disclosedness | Erschlossenheit, die | razprtost | (8.4), (10.1), (10.2), (10.10), (12.2), (12.4), (14.2), (14.5), (16.1), (16.2), (17.1), (17.3), (18.1) □

$\vartheta_{discl(\sigma_{of\text{-}Being}(\Upsilon_{char}(\mathfrak{U}))\models_{exist\_const})}$
disclosednes of existentially constitutive state-of-Being of the character of understanding | Erschlossenheit des konstitutiven Seinkönnens hinsichtlich des Entwurfcharakters des Verstehens, die | razprtje eksistencialno konstitutivne moči-biti glede na projektni karakter razumevanja | (8.4) □

$\vartheta_{disclose(\pi)(\mathfrak{D}))}$
disclosure of possibilities of Dasein | Erschließen von Möglichkeiten des Daseins, das | razprtje možnosti tubiti | (17.3) □

$\vartheta_{disclose(\tau_{there})}$
disclosedness of the "there", the | Erschlossenheit des Da, die | razprtje Tuja | (12.2), (14.2), (16.1), (18.1) □

$\vartheta_{disclose(\mathfrak{B})}$
disclosedness of Being | Erschlossenheit von Sein, die | razprtost biti | (16.2) □

$\vartheta_{disclose(\mathfrak{B}_{in\text{-}the\text{-}world})}$
disclosedness of Being-in-the-world | Erschlossenheit des In-der-Welt-seins, die | razprtost biti-v-svetu | (10.1), (12.4), (17.1) □

$\vartheta_{disclose(\mathfrak{D})}$
Dasein's disclosedness | Erschlossenheit des Daseins, die | razprtost tubiti | (10.10) □

$\vartheta_{disclose(\mathfrak{W}_{world})}$
disclosedness of the world, the | Erschlossenheit der Welt, die | razprtost sveta | (10.2) □

$\vartheta_{diversion}$ diversion | Sichverlegen, das| samoodlaganje | (10.10) □

$\vartheta_{diversion(\mathfrak{U})}$
diversion of the understanding | Sichverlegen des Verstehens, das | samoodlaganje razumevanja | (10.10) □

ϑ_{domain} domain | innerhalb (des Bezirkes) | znotraj (območja) | (14.5) □

$\vartheta_{domain}(\vartheta_{disclose}(\sigma_{sense}))$ domain of discovery of sense | innerhalb des Enteckungsbezirkes des Sinns | znotraj odkritega območja smisla | (14.5) □

ι_{in} in-| Un-, das | ne- | (10.5) □

ι_{inauth} inauthentic | von Selbst abgeschnürt | neavtentičen | (10.5) □

ι_{indeed} indeed, the | Tat, die | dejanje | (17.6) □

ι_{indiff} indifference | Gleichgültigkeit, die | indiferenca | (4.1) □

$\iota_{intuition}$ intuition | Anschauen, das; Anschauung, die | gledanje; zrenje | (15.1), (15.2), (15.3) □

$\iota_{intuition}(\varepsilon_{essence})$ intuition of essences | »Wesensschau«, die | videz bistva | (15.3) □

$\iota_{inventory}$ inventory | Vorhandenes, das | razpoložljivo | (9.1) □

$\iota_{inventory}(\mathfrak{D})$ inventory of Dasein | Vorhandene des Daseins, das | razpoložljivo tubiti | (9.1) □

$\iota_{investigate}$ investigation | Untersuchung, die | raziskava | (16.7) □

$\iota_{involve}$ involvement | Bewandtnis, die | zapletenost | (7.5) □

ι_{item} item(s) | Moment(e) | del(i) | (12.4), (12.5) □

$\iota_{item}(\Upsilon_{const})$ constitutive items, the | Verfassungsmomente, die | konsitutivni deli | (12.4) □

λ_{lack} lack | Un-; Mangel, der | ne-; pomanjkanje | (13.1) □

$\lambda_{lack}(\mathfrak{A}_{acquaint} \models_{with} \mathfrak{W}_{world})$ lack of acquaintance with the world | Unkenntnis der Welt, die | neznanje sveta | (13.1) □

λ_{leeway} leeway | Spielraum, der | igralni prostor; diskurz | (8.4) □

$\lambda_{leeway}(\Upsilon_{char}(\mathfrak{U}))$ leeway of the character of understanding | Spielraum des Entwurfcharakters des Verstehens, der | diskurz projektenga karakterja razumevanja | (8.4) □

λ_{liber} liberty | Willkür, die | prostost | (4.1) □

$\lambda_{liber}(\iota_{indiff})$ liberty of indifference | Gleichgültigkeit der Willkür, die | prostost brezpomembnosti; brezpomembnost prostosti | (4.1) □

λ_{limit} limits | Grenzen, die | meje | (16.7) □

$\lambda_{limit}(\iota_{investigate})$ limits of investigation | Grenzen der Untersuchung, die | meje raziskave | (16.7) □

λ_{logic} logical | logisch | logično | (15.4) □

$\lambda_{logic}(\varphi_{phenomenal})$ phenomenological | phänomenologisch | fenomenološko | (15.4) □

λ_{lower} level, lower | niedriger | niže | (3.10) □

μ_{basic} mode, basic | Grundmodus, der | osnovni modus | (1.5) □

μ_{exist} meaning, existential | existenziale Sinn, der | eksistenčni smisel | (16.7) □

$\mu_{manifold}$ manifold | Mannigfaltige, das | raznovrstno | (7.6) □

$\mu_{manifold}(\pi_{at\text{-}hand})$ manifold present-at-hand, the | mannigfaltige Vorhandene, das | raznovrstno priročno | (7.6) □

μ_{mind} mind | Gemeinte, das | mišljeno, umsko | (8.10) □

$\mu_{mistake}$ mistake | Vergreifen, das | napačno prijetje | (4.3) □

μ_{modif} modification | Modifikation, die | modifikacija | (10.10) □

$\mu_{modif}(\pi_{project})$ modification of projection | Modifikation des Entwurfes, die | modifikacija zasnove | (10.10) □

μ_{more} 'more' | »mehr« | »več« | (9.1) □

$\mu_{more}(\mathfrak{D})$ 'more' of Dasein, the | »Mehr« des Daseins, das | »več« tubiti | (9.1) □

μ_{multi} some; multi- | einige | nekateri | (17.1), (18.1) □

$\mu_{multi}(\varepsilon_{exist})$ existentialia | Existenzialien, die | eksistenciali | (17.1), (18.1) □

$\mu_{multi}(\mathfrak{S}_{mind})$

states-of-mind | Befindlichkeiten, die | počutnosti | (17.1) □

μ_{sign} meaning; significance | Bedeutung, die | pomen | (3.1) □

ν necessity | Notwendigkeit, die | nujnost | (3.10) □

ν_{anew} anew | erneut | nanovo | (17.7) □

$\nu_{anew}(\mathfrak{Q}_{quest} \models_{about} (\mathfrak{B} \models_{of} \tau_{throw}(\pi_{project}(\mathfrak{B}_{in\text{-}the\text{-}world}))))$
anew the question about the Being of thrown projective Being-in-the-world | erneut die Frage nach dem Sein des geworfen-entwerfenden In-der-Welt-sein | nanovo vprašanje po biti vržene-projektirajoče biti-v-svetu | (17.7) □

$\nu_{at\_any\_time}$
the necessary at any time | jemals Notwendige, das | kdaj nujno | (3.8) □

ν_{view} view | Blick, der | pogled | (18.1) □

ξ_{sign} significance | Bedeutsamkeit, die | pomembnost | (2.5), (2.6), (2.7), (2.8), (8.3) □

ξ_{sign_poss} significance, possible
mögliche Bedeutsamkeit, die | mogoča pomembnost | (7.3) □

o_{one} one | wir | mi | (9.1), (10.4), (10.9), (14.1), (14.4), (14.7), (15.1), (15.4), (17.5), (17.7), (18.1) □

$o_{ontology}$ ontology | ontologisch | ontološki | (15.1) □

π possibility; potentiality | Möglichkeit, die; Können, das | možnost; potencialnost | (3.8), (3.9), (3.10), (3.11), (3.12), (3.13), (4.1), (4.3), (4.5), (5.7), (7.3), (7.6), (7.7), (7.9), (8.1), (8.8), (8.9), (8.10), (8.11), (10.1), (10.8), (11.1), (16.3), (17.2), (17.3) □

$\pi(\alpha)$ possibilities of an entity | Möglichkeiten des Seindes, die | možnosti bivajočega | (7.3), (7.9) □

$\pi(\gamma_{char}(\mathfrak{U}))$
possibility of the character of understanding | Möglichkeit des Entwurfcharakters des Verstehens, die | možnost karakterja razumevanja | (8.10) □

$\pi(\upsilon_{unity})$ possibility of unity | Möglichkeit der Einheit, die | možnost enosti | (7.6) □

$\pi(\mathfrak{B}(\mathfrak{D}))$ possibilities of Dasein's Being | Möglichkeiten des Daseins Sein, die | možnosti biti tubiti | (4.3) □

$\pi(\mathfrak{B}(\mathfrak{N}))$ possibility of the Being of Nature | Möglichkeit des Seins von Natur, die | možnost biti narave | (7.7) □

$\pi(\mathfrak{B}_{free})$ possibility of Being-free, the | Möglichkeit des Freiseins, die | možnost prosto-biti | (4.5) □

$\pi(\mathfrak{D})$ Dasein's possibility | Möglichkeit des Daseins, die | možnost tubiti | (3.5), (3.6), (5.7), (8.8), (17.3) □

$\pi(\mathfrak{D} \models; \models \mathfrak{D})$
possibilities of Dasein as it is | Möglichkeiten des Daseins aus denen her es ist, die | možnosti biti, iz katerih bit je | (17.2) □

$\pi(\mathfrak{U})$ possibilities of understanding | Möglichkeit des Verstehens, die | možnost razumevanja | (8.11), (10.1), (11.1) □

$\pi(\mathfrak{D} \models_{first\_find} \not\models)$
possibility of first finding Dasein again, the | sich in Daseins Möglichkeiten erst wieder zu finden | se v možnostih tubiti šele ponovno najti | (5.7) □

$\pi_{at\text{-}hand}$ presence-at-hand | Vorhandenheit, die | priročnost | (3.8), (5.1), (7.6), (14.3), (15.1) □

$\pi_{at\text{-}hand}(\alpha)$ presence-at-hand of something | Vorhandenheit von etwas, die | priročnost nečesa | (14.3) □

π_{basic_1} or π_{basic_2}
basic possibilities | Grundmöglichkeiten, die | temeljne možnosti | (10.9) □

$\pi_{basic\_1}(\mathfrak{U})$ or $\pi_{basic\_2}(\mathfrak{U})$
basic possibilities of understanding | Grundmöglichkeiten des Verstehens, die | temeljne možnosti razumevanja | (10.9) □

π_{def} possibility, definite | bestimmte Möglichkeit | določena možnost | (4.2) □

π_{for} potentiality-for- | (Sein-)können-, das | možnost-za- | (7.2) □

$\pi_{for(\mathfrak{B}_{in\text{-}the\text{-}world})}$
potentiality-for-Being-in-the-world | Sein-können-in- der-Welt, das | bit-možnosti-v-svetu | (7.2) □

$\pi_{for\text{-}Being}$ potentiality-for-Being | Seinkönnen, das | moči-biti | (3.3), (3.6), (3.13), (4.1), (4.3), (4.5), (5.1), (5.3), (5.7), (6.1), (8.4), (9.2), (9.3), (10.1), (10.8), (11.1), (16.1), (17.4) □

$\pi_{for\text{-}Being(\mathfrak{D})}$
Dasein's potentiality-for-Being | Seinkönnen des Daseins, das | potencialnost biti tubiti; moči-biti tubiti | (4.5), (5.7), (6.1), (9.2), (9.3), (11.1), (16.1), (17.4) □

$\pi_{interconn}$ possible interconnection | Möglichkeit des Zusammenhangs, die | možnost povezave | (7.5) □

$\pi_{interconn(\mathfrak{R}_{to\text{-}hand})}$
possible interconnection of the ready-to-hand | Möglichkeit des Zusammenhangs von Zuhandenem, die | možnost povezave priročnega | (7.5) □

π_{log} possibility, logical | logische Möglichkeit, die | logična možnost | (3.7) □

$\pi_{peculiar}$ peculiar | eigen | svojski | (14.4) □

$\pi_{peculiar(\mathfrak{S}_{see})}$
peculiar of seeing, the | Eigene des Sehens, das | svojsko videnja | (14.4) □

π_{philo} philosophy | Philosophie, die | filozofija | (14.6) □

π_{plan} plan | Plan, der | načrt | (8.6) □

π_{prim} primordiality; primordial | gleichursprünglich | prvobiten | (12.5), (17.1) □

$\pi_{prim(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world}))}$
primordial disclosedness of Being-in-the-world | ursprüngliche Erschlossenheit des In-der-Welt-seins, die | izvirna razprtost biti-v-svetu | (17.1) □

$\pi_{prim(\iota_{item}) \models \mathfrak{S}_{exist}(\alpha))}$
primordiality in those items which are constitutive for existence of entities | gleichursprünglich der konstitutiven Momente der Existenz des Seiendes | prvobitno konstitutivnega v eksistenci bivajočega | (12.5) □

π_{prior} priority | Vorrang, der | prednost | (15.1) □

$\pi_{prior(\pi_{at\text{-}hand} \subset \tau_{trad(o_{ontology})})}$
priority of the present- at-hand in traditional ontology, the | traditionelle ontologische Vorrang des Vorhandenen, der | tradicionalna ontološka prednost priročnega | (15.1) □

$\pi_{prior(\pi_{pure(\iota_{intuition})})}$
priority of pure intuition | Vorrang des puren Anschauen, der | prednost čistega pogleda | (15.1) □

$\pi_{project}$ projection | Entwurf, der | osnutek; projekcija | (8.2), (8.4), (8.9), (8.10), (9.1), (9.4), (10.1), (10.10), (12.1), (16.4), (16.5), (17.3), (17.4), (17.5), (17.7) □

$\pi_{project(\vartheta_{disclose}(\pi(\mathfrak{D})))}$
projective disclosure of possibilities of Dasein, the | entwerfende Erschließen der Möglichkeiten von Dasein, das | projektirajoče razprtje možnosti tubiti | (17.3) □

$\pi_{project(\pi_{for\text{-}Being}(\mathfrak{D}))}$
projection of Dasein's potentiality-for-Being | Entwurf des Daseins Seinkönnens, der | projekt moči-biti tubiti | (17.4) □

$\pi_{project(\mathfrak{B}_{in\text{-}the\text{-}world})}$
projection of Being-in-the-world | Entwurf des In-der-Welt-seins, der | projekt biti-v-svetu | (16.5), (17.7) □

$\pi_{project(\mathfrak{U})}$ projection of understanding | Entwurf des Verstehens, der | zasnova razumevanja | (12.1) □

π_{pure} pure | pur | čist | (15.1) □

$\pi_{pure(\iota_{intuition})}$
pure intuition | pure Anschauen, das | čisti pogled | (15.1) □

$\pi_{pure(\iota_{intuition} \models_{of} \pi_{prior(\pi_{pure(\iota_{intuition})})})}$
pure intuition of its priority | pure Anschauen seines Vorranges, das | čisti pogled njegove prednosti | (15.1) □

$\pi_{solicitude}$ ways of solicitude; solicitude | Weisen des Besorgens, die | načini (možnosti) skrbi | (3.6),

(12.2) □

$\pi_{solicitude(\mathfrak{D})}$
ways of solicitude of Dasein | Daseins Weisen des Besorgens, die | načini skrbi tubiti | (3.6) □

π_{thrown} possibility, thrown | geworfene Möglichkeit, die | vržena možnost | (4.4) □

ρ_{range} range | Umkreis, der | območje | (10.1) □

σ_{exist} structure, existential | existenziale Struktur, die | eksistencialna struktura | (1.1), (5.1), (8.2) □

σ_{self} Self, the | Selbst, das | Se | (10.4), (10.5), (12.4) □

$\sigma_{self(o_{one})}$ one's own Self | sein Selbst, das | svoj Se | (10.4) □

$\sigma_{self(\mathfrak{D})}$ Dasein's Self | Selbst des Daseins, das| Se tubiti | (10.5) □

σ_{sense} sense | Sinn, der | smisel | (4.1), (12.4), (14.5), (15.1), (17.5) □

$\sigma_{sense(\lambda_{liber(\iota_{indiff})})}$
sense of the liberty of indifference, the | Sinn der Gleichgültigkeit der Willkür, der | smisel prostosti in diference | (4.1) □

$\sigma_{sense(\tau_{throw(\pi_{project})})}$
sense of thrown projection, the | im Sinne des geworfenen Entwurfs | v smislu vrženega projekta | (17.5) □

$\sigma_{sense(\mathfrak{U}_{well})}$
sense which is well understood | »Selbsterkenntnis«, die | samospoznanje | (12.4) □

σ_{sight} "sight"; sight | Sicht, die | vid | (12.1), (12.2), (12.3), (14.1), (14.4), (14.7), (15.1) □

$\sigma_{sight(\mathfrak{D})}$ Dasein's "sight" | Sicht des Daseins, die | vid tubiti | (12.2), (12.2), (12.3) □

$\sigma_{sight((\mathfrak{B} \models_{as} \mathfrak{B}) \models_{sake} (\mathfrak{D} \models; \models \mathfrak{D}))}$
sight which is directed upon Being as such, for the sake of which any Dasein is as it is | die Sicht auf das Sein als solches, umwillen dessen das Dasein je ist, wie es ist | vid biti kot take, zaradi katerega je tubit to, kar je | (12.2) □

σ_{sign} significance | Bedeutsamkeit, die | pomembnost | (16.2) □

$\sigma_{sign(\mathfrak{W}_{world})}$
significance of the world | Bedutsamkeit der Welt, die | pomembnost sveta | (16.2) □

σ_{sign_exist} existential signification | existenziale Bedeutung, die | eksistemcialni pomen | (14.4) □

$\sigma_{sign\_exist(\sigma_{sight})}$
existential signification to "sight" | existenziale Bedeutung der Sicht, die | eksistencialni pomen videnja | (14.4) □

σ_{so} so | So, das | Takó | (18.1) □

$\sigma_{solution}$ 'solution' | »Lösung«, die | »rešitev« | (17.7) □

$\sigma_{solution(\mathfrak{B})}$
Being's solution | Seins Lösung, die | bitna rešitev | (17.7) □

σ_{stage} stage | Stufe, die | stopnja | (16.6) □

$\sigma_{stage(\varepsilon_{early})}$
earlier stage | früher | prej | (16.6) □

$\sigma_{structure}$ structure | Struktur, die | struktura | (15.4) □

$\sigma_{structure(\mathfrak{B})}$
structure of Being | Seinsstruktur, die | struktura biti | (15.4) □

τ_{term_uni} universal term | universale Terminus, der | univerzalni têrmin | (14.7) □

τ_{there} there, the | Da, das | tu | (2.2), (2.3), (2.4), (5.4), (5.5), (9.4), (12.2), (14.2), (16.1), (16.6), (17.4), (17.5), (18.1) □

$\tau_{there(\mathfrak{D})}$ there of Dasein, the | Da des Daseins, das | tu tubiti | (2.2), (2.4), (5.5) □

τ_{this} these | diese | te | (18.1) □

$\tau_{this(\mu_{multi}(\varepsilon_{exist}))}$
these existentialia | diese Existenzialien | ti eksistenciali | (18.1) □

τ_{thrown} or τ_{throw}
throwness; thrown | Geworfenheit, die; geworfen | vrženost; vržen | (5.6), (8.5), (17.4), (17.7) □

$\tau_{throw(\pi_{project})}$
thrown projection | geworfene Entwurf, der | vrženi projekt | (17.5) □

$\tau_{throw(\pi_{project}(\mathfrak{B}_{in\text{-}the\text{-}world}))}$
thrown projective Being-in-the-world | geworfen-entwerfende

In-der-Welt-sein, das | vržena projektirajoča bit-v-svetu | (17.7) □

$\tau_{throw(\mathfrak{D})}$ Dasein's throwness | Geworfenheit des Daseins, die | vrženost tubiti | (17.4) □

τ_{total} totality | Ganzheit, die | celost | (7.5) □

$\tau_{total(\iota_{involve})}$ totality of involvements, the | Bewandtnisganzheit, die | celotna vpletenost | (7.5) □

τ_{trad} tradition; traditional | Tradition, die; traditionell | tradicija; tradicionalen | (14.6), (14.7), (15.1) □

$\tau_{trad(o_{ontology})}$ traditional ontology | traditionell ontologisch | tradicionalno ontološki | (15.1) □

$\tau_{trad(\pi_{philo})}$ tradition of philosophy | Tradition der Philosophie, die| filozofska tradicija | (14.6) □

$\tau_{transpar}$ "transparency" | Durchsichtigkeit, die | transparentnost; prozornost | (12.3), (12.4), (12.5) □

$\tau_{transpar(\alpha)}$ transparent to entities | durchsichtig gewordene Seiende, das | transparentno nastalo bivajoče | (12.5) □

υ_{unity} unity | Einheit, die | enost | (7.6) □

$\upsilon_{unity(\mu_{manifold(\pi_{at\text{-}hand})})}$ unity of the manifold present-at-hand, the | Einheit des mannigfaltigen Vorhandenen, die | enotnost raznolično priročnega | (7.6) □

$\upsilon_{unity(\mathfrak{N})}$ unity of Nature, the | Einheit der Natur, die | enotnost narave | (7.6) □

υ_{way} the most primordial and ultimate positive way | ursprünglichste und letzte positive ontologische Bestimmtheit, die | najizvornejša in poslednja pozitivna ontološka določenost | (3.11) □

φ_{basis} phenomenal basis | phänomenale Boden, der | fenomenalno ozadje | (3.13) □

φ_{fact} factical; facticity | faktisch; Faktizität, die | faktičen; fakticiteta | (8.4), (9.2), (17.4) □

$\varphi_{fact(\pi_{for\text{-}Being})}$ factical potentiality-for-Being, the | faktische Seinkönnen, das | faktična možnost biti | (8.4) □

$\varphi_{fact(\tau_{throw(\mathfrak{D})})}$ fact of Dasein's throwness | Faktum der Geworfenheit des Daseins, das | fakt vrženosti tubiti | (17.4) □

$\varphi_{fact(\mathfrak{D})}$ facticity of Dasein | Faktizität des Daseins, die | fakticiteta tubiti | (9.2) □

$\varphi_{feature}$ feature | Eigenschaft, die | značilnost | (14.4) □

$\varphi_{feature(\pi_{peculiar(\mathfrak{S}_{see})})}$ peculiar feature of seeing, the | Eigentümlichkeit des Sehens, die | posebnost videnja | (14.4) □

φ_{full} full; fullness | voll | poln | (10.1), (10.10), (12.4), (17.7), (18.1) □

$\varphi_{full(\gamma_{char(\varepsilon_{enigma(\mathfrak{B}(\mathfrak{D}))})})}$ full enigmatical character of Dasein's Being | volle Rätselhaftigkeit des Seins des Daseins, die | polna skrivnostnost biti tubiti | (17.7) □

$\varphi_{full(\vartheta_{disclose(\tau_{there})})}$ full disclosedness of the "there", the | volle Erschlossenheit des Da, die | polna razprtost Tuja | (18.1) □

$\varphi_{full(\vartheta_{disclose(\mathfrak{B}_{in\text{-}the\text{-}world})})}$ full disclosedness of Being-in-the-world, the | volle Erschlossenheit des In-der-Welt-seins, die | polna razprtost biti-v-svetu | (10.1), (12.4) □

$\varphi_{full(\vartheta_{disclose(\mathfrak{D})})}$ Dasein's full disclosedness | volle Erschlossenheit des Daseins, die | polna razprtost tubiti | (10.10) □

$\varphi_{phenomenal}$ phenomenological; phenomenal | phänomenologisch; phänomenal | fenomenološki; fenomenski | (15.3), (15.4), (18.1) □

$\varphi_{phenomenal(\alpha_{adequate(\mathfrak{W}_{world})})}$ phenomenally adequate way | phänomenal hinreichend in den Blick | fenomensko zadosten v pogledu |

(18.1) □

$\varphi_{phenomenal(\iota_{intuition}(\varepsilon_{essence}))}$ phenomenological intuition of essences | phänomenologische »Wesensschau« | fenomenološki pogled na bistvo | (15.3) □

φ_{sake} "for-the-sake-of-which" | Worum-willen, das | zaradi-česa | (2.4), (2.5), (2.6), (2.8), (8.3), (10.2), (16.2) □

$\varphi_{(x.y)}$ formula φ, marked by (x.y) | Formel φ bezeichnet mit (x.y), die | formula φ, označena z (x.y) | $\varphi_{(1.1)}, \cdots, \varphi_{(18.1)}$ (16.7); $\varphi_{(4.3)}$ (4.4); $\varphi_{(7.7)}$ (7.8), (7.9); $\varphi_{(8.1)}$ (8.2); $\varphi_{(8.9)}$ (8.10); $\varphi_{(14.4)}$ (10.2) □

ω other(s) | anderen, die | drug(i) | (3.6), (12.5) □

$\omega_{oneself}$ oneself | Sichverhalten, das | Se-vedénje | (8.6) □

Fraktur Operands

$\mathfrak{A}$ ability | Fähigkeit, die | sposobnost | (7.4) □

$\mathfrak{A}_{access}$ access | Zugangsart, die | dostop | (14.6), (14.7) □

$\mathfrak{A}_{acquaint}$ acquaintance | Kenntnis, die | znanje; védenje | (13.1) □

$\mathfrak{A}_{answer}$ answering | Antworten, das | odgovarjanje | (17.6) □

$\mathfrak{A}_{aware}$ awareness | Vernehmen, das | slišanje; zavest | (14.3) □

$\mathfrak{A}_{aware(\alpha)}$ awareness of something | Vernehmen eines (Vorhandenen), das | zavest o nečem | (14.3) □

$\mathfrak{A}_{detriment}$ detriment*ality* | Abträg*lichkeit*, die | odnes*ljivost;* škod*ljivost* | (7.4) □

$\mathfrak{A}_{detriment(\alpha)}$ detrimentality of something (of that which is) | Abträglichkeit des Seinden, die | odnesljivost nečesa | (7.4) □

$\mathfrak{A}_{service}$ service*ability* | Verwend*barkeit*, die | pripravn*ost* | (7.4) □

$\mathfrak{A}_{service(\alpha)}$ servicability of something | Verwendbarkeit von Seinden, die | pripravnost nečesa | (7.4) □

$\mathfrak{A}_{use}$ us*ability* | Dien*lichkeit*, die | primer*nost* | (7.4) □

$\mathfrak{A}_{use(\alpha)}$ usability of something | Dienlichkeit von etwas, die | primernost nečesa | (7.4) □

$\mathfrak{B}$ Being | Sein, das | bit | (3.2), (3.3), (4.3), (5.1), (5.4), (7.7), (7.9), (8.3), (8.5), (8.6), (8.11), (9.1), (9.4), (12.2), (14.6), (14.7), (15.4), (16.2), (16.3), (16.4), (16.5), (16.6), (16.7), (17.5), (17.6), (17.7), (18.1) □

$(\mathfrak{B} \models; \models \mathfrak{B})$ Being as informing | Sein qua Informieren, das | bit kot informiranje | (3.2) □

$\mathfrak{B}(\alpha)$ Being of entity | Sein des Seienden, das | bit bivajočega | (7.9), (16.5) □

$\mathfrak{B}(\vartheta_{disclose})$ Being of disclosedness | Sein der Erschlossenheit, das | bit razprtosti | (16.2) □

$\mathfrak{B}(\pi_{for\text{-}Being})$ Being of potentiality-for-Being | Sein des Seinkönnens, das | bit moči-biti | (5.1) □

$\mathfrak{B}(\tau_{there})$ Being of the »there« | Sein des Da, das | bit za »tu« | (5.4), (9.4), (17.5) □

$\mathfrak{B}_{\mathfrak{D}}$ or $\mathfrak{B}(\mathfrak{D})$ Being, Dasein's | Sein des Daseins, das | bit tubiti | (1.5), (4.3), (5.1), (8.3), (8.5), (8.6), (8.11), (9.1), (12.2), (17.5), (17.6), (17.7) □

$\mathfrak{B}(\mathfrak{N})$ Being of Nature, the | Sein der Natur, das | bit narave | (7.7) □

$\mathfrak{B}_{alongside}$ Being-alongside | Sein bei, das | pri-bit | (12.5) □

$\mathfrak{B}_{alongside(\mathfrak{W}_{world}, \mathfrak{B}_{with}(\omega))}$ Being-alongside the world and Being-with others | das Sein bei der Welt, im Mitsein mit Anderen | pri-bit sveta, v z-biti z drugimi | (12.5) □

$\mathfrak{B}_{exist}$ Being, existential | existenziale Sein, das | eksistencialna bit | (6.1) □

$\mathfrak{B}_{free}$ Being-free | Freisein, das | bit-prostosti | (4.5) □

(Will be continued)

SINHRONIZIRANA PODATKOVNA PRETOKOVNA RAČUNALNIŠKA ARHITEKTURA (I)

INFORMATICA 2/92

Keywords: parallel processing, dataflow computing, scheduling, allocator, performance evaluation parallel computer architecture.

Jurij Šilc
Laboratorij za računalniške arhitekture
Institut Jožef Stefan, Ljubljana

Delo obravnava problem časovne optimizacije asinhronega procesiranja na omejenem številu procesorjev. Predlagamo izvirno rešitev, ki temelji na uvedbi nekaterih mehanizmov sinhronizacije v asinhrono računanje. Graf pretoka podatkov, ki opisuje asinhrono procesiranje, opremimo s časovno optimalno sprožitveno funkcijo, ki služi tako pri vlaganju grafa v računalnik, kakor tudi pri njegovem časovno optimalnejšem izvrševanju. V ta namen smo razvili hevristična algoritma `pOptSinh` in `TOptSinh` za konstrukcijo optimalnih sprožitvenih funkcij, ki po pesimistični oceni vračata optimalno rešitev v 80% primerov. Nadalje predlagamo algoritma za dodeljevanje `MinGlDol` in `MinGlGor`, ki temeljita na optimizaciji medprocesorskih komunikacij. V primerjavi z znanimi razvrščevalnimi algoritmi dobimo s predlaganima algoritmoma boljše rezultate, kar potrjujejo analizirani primeri algoritmov za izračun hitre Fourierjeve transformacije, dinamične analize scene in LU razcepa matrike. Končno podajamo tudi zasnovo hibridne vzporedne arhitekture računalnika, ki podpira predlagano preoblikovanje asinhronega računanja.

SYNCHRONOUS DATAFLOW COMPUTER ARCHITECTURE - We discuss the problem of time optimization of asynchronous processing on a limited number of processors. We present an original solution to the problem based on introduction of synchronization mechanisms into asynchronous processing. The dataflow graph describing asynchronous processing is associated with the corresponding time-optimal firing function. This function is used both for loading a dataflow graph into the computer and for time-optimal graph execution. In order to do this, we have developed two heuristic algorithms, `pOptSinh` and `TOptSinh`, which are used for optimal firing function construction. According to conservative estimates, these algorithms return optimal functions with 80% probability. Furthermore, we propose two scheduling algorithms, `MinGlDol` and `MinGlGor`, which are based on interprocessor communication minimization. These two algorithms give better results compared to some other well known scheduling algorithms. This fact is illustrated in Fast Fourier Transformation, Dynamic Scene Analysis, and LU matrix decomposition algorithms. Finally, we present a design of hybrid parallel computer architecture capable of supporting modified asynchronous computing.

1. Uvod

Asinhrono (podatkovno vodeno) procesiranje [1, 2, 3, 4, 5, 6, 7] najlaže predstavimo z grafom pretoka podatkov [8], katerega točke ponazarjajo operacije oz. ukaze, usmerjene povezave pa so nosilke vrednosti operandov oz. podatkov. Ko v neko točko stečejo podatki po vseh njenih vhodnih povezavah, postane točka izvršljiva in prične takoj z izvrševanjem pridruženega ukaza. Pravimo, da se je točka sprožila. Ko točka konča izvrševanje ukaza, pošlje rezultat v vse svoje izhodne povezave. Graf pretoka podatkov je "strojni" jezik računalnikov, ki podpirajo podatkovno vodeno procesiranje in je rezultat prevajanja visokega programskega jezika z enkratno prireditvijo [9, 10]. Čeprav segajo osnovne ideje, na katerih temeljijo takšni računalniki, v pozna šestdeseta leta, so šele s pojavom VLSI tehnologije postale te ideje tudi uresničljive [11, 12]. V osemdesetih letih so se porodili v svetu številni projekti, katerih cilj je realizacija podatkovno pretokovnega računalnika [13, 14, 15, 16, 17, 18].

Intuitivno si lahko zamislimo implementacijo asinhronega procesiranja tako, da je vsaka točka podprta s svojim procesorjem in vsaka povezava s komunikacijskim kanalom. Izkaže se, da ni nujno vsaki točki prirediti lasten procesor že vnaprej, saj je v danem trenutku izvršljivih le nekaj točk. Takšen pristop se je uveljavil v krožni arhitekturi, kjer se dodeli točki procesor šele tedaj, ko

postane slednja izvršljiva. Ker se v veliki večini algoritmov vsebovana vzporednost dinamično spreminja, ni smiselno uporabiti toliko procesorjev, kot jih zahteva maksimalna vsebovana vzporednost, saj bi bila njihova izkoriščenost nizka. Po drugi strani pa lahko povzroči pomanjkanje procesorjev občutno upočasnitev procesiranja, ker prihaja do ti. nasičenj, ko izvršljive točke čakajo v vrsti na sprostitev procesorjev. Zaradi nedeterminističnega vstopanja so točke v vrsti neurejene glede na svojo pomembnost: npr. kritična točka (tj. točka, ki bi se morala takoj izvršiti) ne more prehiteti ostalih točk, ki se nahajajo v vrsti pred njo. Nedeterministično vstopanje v vrsto v splošnem podaljša čas izvrševanja grafa pretoka podatkov.

Predpostavimo, da je na voljo podatkovno pretokovni računalnik s potencialno neskončnim številom procesorjev ter izberimo poljuben graf pretoka podatkov. Denimo, da je za najhitrejšo asinhrono izvršitev izbranega grafa potrebnih vsaj m procesorjev. Tedaj označimo s T_m najkrajši čas, v katerem se ta graf izvrši na m procesorjih. Na računalniku z $n < m$ procesorjev pa bi se v splošnem isti graf asinhrono izvršil v času $T_n = T_m + \Delta T_n$, kjer $\Delta T_n \geq 0$. Za nalogo si zadajmo minimizirati podaljšek izvrševanja ΔT_n, tj. časovno optimizirati asihrono procesiranje pri n danih procesorjih.

2. Vzporedno procesiranje in arhitekture

Cilj vzporednega procesiranja in pripadajočih računalniških arhitektur [19, 20] je povečati "moč" računalnikov, tj. povečati njihovo zmogljivost in hitrost procesiranja ter odpraviti ozka grla v procesu računanja, ki so posledica *von Neumannove* arhitekture. Zaporedni računalniki so neučinkoviti pri reševanju nalog, ki se porajajo na področjih umetne inteligence (razpoznavanje slik in govora, ekspertni sistemi, avtomatsko dokazovanje itd.) in pri zahtevnih numeričnih izračunih (simulacija, modeliranje, grafika itd.). Zato je v zadnjih letih poseben poudarek na raziskavah novih načinov vzporednega procesiranja in programiranja [4, 21, 22] ter na vzporednih računalniških arhitekturah [19, 23, 24, 25, 26].

Eden obetavnejših načinov vzporednega procesiranja izvira iz modela *podatkovno vodenega računanja* [6], ki ga podpirajo pripadajoče *podatkovno pretokovne* arhitekture [3, 27, 28]. Ker bo podatkovno vodeno računanje predmet nadaljnje obravnave, določimo najprej mesto pripadajočih podatkovno pretokovnih arhitektur v množici računalniških arhitektur.

2.1. Razvrstitev arhitektur

Razvrščanja se lahko lotimo z zelo različnih zornih kotov. Tako predlaga *Feng* [29] razvrščanje na osnovi stopnje vzporednosti ter *Händler* [30] razvrščanje na podlagi stopnje vzporednosti in cevljenja[1]. Najpogosteje se uporablja *Flynnova* [31] razvrstitev računalniških arhitektur, ki temelji na ukaznih in podatkovnih tokovih . Arhitekture razvršča v štiri skupine: en ukazni in podatkovni tok (SISD), en ukazni in več podatkovnih tokov (SIMD), več ukaznih in en podatkovni tok (MISD) ter več ukaznih in podatkovnih tokov (MIMD). *Skillicorn* [32] je predlagal razširjeno Flynnovo razvrstitev. Na osnovi števila ukaznih in podatkovnih procesorjev, ukaznih in podatkovnih pomnilnikov ter načina njihovega povezovanja je razdelil računalniške arhitekture v 28 razredov. V razredih od 1 do 5 so podatkovno pretokovne in redukcijske arhitekture, v razredu 6 so zaporedne von Neumannove arhitekture, razredi od 7 do 10 obsegajo procesorska polja, razreda 11 in 12 sta MISD arhitekture in končno v razrede od 13 do 28 razvrsti večprocesorske arhitekture. *Dasgupta* [33] je odpravil nekatere pomankljivosti v Skillicornovi razvrstitvi in predlagal hierarhični sistem razvrstitve arhitektur. *Treleaven* [4] razvršča računalniške arhitekture glede na: organizacijo računanja, programsko organizacijo in organizacijo stroja. V slednji razvrstitvi je mesto podatkovno pretokovnih arhitektur zelo jasno opredeljeno, zato bomo v nadaljevanju to razvrstitev v grobem tudi predstavili.

2.1.1 Organizacija računanja

Faze računanja. Računanje je proces, ki ga sestavlja zaporedno ponavljanje treh faz, in sicer: *izbiranje*, *preverjanje* in *izvrševanje*.

Izbiranje: Določi se množica ukazov programa, ki so možni kandidati za izvrševanje. V splošnem izbira ukaza še ne pomeni njegove takojšnje

[1] Cevljenje iz angl. "pipelining"

izvršitve. Pravilo, po katerem se ukazi izbirajo, imenujemo *pravilo izbiranja*. Poznamo tri pravila izbiranja: *brezpogojno* (ukaz se izbere ne glede na njegov položaj v programu), *notranje* (izberejo se najglobje gnezdeni ukazi izraza) in *zunanje* (izbirajo se samo negnezdeni ukazi).

Preverjanje: Ugotavlja se izvršljivost ukaza, kar pomeni, da se preverja prisotnost vseh potrebnih vrednosti operandov v izbranih ukazih. Pravilo za preverjanje imenujemo *pravilo izvršitve*. Izvršljivi ukazi preidejo v tretjo fazo računanja. Če ukaz ni izvršljiv, se bodisi odloži ter se izbere kasneje ali pa se zahtevajo manjkajoče vrednosti operandov.

Izvrševanje: Izvršljivi ukazi se dejansko izvrše.

Vodenje računanja. Ločimo tri osnovne načine vodenja računanja: *krmilno* vodeno računaje, *z zahtevo* vodeno računanje in *podatkovno* vodeno računanje.

Krmilno vodenje: Izbiranju ukazov, ki je brezpogojno, sledi izvrševanje ukaza. Torej se pri krmilnem vodenju ne izvaja preverjanja ukazov.

Vodenje z zahtevo: Ukaz se izbere (navadno po pravilu zunanjega izbiranja) tedaj, ko prej izbrani ukaz zahteva njegov rezultat. Nato se ugotovi, če so izbrani ukazi izvršljivi, če so torej znane vrednosti njihovih operandov; če vrednost operanda še ni znana, se posreduje zahteva tistemu ukazu, ki ga more izračunati. Izvršijo se vsi tisti ukazi, katerih potrebni operandi so znani.

Podatkovno vodenje: Izbiranje poteka tako, da se vsakemu ukazu dodeli procesor. Preverjanje sestoji iz sprotnega ugotavljanja vrednosti vseh vhodnih operandov, sledi mu izvrševanje vseh preverjenih ukazov.

2.1.2 Programska organizacija

Krmilni mehanizem. Krmilni mehanizem določa vpliv izvršitve nekega ukaza na izvrševanje ostalih ukazov ter je bodisi *zaporedni* (ukazi se izbirajo drug za drugim in se po vrsti izvršujejo), *rekurzivni* (posreduje se zahtevo po operandih ter sproži izvrševanje tistega ukaza, katerega rezultat je bil zahtevan) ali *sočasni* (ugotavlja se prisotnost vhodnih operandov ter sproži izvrševanje tistih ukazov, ki imajo na voljo vse potrebne operande).

Podatkovni mehanizem. Podatkovni mehanizem določa način, po katerem si ukazi delijo skupne operande. Ukazi si lahko delijo skupne operande s pomočjo: *vstavljene vrednosti* (če je vrednost operanda znana že pred pričetkom računanja, se takoj vpiše v vse ukaze, ki jo potrebujejo), *sprotne vrednosti* (kopije operanda, ki se je izračunal, se posredujejo vsem ukazom, ki jim je ta operand skupen) in *reference* (vsi ukazi vsebujejo tudi referenco na skupen operand).

Modeli računanja. Vodenja računanja realiziramo z različnimi *modeli* računanja. V vsakem modelu se zrcali izvirni način vodenja računanja v obliki *krmilnega* ter *podatkovnega* mehanizma. Opredelitev različnih modelov računanja glede na podatkovni in krmilni mehanizem prikazuje slika 1.

Krmilno vodenje: Realizirati ga je mogoče z naslednjimi modeli: z zaporednim krmilnim tokom, s sočasni krmilnim tokom in s tokom krmilnih paketov.

Vodenje z zahtevo: Uporabljata se dva modela, in sicer: redukcija izrazov in redukcija grafov.

Podatkovno vodenje: Pripada mu le model računanja s tokom podatkovnih paketov.

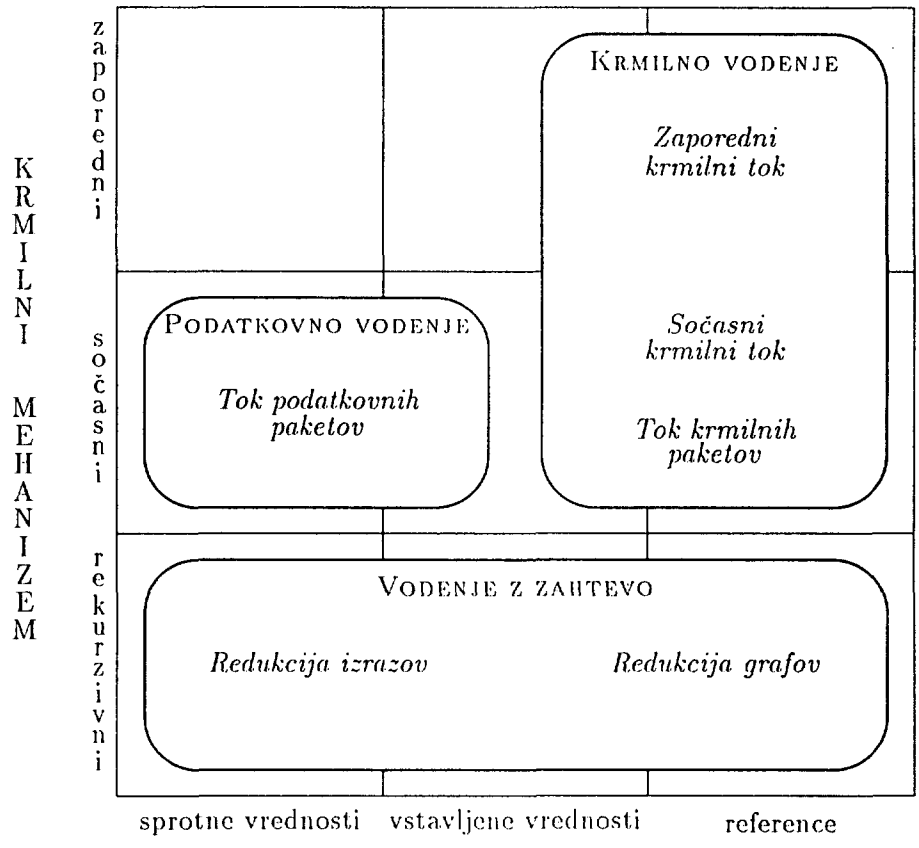
2.1.3 Organizacija stroja

Pod izrazom "organizacija stroja" razumemo način sestavljanja in medsebojnega povezovanja osnovnih računalniških materialnih virov: procesorjev, pomnilnikov in komunikacijskih enot. Ločimo tri osnovne organizacije.

Centralizirana organizacija. Sestavljajo jo procesor, pomnilnik in vodilo med njima. To je tradicionalna von Neumannova arhitektura.

Organizacija za obravnavo izrazov. Osnovni gradniki, ki so sestavljeni iz procesorja, pomnilnika in komunikacijske enote, so povezani v pravilno strukturo.

Organizacija za posredovanje paketov. Procesna enota (več procesorjev), pomnilniška enota in komunikacijska enota so preko čakalnih vrst krožno povezane. Izvrševanje programa poteka cevljeno v obliki enosmernega pretoka paketov skozi omenjene enote, ki delujejo sočasno. Vrste opravljajo nalogo začasnega skladiščenja paketov: paketi, ki so pripravljeni za obdelavo v eni od enot, čakajo v ustrezni vrsti, dokler jih ta enota ni pripravljena sprejeti.



Slika 1: Modeli računanja.

2.2 Podatkovno pretokovne arhitekture

Podatkovno pretokovni računalniki [2] se bistveno razlikujejo od krmilno vodenih računalnikov. Prvič: njihova struktura je zasnovana tako, da poteka procesiranje na osnovi *vrednosti*, ne pa naslovov spremenljivk. In drugič: v podatkovno pretokovnih računalnikih ni ničesar, kar bi bilo podobno programskemu števcu, saj postanejo operacije izvršljive takoj, ko so na voljo vsi potrebni vhodni operandi. Torej temeljijo ti računalniki na načelih *asinhronosti* in *funkcionalnosti* [34].

2.2.1 Podatkovno vodeno računanje

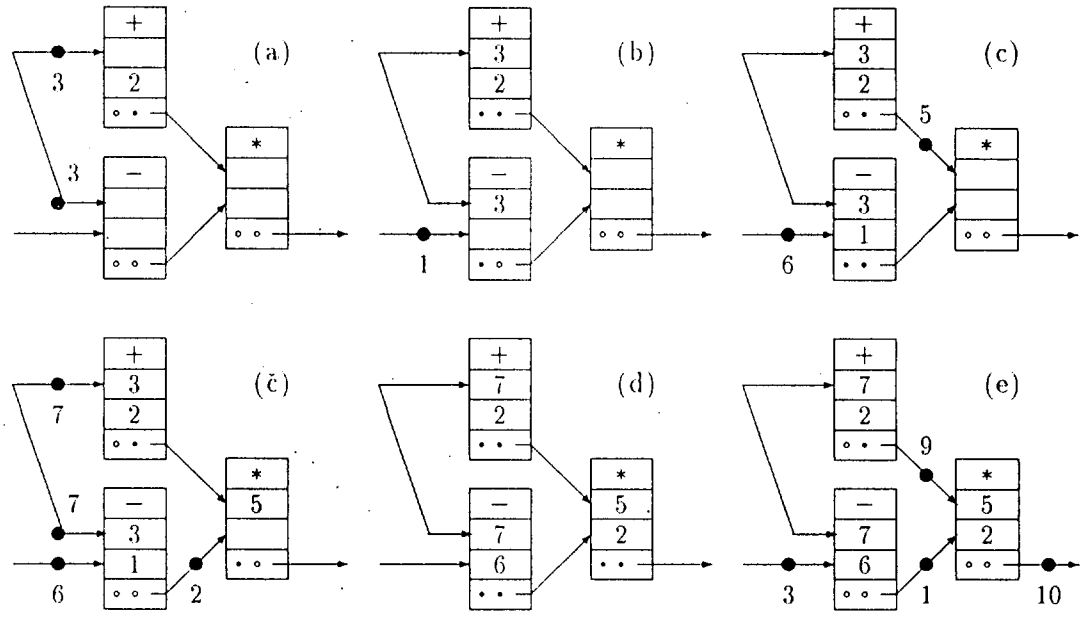
Model, s katerim realiziramo podatkovno vodeno računanje, je tok podatkovnih paketov. Opredeljujeta ga sočasni krmilni mehanizem ter podatkovni mehanizem s sprotnimi in vstavljenimi vrednostmi. Računanje se odvija s pomočjo podatkovnih paketov[2]. Ukazi sestojijo iz operacij, operandov (vstavljene ali sprotne vrednosti) in kazalcev na neposredne naslednike. Ukaz se prične izvrševati takoj, ko mu je zadnji od neposrednih predhodnikov poslal podatkovni paket.

Podatkovno vodeno računanje si ilustrirajmo s primerom $z = (x + 2) * (x - y)$, $x = 3,7\ldots$ in $y = 1,6,3\ldots$, ki je prikazan na sliki 2.

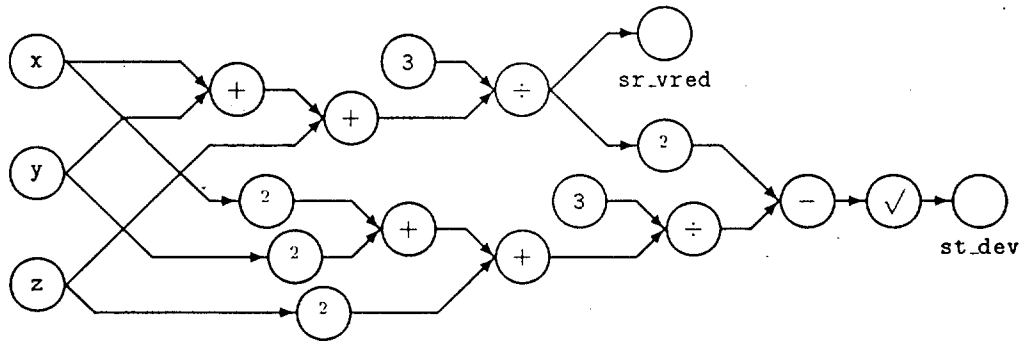
2.2.2 Graf pretoka podatkov

Strojni jezik podatkovno pretokovnih računalnikov je *graf pretoka podatkov* (*GPP*) [8], ki je posebna oblika *Petrijeve mreže* [35]. Točke v *GPP* ponazarjajo operacije, usmerjene povezave pa so nosilke podatkovnih paketov (vrednosti operandov). *GPP* je rezultat prevajanja visokega *podatkovno pretokovnega jezika* (VAL, ID, LUCID, SISAL, VALID itd.) [9]. Osnovni jezik strojnega nivoja v podatkovno pretokovnih arhitekturah je torej *GPP*. Izvrševanja programa si zatorej predočimo s pretokom podatkov v *GPP*. Ko v neko točko stečejo

[2]Uporablja se tudi izraz žeton (iz angl. "token").



Slika 2: Podatkovno vodeno računanje $z = (x + 2) * (x - y)$.



Slika 3: *GPP* funkcije `Statistika`.

podatki po vseh njenih vhodnih povezavah, postane točka *izvršljiva* in prične takoj z izvrševanjem pridružene operacije. Pravimo, da se je točka *sprožila*. Ko točka izvrševanje konča, pošlje podatkovne pakete (rezultat) v vse svoje izhodne povezave. Samoumevno je, da stoji za vsako točko *procesni element* in za vsako povezavo *komunikacijski kanal*.

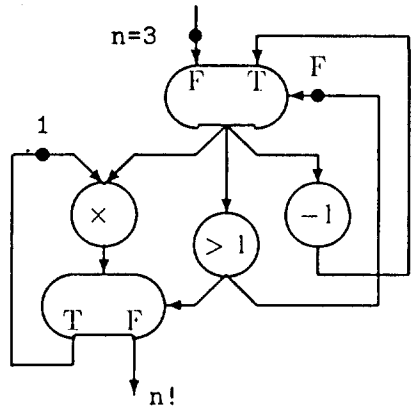
Za ilustracijo vzemimo še fukcijo `Statistika`, ki vrne srednjo vrednost `sr_vred` in standardni odklon `st_dev` vhodnih spremenljivk `x`, `y` in `z`. Zapis funkcije v visokem podatkovno pretokovnem jeziku VAL [10] je naslednji:

```
function Statistika (
         x,y,z: real
         returns real, real);
   let
      sr_vred := (x + y + z)/3;
      st_dev := SQRT((x**2 + y**2 +
                z**2)/3 - sr_vred**2);
   in
      sr_vred, st_dev
   endlet
endfun
```

V primeru izračuna funkcije `Statistika` smo dobili *aciklični GPP*. Tedaj ko v algoritmih nastopajo ponovitve (*iteracije*), pa se srečujemo s *cikličnimi GPP*. Vzemimimo npr. izračun fakto-

riele števila n. Ustrezen program za izračun n! bi se zapisal v visokem podatkovno pretokovnem jeziku ID kot (*GPP* prikazuje slika 4):

```
( initial j <- n; k <- 1
  while j > 1 do
     new j <- j - 1; new k <- k * j;
  return k )
```



Slika 4: *GPP* za izračun n!.

Če primerjamo *GPP* za izračun funkcije Statistika (slika 3) z *GPP* za izračun n! (slika 4), opazimo, da se v slednjem poleg zank pojavljajo tudi *pogojni konstrukti* ter da se po nekaterih povezavah pretakajo tudi podatki, ki predstavljajo logični vrednosti T in F. V splošnem so povezave nosilke različnih tipov podatkov: *integer, real, boolean* itd..

Pri cikličnih *GPP* pogosto nastanejo dodatne težave. Kadar v posamezni ponovitvi zanke naletimo na podatkovne odvisnosti med operacijami, to ne ustavi nadaljnjih ponovitev, čeprav predhodna ponovitev ni v celoti končana. Tako se v določenih povezavah podatki prekrivajo. Vzemimo za primer naslednji segment programa:

```
input a
      a[0] = 1
      b[0] = 1
for i from 1 to n
   begin
      a[i] = i + a[i-1]
      b[i] = a[i] * b[i-1]
   end
output b
```

Predpostavimo, da je za seštevanje potrebna 1 časovna enota in za množenje 2 časovni enoti. Pričakovani rezultat bi bil: b[0] = 1, b[1] = 2, b[2] = 8, b[3] = 56 itd. Dejansko pa bi dobili naslednje rezultate: b[0] = 1, b[1] = 2, b[2] = 14, b[3] = 308 itd. Oglejmo si nekaj prvih korakov izvajanja programa. Zaradi hitrejšega izvrševanja zanke pri seštevanju se nekateri elementi vektorja a[i] preprosto prekrijejo. Tako se a[2], ki je bil v trenutku $t = 2$ še 4, prekrije z vrednostjo 7, še preden se uporabi kot faktor pri izračunu b[2].

2.2.3 Statični in dinamični modeli računanja

Iz zadnjega primera vidimo, da ni mogoče s prisotnostjo kateregakoli podatka razglasiti točke za izvršljivo, saj v splošnem pripadajo vhodni podatki različnim delom izračuna. Obstaja nekaj rešitev nastalega problema oz. modelov podatkovno vodenega računanja, ki vplivajo tudi na izvedbo podatkovno pretokovnega računalnika.

Model 1: Vsaka ponovitev zanke se sme pričeti šele tedaj, ko se je predhodna končala. Ta model [1-1] ne dovoljuje vzporednosti med ponovitvami in zahteva dodatno ugotavljanje konca ponovitve.

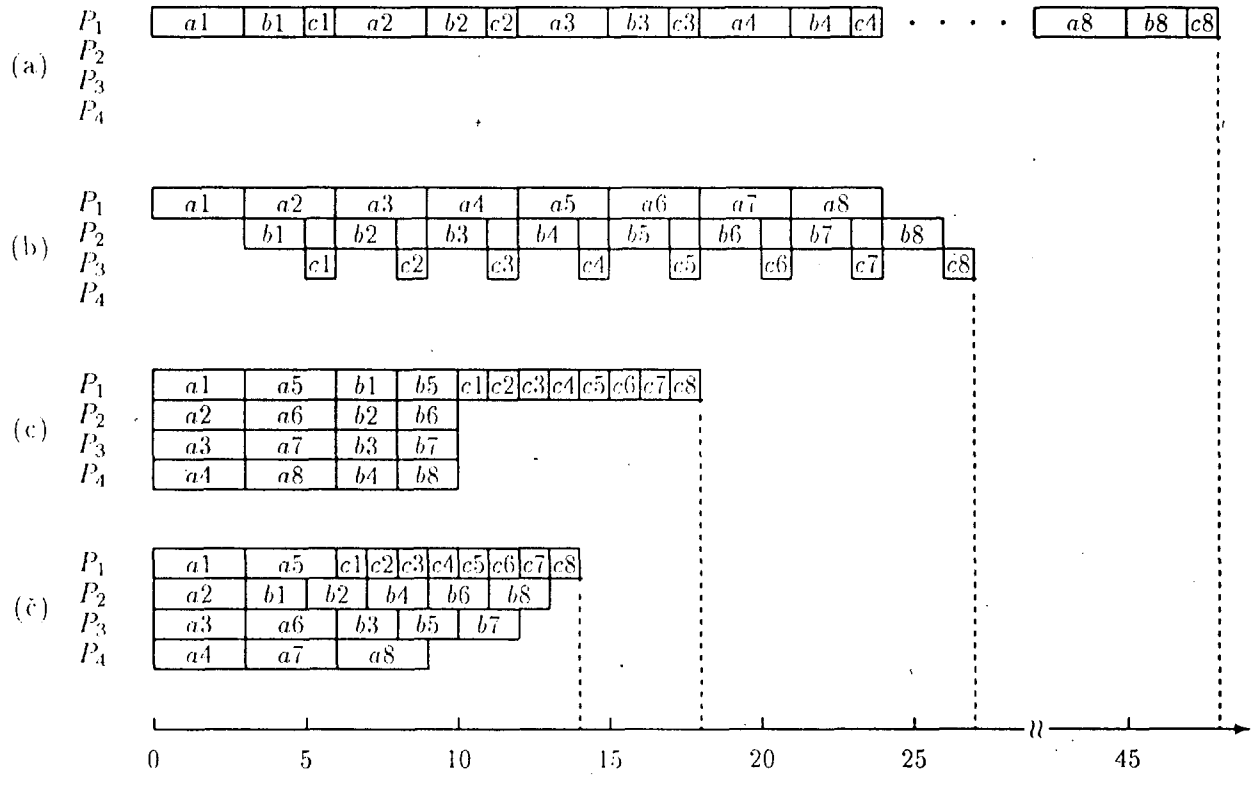
Model 2: V vsaki povezavi *GPP* dovoljujemo istočasno le en podatkovni paket [1]. To pomeni, da se sme točka sproži le tedaj, ko so prisotni podatkovni paketi v vseh vhodnih povezavah in ni v izhodnih povezavah nobenega podatkovnega paketa. Takšno pravilo izvršitve uresničimo s pomočjo *potrditvenih signalov*. Prednost tega modela je, da omogoča cevljenje, slabost pa, da se število povezav podvoji.

Model 3: Ciklični *GPP* transformiramo tako, da vsako ponovitev opišemo z acikličnim podgrafom. Takšen model zahteva večje programske pomnilnike in sprotno generiranje koda v primeru, ko je izstop iz zanke izračunan šele v času njenega izvajanja. Ti zahtevi se odražata kot resna pomankljivost v praktičnih sistemih.

Model 4: Podatkovne pakete opremimo z dodatnimi oznakami[3] [15, 36], ki vsebujejo indeks oz. nivo ponovitve. Točka se sme sproži le tedaj, ko imajo vsi vhodni podatkovni paketi enako oznako. Ta model zahteva povečan pretok po *GPP* ter mehanizme za primerjanje oznak.

Model 5: Povezave v *GPP* opravljajo tudi funkcijo vrste [37], kar pomeni, da so v njih podatkovni paketi ravrščeni v istem vrstnem redu, kot

[3] V angl. literaturi imenujejo tak paket "tagged token".



Slika 5: Primerjava modelov podatkovno vodenega računanja.

so vstopali v povezave[4]. Takšen model omogoča enako stopnjo vzporednosti kot model z označenimi paketi, le da je izvedba čakalnih vrst cenovno zelo zahtevna.

Primerjavo navedenih petih modelov podatkovno vodenega računanja si oglejmo na naslednjem testnem programu [5]:

```
input d, e, f
      c[0] = 0
for i from 1 to 8
   begin
      a[i] = d[i] / e[i]
      b[i] = a[i] * f[i]
      c[i] = b[i] + c[i-1]
   end
output a, b, c
```

Predpostavimo, da so potrebne za deljenje tri, za množenje dve in za seštevanje ena časovna enota. Zamislimo si podatkovno pretokovni računalnik s štirimi procesorji P_1, P_2, P_3 in P_4, ki izvajajo katerokoli od potrebnih operacij. Idealizirajmo računalnik tako, da ne bo zakasnitev v pomnilniku in pri medprocesorskih komunikacijah. Nadalje definirajmo *pospešitev* računanja S_p in *izkoriščenost* procesorjev E_p, s katerima bomo merili kvaliteto računalnika. Če je T_1 čas potreben za zaporedno izvrševanje programa in T_p čas izvrševanja istega programa na p procesorjih, potem je

$$S_p = \frac{T_1}{T_p} \text{ in } E_p = \frac{S_p}{p}.$$

Za naš testni program bi dobili $T_1 = 48$. V tabeli 1 so povzeti rezultati primerjave posameznih modelov podatkovno vodenega računanja. Podrobnejša analiza je prikazana tudi na sliki 5.

Model 1: Tak model nas pripelje do zaporednega izvajanja programa, saj se naslednja ponovitev sme pričeti šele, ko se predhodna konča. Za takšno izvrševanje zadošča en procesor (slika 5a).

Model 2: Ker dovoljujemo v povezavah istočasno le en podatkovni paket, nas to privede do cevljenja prireditvenih stavkov znotraj zanke. Za takšno izvrševanje bi zadoščali že trije procesorji (slika 5b).

[4] Povezave so FIFO vrste.

$p = 4$	T_p	S_p	E_p
Model 1	48	1	.250
Model 2	27	1.778	.444
Modeli 3,4,5	14	3.429	.857

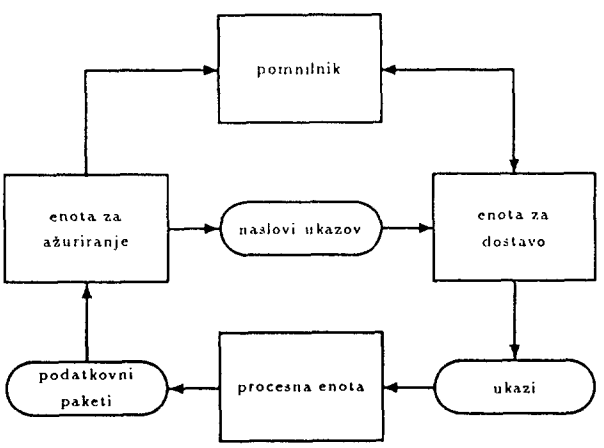
Tabela 1: Primerjava modelov podatkovno vodenega računanja.

Modeli 3, 4, 5: Najboljše rezultate dobimo pri statičnem računanju z razvitjem zank ter pri obeh dinamičnih modelih (slika 5č).

Zaradi primerjave je na sliki 5c prikazan tudi potek računanja na vektorskih računalnikih [19]. Med prevajanjem testnega programa bi se prva dva prireditvena stavka v zanki vektorizirala, tako da bi dobili $T_p = 18, S_p = 2.667$ in $E_p = 0.667$.

2.2.4 Modeli računalnikov

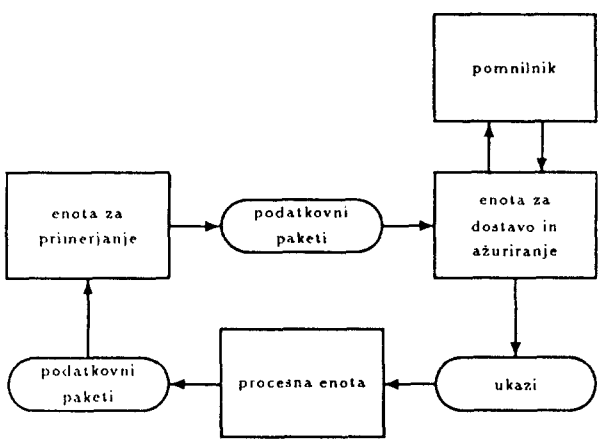
Z razvojem VLSI tehnologije so postale uresničljive mnoge ideje v računalniških arhitekturah non-von Neumannovega tipa. Tako je bilo v zgodnjih osemdesetih letih mogoče realizirati tudi prve podatkovno pretokovne računalnike. Dober pregled in primerjavo zgodnjih podatkovno pretokovnih arhitektur je podal *Srini* v delu [27]. Nekatere novejše arhitekture pa so predstavljene v [38, 28, 39, 40, 7]. Podatkovno pretokovno računanje (statično in dimanično) običajno uresničimo na računalnikih, ki temeljijo na organizacijah za posredovanje paketov. Oglejmo si arhitekturi, ki sta značilni za statični in dinamični model podatkovno vodenega računanja.



Slika 6: Statična arhitektura.

Statična arhitektura. V statični arhitekturi (slika 6) je uporabljen sinhronizacijski mehanizem, ki temelji na *shranjevanju paketov*. Podatkovni paketi se shranjujejo v ukaze, ki se izvrše, ko prejmejo vse vhodne operande. Ukazi se nahajajo v *pomnilniku* in vsebujejo: operacijo, prostor za vhodne operande in kazalce na neposredne naslednike. Enota za *ažuriranje* hrani naslov ukaza ter števec manjkajočih vhodnih operandov (podatkovnih paketov). Podatkovni paket, ki zapusti *procesno* enoto, vsebuje poleg rezultata tudi naslov ukaza, ki mu je namenjen. Enota za ažuriranje posreduje rezultat ustreznim ukazom v pomnilniški enoti ter zmanjša števec manjkajočih vhodnih operandov za ena. V primeru, ko pade števec na nič, posreduje naslov ukaza enoti za *dostavo*. Leta pošlje naslovljeni ukaz iz pomnilnika v *procesno* enoto.

Značilni prestavniki statičnih arhitektur so: SDFM[5] (Tehnološki institut Massachusetts, ZDA) [1], LAU[6] večprocesorski sistem (CERT -- Toulouse, Francija) [14], DDP[7] (Texas Instruments, ZDA) [13], μPD7281[8] (NEC, Japonska) [11] itd..



Slika 7: Dinamična arhitektura.

Dinamična arhitektura. Za razliko od statične arhitekture je v dinamični arhitekturi (slika 7) uporabljen sinhronizacijski mehanizem, ki temelji na *primerjanju paketov*. Ustrezno označeni podatkovni paketi, ki zapustijo *procesno* enoto, se zbirajo v množicah v enoti za *primerjanje*. Množice so opremljene z naslovi pripadajočih ukazov v *pom-*

[5] Static Dataflow Machine.

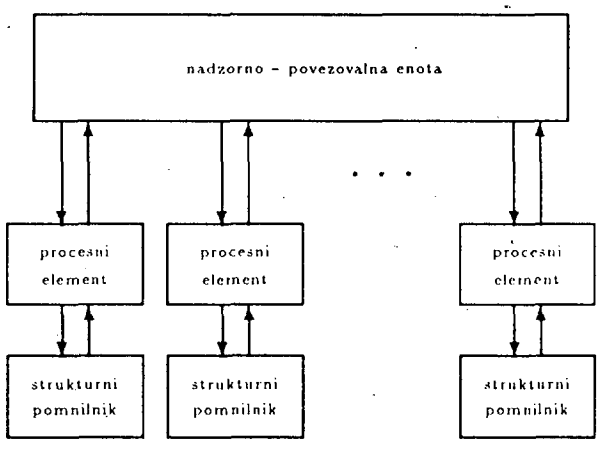
[6] "Language á assignation unique" pomeni jezik z enkratno prireditvijo.

[7] Distributed Data Processor.

[8] Prvi VLSI podatkovno pretokovni mikroprocesor.

nilniku. Enota za primerjanje razporeja prispele pakete v množice glede na njihove oznake. Ko se množica napolni, se vsi paketi prenesejo v enoto za *dostavo in ažuriranje.* Le-ta združi prispele pakete s pripadajočim ukazom iz pomnilnika ter jih posreduje procesni enoti.

Med dinamične arhitekture uvrščamo: TTDFM[9] (Tehnološki institut Massachusetts, ZDA) [41], MDFC[10](Univerza v Manchestru, Velika Britanija) [16], PIM-D[11] (ICOT, Japonska) [42], PATTSY[12] (Univerza v Queenslandu, Avstralija) [43] itd..



Slika 8: Hibridna arhitektura.

Hibridna arhitektura. Poleg omenjenih statičnih in dinamičnih arhitektur se predvsem v zadnjem času pojavljajo mnoge izvedbe arhitektur, ki jih ni mogoče uvrstiti v omenjeni dve skupini. Za te arhitekture je značilna organizacija, ki jo prikazuje slika 8. *GPP* se porazdeli med *strukturne pomnilnike* ob posameznih *procesnih elementih.* Procesni elementi izvršujejo pripadajoče dele *GPP* ter preko *nadzorno - povezovalne* enote izmenjujejo podatkovne pakete, pripadajoče delom *GPP*, ki se nahajajo v drugih strukturnih pomnilnikih. Izvrševanje *GPP* poteka bodisi statično bodisi dinamično.

Nekateri novejši računalniki iz tega razreda so: HDFM[13] (Hughes Aircraft Co., ZDA) [17], Sigma-1[14] (Electrotechnical Laboratory, Japonska) [18], EPSILON-2[15] (Sandia National Laboratories, ZDA) [44], Monsoon[16] (Tehnološki institut Massachusetts, ZDA) [45] itd..

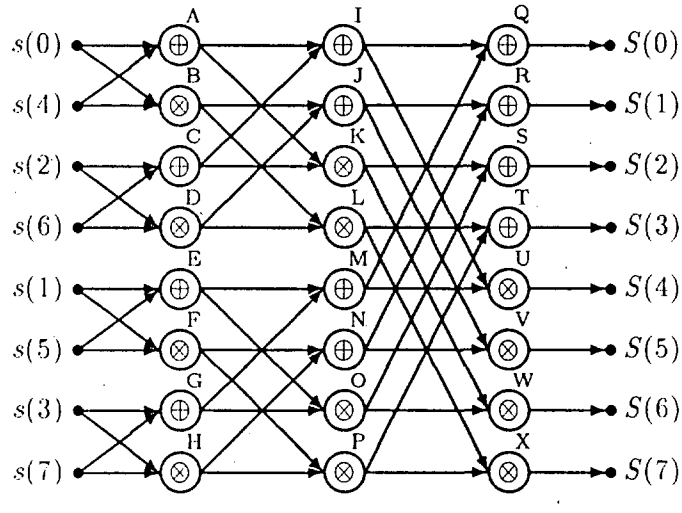
[9]Tagged-Token Dataflow Machine
[10]Manchester Dataflow Computer.
[11]Parallel Inference Machine.
[12]Processor Array Tagged-Token System.
[13]Hughes Dataflow Multiprocessor (statična arhitektura).
[14]Dinamična arhitektura.
[15]Hibridno krmilno-podatkovno vodeno računanje.

3. Motivacija in opis problema

Maloprej smo videli, da se tako v statičnih kot v dinamičnih arhitekturah izvršljivi podatkovni paketi (ukazi) preko vrste dodeljujejo procesorjem. Žal pa ima procesna enota omejeno število procesorjev, zato se v primeru, ko so vsi procesorji zasedeni, paketi kopičijo v vrsti pred procesno enoto, kjer čakajo na svojo izvršitev. Posledica takšnega čakanja je lahko občutno upočasnjeno izvrševanja *GPP*. Da se temu izognemo, je potrebno bodisi povečati število procesorjev v procesni enoti ali pa zagotoviti ustrezen način vstopanja paketov v čakalno vrsto (ali izstopanja iz vrste) pred procesno enoto in ustrezeno dodeljevanje paketov prostim procesorjem.

3.1 Primer računanja FFT

Problem upočasnjenega izvrševanja *GPP*, ki nastane zaradi pomanjkanja procesorjev, osvetlimo s primerom izračuna hitre Fourierjeve transformacije (FFT) na 8 točkah (slika 9).



Slika 9: *GPP* za FFT z 8 točkami.

Osnovni operaciji sta seštevanje (označeno z $\oplus$) ter odštevanje in množenje (označeno z $\otimes$). Če sta I_m in I_n vhodna podatka, potem velja

$$I_m \oplus I_n \equiv I_m + In$$

[16]Dinamična arhitektura.

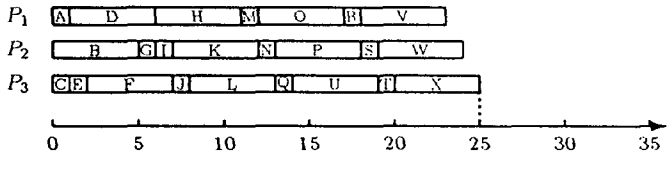
in

$$I_m \otimes I_n \equiv (I_m - I_n) \times e^{2\pi ik/8}, 0 \leq k \leq 7.$$

Predpostavimo, da je za izvršitev operacije $\oplus$ potrebna 1 časovna enota in za izvršitev operacije $\otimes$ 5 časovnih enot. Potem je čas potreben za zaporedno izvrševanje FFT algoritma $T_1 = 72$ časovnih enot. Vzporedno izvrševanje algoritma bi ob predpostavki, da imamo na voljo neomejeno mnogo procesorjev, zahtevalo le $T_\infty = 15$ časovnih enot. Torej je idealna pospešitev FFT algoritma $S_\infty = 4.8$.

3.1.2 Primer 1: Premalo procesorjev

Naj bo dan podatkovno pretokovni računalnik s tremi procesorji P_1, P_2 in P_3, ki morejo izvajati katerokoli od operacij $\oplus$ in $\otimes$. Idealizirajmo računalnik tako, da ne bo zakasnitev v pomnilniku in pri medprocesorskih komunikacijah. Predpostavimo statično arhitekturo (slika 6), ki podpira podatkovno pretokovno računanaje opredeljemo z modelom 3. Točke GPP bi se porazdelile med procesorje P_1, P_2 in P_3, kot je prikazano na sliki 10. Za ta primer dobimo $T_p = 25, S_p = 2.88$ in $E_p = 0.96$. Pospešitev S_p je manjša od S_∞.

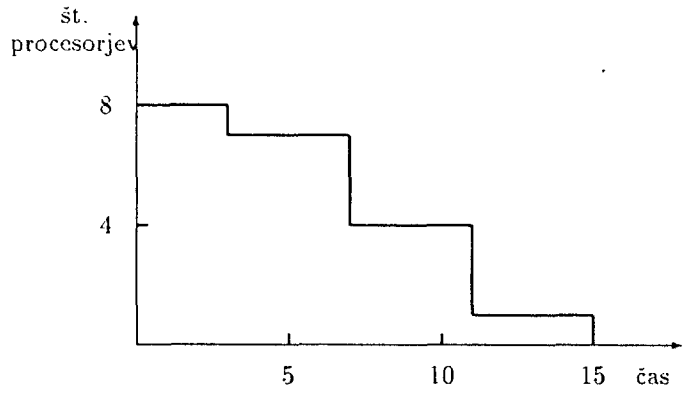


Slika 10: Izvrševanje FFT pri $p = 3$.

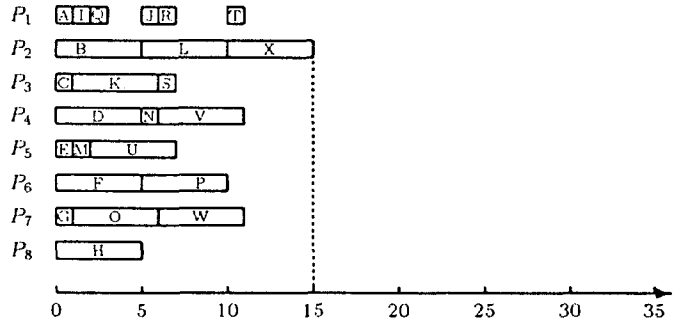
3.1.2 Primer 2: Optimizacija procesorjev

Predpostavimo enako arhitekturo kot v primeru 1, le da vsebuje procesna enota neomejeno mnogo procesorjev. Naj nam bo število delujočih procesorjev v trenutku t merilo vzporednosti algoritma v tem trenutku. Za FFT algoritem je prikazan časovni potek vzporednost na sliki 11. Opazimo, da vzporednost niha med 1 in 8 ter da je povprečna vzporednost enaka idealni pospešitvi algoritma $S_\infty = 4.8$. Postavlja se vprašanje, koliko procesorjev p bi zadoščalo, da bi veljalo $S_p = S_\infty$. Če bi se odločili za število procesorjev, ki je določeno z maksimalno vzporednostjo, tj. $p = 8$, bi dobili rezultat, ki je prikazan na sliki 12.

Dobimo $T_p = 15$ in $S_p = 4.8$. S tem smo zagotovili, da je $S_p = S_\infty$, žal pa je izkoriščenost procesorjev le $E_p = 0.6$. E_p je mogoče povečati le tako,



Slika 11: Dinamična vzporednost FFT algoritma.



Slika 12: Izvrševanje FFT pri $p = 8$.

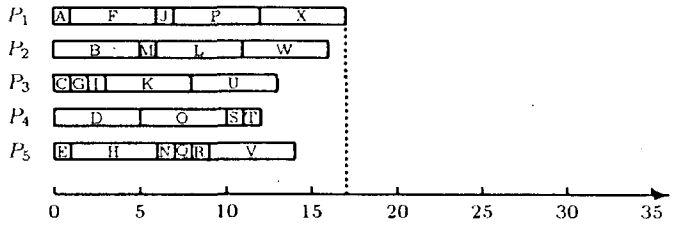
da uporabimo manj procesorjev. Sedaj je mogoče uvesti prvi problem, ki ga želimo rešiti, tj. *optimizacija procesorjev.*

Optimizacija procesorjev: *Za dani GPP moramo določiti takšno število procesorjev p, da bo $S_p = S_\infty$ in da bo E_p čim bliže 1.*

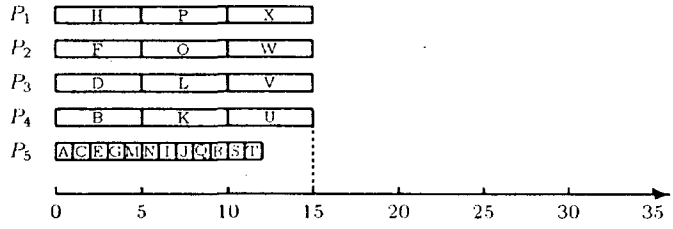
Morda bi zadoščalo, če bi vzeli toliko procesorjev, kot jih določa povprečna vzporednost S_∞. V našem primeru bi bilo $p = \lceil S_\infty \rceil = 5$. To možnost prikazuje slika 13. V tem primeru je rezultat naslednji: $S_p = 4.235$ in $E_p = 0.847$, kar pomeni, da je $S_p < S_\infty$.

Pojavi se vprašanje, ali bi bilo mogoče zagotoviti ustrezen način vstopanja paketov v čakalno vrsto (ali izstopanja iz vrste) pred procesno enoto in s tem takšno dodeljevanje paketov prostim procesorjem, da bi uspeli v primeru $p = 5$ zagotoviti tudi $S_p = S_\infty$. V našem primeru je to mogoče, kar prikazuje slika 14. Rezultat, ki ga dobimo po takšni optimizaciji, je: $S_p = 4.8$ in $E_p = 0.96$.

Postopek *optimizacije procesorjev* bomo opisali v



Slika 13: Izvrševanje FFT pri $p = 5$.



Slika 14: Optimizirano izvrševanje FFT pri $p = 5$.

nadaljevanju.

3.1.3 Primer 3: Optimizacija časa

Tudi sedaj predpostavimo enako arhitekturo kot v primeru 1; torej podatkovno pretokovni računalnik s $p = 3$ procesorji P_1, P_2 in P_3. Upeljimo najprej *upad idealne pospešitve* računanja D_p, ki ga definiramo kot

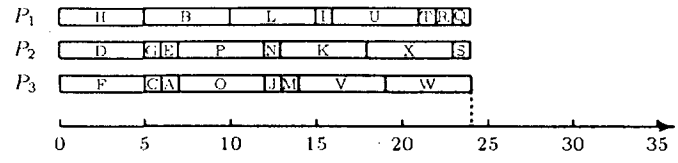
$$D_p = \frac{S_\infty - S_p}{S_p} = \frac{T_p - T_\infty}{T_\infty}.$$

V primeru 1 dobimo $D_p = 0.667$, kar pomeni 66.7% upad idealne pospešitve. Tudi tokrat je mogoče zagotoviti ustreznejši način vstopanja paketov v čakalno vrsto (oz. izstopanja iz vrste) pred procesno enoto in s tem zagotoviti manjši upad idealne pospešitve. Rešitev prikazuje slika 15. Rezultat, ki ga dobimo po takšni optimizaciji, je: $S_p = 3, D_p = 0.6$. V tem primeru dobimo tudi $E_p = 1$, kar pomeni, da smo dosegli tudi največjo izkoriščenost procesorjev. Ker se bomo v nadaljevanju še sklicevali na to rešitev, jo poimenujmo "optimalna".

Uvedimo sedaj drugi problem, ki ga želimo rešiti, tj. *optimizacija časa.*

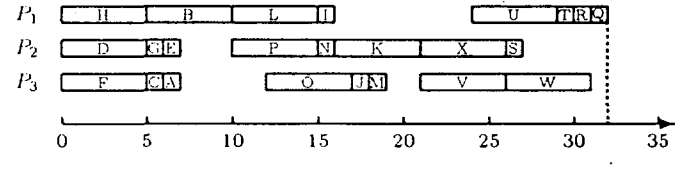
Optimizacija časa: *Za dani GPP in dano število procesorjev p, moramo določiti takšno dodeljevanje, da bo D_p čim bliže* 0.

Poizkusimo problem optimizacije časa še nekoliko zaplesti. Vzemimo sedaj hibridno arhitekturo



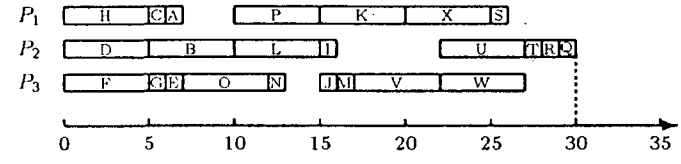
Slika 15: Optimalno izvrševanje FFT pri $p = 3, t_c = 0$.

(slika 8). Število procesorjev naj ostane $p = 3$, predpostavimo pa, da je za *medprocesorsko komunikacijo* t_c potrebnih 5 časovnih enot. Dodeljevanje procesorjev, kot je bilo v primeru "optimalne" rešitve, rezultira v $S_p = 2.25, D_p = 1.133$ in $E_p = 0.75$ (slika 16).



Slika 16: Izvrševanje FFT pri $p = 3, t_c = 5$.

S primernim dodeljevanjem procesorjev je mogoče doseči določene izboljšave. Tako npr. z rešitvijo, ki je podana na sliki 17, dosežemo $S_p = 2.4, D_p = 1$ in $E_p = 0.8$. Za izboljšanje D_p gre zahvala predvsem zmanjšanju medprocesorskih komunikacij. Uporabljenih je le 14 medprocesorskih komunikacij, medtem ko jih je bilo v prejšnjem primeru kar 24.



Slika 17: Optimizirano izvrševanje FFT pri $p = 3, t_c = 5$.

Tudi postopek *optimizacije časa* bomo opisali v nadaljevanju.

3.2 Primerjava rešitev

Iz dosedanje analize (tabela 2) vidimo, da tako problem optimizacije procesorjev kakor tudi optimizacije časa uspešno rešimo le, če zagotovimo ustrezen način dodeljevanja paketov prostim pro-

cesorjem (preko čakalne vrste), oz. zagotovimo ustrezno porazdelitev GPP med procesorje. Kot bomo videli kasneje, je to moč doseči le s preoblikovanjem podatkovno vodenega računanja.

p	t_c	T_p	S_p	D_p	E_p	vodenje računanja
8	0	15	4.8	0	0.6	podatkovno
5	0	17	4.235	0.133	0.847	podatkovno
3	0	25	2.88	0.667	0.96	podatkovno
5	0	15	4.8	0	0.96	mod. podatkovno
3	0	24	3	0.6	1	mod. podatkovno
3	5	33	2.182	1.2	0.727	podatkovno
3	5	30	2.4	1	0.8	mod. podatkovno

Tabela 2: Primerjava računanja FFT.

3.3 NP-polnost problema

Sedaj bomo spoznali, da je problem *optimizacije časa* NP-poln. Neposredna posledica tega je tudi NP-polnost problema *optimizacije procesorjev*. Ti dejstvi sta pomembni, saj opravičujeta uporabo hevrističnih metod pri reševanju obeh problemov.

Kot vemo, je $\mathcal{NP}$ razred problemov, ki so nedeterministično rešljivi v polinomskem času (in zato deterministično rešljivi v eksponentnem času). Nekateri med njimi so deterministično rešljivi celo v polinomskem času in tvorijo razred $\mathcal{P}$. Seveda je $\mathcal{P} \subseteq \mathcal{NP}$. Odprto pa je vprašanje, ali je $\mathcal{P} \neq \mathcal{NP}$. Zato ne vemo, ali obstaja problem, ki je deterministično rešljiv v eksponentnem, ne pa tudi polinomskem času.

Nekateri problemi iz razreda $\mathcal{NP}$ imajo še posebej zanimivo in pomembno lastnost, ki se kaže v tem, da moremo reševanje kateregakoli problema iz $\mathcal{NP}$ v polinomskem času prevesti na reševanje kakega od teh odlikovanih problemov, imenovanih NP-polni problemi. V tem smislu so NP-polni problemi med najtežjimi v razredu $\mathcal{NP}$. Če bi nam uspelo pokazati za vsaj enega izmed njih, da je deterministično rešljiv v polinomskem času, bi s tem pokazali, da je v polinomskem času rešljiv vsak problem iz $\mathcal{NP}$. Kljub velikim naporom pa doslej kaj takega še nikomur ni uspelo, čeprav zbirka znanih NP-polnih problemov šteje že nekaj sto problemov. Zato prevladuje prepričanje, da NP-polni problemi v resnici niso deterministično polinomsko rešljivi. Praktična posledica tega pa je, da se v primeru, ko naletimo na NP-poln problem, vedemo pragmatično, torej ne iščemo polinomskega algoritma (zaradi majhne verjetnosti, da bi ga našli), ampak raje pričnemo sestavljati hevristični algoritem.

Problem optimizacije časa ter problem optimizacije procesorjev spadata v razred NP-polnih problemov! Prvi se v literaturi pojavlja pod imenom DODELJEVANJE UREJENIH OPRAVIL (PCS[17]) v naslednji obliki

Naj bo dana množica $\mathcal{V} = \{v_1, \ldots, v_n\}$ *opravil dolžine* 1, *ki so urejena z relacijo* $\longmapsto$ *stroge delne urejenosti,* $p \in \mathbb{N}$ *procesorjev ter mejni čas* $T \in \mathbb{N}$. *Ali obstaja takšna preslikava* s *množice* $\mathcal{V}$ *v množico* $\{0, 1, \ldots, T\}$, *da za vse* $\tau \in \{0, 1, \ldots, T\}$ *ter* $u, v \in \mathcal{V}$ *velja* $| s^{-1}(\tau) | \leq p$ *in hkrati* $u \longmapsto v \Rightarrow s(u) < s(v)$?

Dokaz NP-polnosti problema PCS je podan v [46]. Ker imajo vsa opravila enako časovno zahtevnost, je PCS le posebna oblika problema optimizacije časa - torej je slednji gotovo NP-poln. NP-polnost problema optimizacije procesorjev pa pokažemo tako, da nanj polinomsko časovno prevedemo problem PCS [47]. Če namreč nek NP-poln problem A polinomsko časovno prevedemo na problem $B \in \mathcal{NP}$, potem je (zaradi tranzitivnosti relacije polinomske prevedljivosti) tudi B NP-poln.

3.3.1 Reševanje NP-polnih problemov

Kot smo uvodoma omenili, imajo najboljši znani algoritmi za reševanje NP-polnih problemov vsi eksponentno časovno kompleksnost. Zaradi hitre rasti eksponentne funkcije to običajno pomeni, da ne znamo dobiti optimalnih rešitev že pri relativno majhnih problemih. Eden od pristopov k reševanju NP-polnega problema je zato pospešitev danega algoritma, tj. pospešitev pregledovanja množice možnih rešitev. Takšen pristop ponuja delno rešitev ter sproti preverja, ali le-ta vodi do optimalne rešitve. Kadar temu ni tako, delno rešitev opusti ter sestavi kako drugo delno rešitev. Med takšne pristope spada npr. *razveji in omeji.*

Drug pristop pri reševanju NP-polnega problema je razvoj algoritma, ki v sprejemljivem času vrne rešitev, ki usreza danim zahtevam. Takšni algoritmi se imenujejo *hevristični.* Dve družini hevrističnih algoritmov obsegata *aproksimacijske* ter *verjetnostne* algoritme. Aproksimacijski algoritmi vračajo rešitve, za katere je znano, koliko največ utegnejo odstopati od optimalne reštve. Verjetnostni algoritmi [48] pa ob polinomski časovni zatevnosti dovolj pogosto (z določeno verjetnostjo) vračajo optimalne rezultate. Slednji bodo

[17] Precedence Constrained Scheduling

uporabljeni pri reševanju problemov optimizacije časa in optimizacije procesorjev.

Literatura

[1] J. B. Dennis. The Variaties of Data Flow Computers. In *Proc. First Int'l Conf. Distributed Comp. Sys.*, pages 430–439, October 1979.

[2] J. B. Dennis.. Data Flow Supercomputers. *IEEE Computer*, 13(11):48–56, November 1980.

[3] S. Ribarić. Računari upravljani tokom podataka. *Informatica*, 6(4):3–11, 1982.

[4] P. C. Treleaven, D. R. Brownbridge, and R. P. Hopkins. Data-Driven and Demand-Driven Computer Architectures. *Computing Surveys*, 14(1):93–143, March 1982.

[5] J. Šilc in B. Robič. Osnovna načela DF sistemov. *Informatica*, 9(2):10–15, 1985.

[6] J. B. Dennis. Dataflow Computation: A Case Study. In V. M. Milutinović, editor, *Computer Architecture - Concepts and Systems*, pages 354–404. North-Holland, 1988.

[7] L. Bic and J. L. Gaudiot. Special Issue on Data-Flow Computing. *Journal of Parallel and Distributed Computing*, 10(4):277–385, December 1990.

[8] A. L. Davis and R. M. Keller. Data Flow Program Graphs. *IEEE Computer*, 15(2):26–41, February 1982.

[9] W. B. Ackerman. Data Flow Languages. In *Proc. National Comp. Conf.*, pages 1087–1095, June 1979.

[10] J. R. McGraw. Data-Flow Computing: The VAL Language. *ACM Trans. Prog. Languages and Systems*, 4(1):44–82, January 1982.

[11] T. Jeffery. The μPD7281 Processor. *Byte*, pages 237–246, November 1985.

[12] S. Weiss, I. Spillinger, and G. M. Silberman. Architectural Improvements for Data-Driven VLSI Processing Arrays. In *Proc. Functional Programming Languages and Computer Architecture*, pages 243–259, September 1989.

[13] M. Cornish. The TI Dataflow Architecture: The Power of Concurrency for Avionics. In *Proc. 3rd Conf. Digital Avionics Systems*, pages 19–25, November 1979.

[14] D. Comte, N. Hifdi, and J. C. Syre. The Data Driven LAU Multiprocessor System: Results and Perspectives. In *Proc. World Comp. Congress IFIP'80*, pages 175–180, October 1980.

[15] J. Gurd and I. Watson. Preliminary Evaluation of a Prototype Dataflow Computer. In *Proc. 9th World Comp. Congress IFIP'83*, pages 545–551, September 1983.

[16] J. R. Gurd C. C. Kirkham, and I. Watson. The Manchester Prototype Dataflow Computer. *Comm. ACM*, 28(1):34–52, January 1985.

[17] R. Vedder, M. Campbell, and G. Tucker. The Hughes Data Flow Multiprocessor. In *Proc. 5th Int'l Conf. Distributed Computing Systems*, pages 2–9, May 1985.

[18] T. Shimada, K. Hiraki, K. Nishida, and S. Sekiguchi. Evaluation of a Prototype Data Flow Processor of the SIGMA–1 for Scientific Computations. In *Proc. 13th Annual Int'l Symp. Comp. Arch.*, pages 226–234, June 1986.

[19] K. Hwang and F. A. Briggs. *Computer Architecture and Parallel Processing*. McGraw-Hill Book Company, 1984.

[20] B. Robič in J. Šilc. Razvrstitev novogeneracijskih računalniških arhitektur. *Informatica*, 10(4):18–32, 1986.

[21] N. Gehani and A. D. McGettrick, editors. *Concurrent Programming*. Addison-Wesley Publishing Comp., 1988.

[22] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. The Benjamin/Cummings Publishing Comp., 1989.

[23] V. M. Milutinović, editor. *Computer Architecture - Concepts and Systems*. North-Holland, 1988.

[24] L. M. Patnaik, R. Govindarajan, M. Špegel, and J. Šilc. A Critique on Parallel Computer Architectures. *Informatica*, 12(2):47–64, 1988.

[25] V. M. Milutinović, editor. *High-Level Computer Architecture*. Computer Science Press, 1989.

[26] R. Duncan. A Survey of Parallel Computer Architectures. *IEEE Computer*, 23(2):5–16, February 1990.

[27] V. P. Srini. An Architectural Comparison of Dataflow Systems. *IEEE Computer*, 19(3):68–88, March 1986.

[28] J. Šilc and B. Robič. The Review of Some Data Flow Computer Architectures. *Informatica*, 11(1):61–66, 1987.

[29] T. Y. Feng. Some Caracteristic of Associative/Parallel Processing. In *Proc. 1972 Sagamore Computing Conf.*, pages 5–16, August 1972.

[30] W. Händler. The impact of classification schemas on computer architecture. In *Proc. 1977 Int'l Conf. Parallel Processing*, pages 7–15, August 1977.

[31] M. J. Flynn. Some Computer Organizations and Their Effectiveness. *IEEE Trans. Computers*, C-21(9):948–960, September 1972.

[32] D. B. Skillicorn. A Taxonomy for Computer Architectures. *IEEE Computer*, 21(11):46–57, November 1988.

[33] S. Dasgupta. A Hirarchical Taxonomic System for Computer Architectures. *IEEE Computer*, 23(3):64–74, March 1990.

[34] D. D. Gajski, D. A. Padua, D. J. Kuck, and R. H. Kuhn. A Second Opinion on Data Flow Machines and Languages. *IEEE Computer*, 15(2):58–89, February 1982.

[35] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[36] Arvind and D. E. Culler. Dataflow Architectures. *Ann. Rev. Comput. Sci.*, 1:225–253, 1986.

[37] A. L. Davis. A Dataflow Evaluation System Based on the Concept of Recursive Locality. In *Proc. 1979 Nat. Computer Conf.*, pages 1079–1086, 1979.

[38] J. Šilc and B. Robič. Data Flow Based Parallel Inference Machine. *Informatica*, 11(4):27–34, 1987.

[39] A. V. S. Sastry, L. M. Patnaik, and J. Šilc. Dataflow Architectures for Logic Programming. *Elektrotehniški vestnik*, 55(1):9–19, januar 1988.

[40] J. Šilc and B. Robič. Efficient Dataflow Architecture for Specialized Computations. In *Proc. 12th World Congress on Scientifics Computation*, pages 4.681–4.684, July 1988.

[41] Arvind and V. Kathail. A Multiple Processor Dataflow Machine That Supports Generalized Procedures. In *Proc. 8th Annual Int'l Symp. Comp. Arch.*, pages 291–302, May 1981.

[42] N. Ito, M. Sato, E. Kuno, and K. Rokusawa. The Architecture and Preliminary Evaluation Results of the Experimental Parallel Inference Machine PIM-D. In *Proc. 13th Annual Int'l Symp. Comp. Arch.*, pages 149–156, June 1986.

[43] V. L. Narasimhan and T. Downs. Operating System Features of a Dynamic Dataflow Array Processing System (PATTSY). In *Proc. 3rd Annual Parallel Processing Symp.*, pages 722–740, March 1989.

[44] V. G. Grafe and J. E. Hoch. The Epsilon-2 Multiprocessor System. *Journal of Parallel and Distributed Computing*, 10(4):309–318, December 1990.

[45] D. E. Culler and G. M. Papadopoulos. The Explicit Token Store. *Journal of Parallel and Distributed Computing*, 10(4):289–308, December 1990.

[46] K. Mehlhorn. *Graph Algorithms and NP-Completeness*. Springer-Verlag, 1984.

[47] B. Robič. Minimizacija števila procesorjev v podatkovno pretokovni arhitekturi. Magistersko delo, Fakulteta za elektrotehniko, Ljubljana, 1987.

[48] J. Žerovnik. *Verjetnost v kombinatorični optimizaciji*. Doktorska disertacija, Fakulteta za elektrotehniko in računalništvo, Ljubljana, 1992. V pripravi.

Novice in zanimivosti

Fotonika namesto elektronike I

Zakaj dovršitev elektronike?

Če razumemo tehniko vselej kot dovršitev določene tehnike in začetek nove, potem je podobno moč razumeti dovršitev elektronike in začetek fotonike. Smisel znanosti in tehnike se utemeljuje vselej v določeni dovršitvi starega in začetku novega. Za vsakim modernizmom prihaja postmodernizem, ki preide v modernizem, za katerim se dogaja njemu sledeči postmodernizem.

Fotonika je bržkone nekaj povsem novega v znanstvenotehnološkem smislu, kjer je delež znanosti na eni strani in tehnologije na drugi radikalno različen od prejšnjega, elektronskega, ko gre vnovič za razmerje mikro in makro sveta, individualne in masovne fenomenalnosti, dovršenosti starega in nujnosti novega. Nova tehnologija se s fotoniko začne tudi praktično in ne le teorijsko pomikati v svet kvantne fizike in področij, ki so danes še onkraj nje same. V smislu artikuliranja tehnoloških generacij pomeni fotonika mutacijo, predvsem kot tehnološko odvrnitev oziroma — biološko primerjalno — genetski razcep, ko postaja tehnika kot intelektualna vsebina del evolucije in tudi metaevolucija človekovega biološkega razvoja.

Laboratoriji največjih (AT&T, IBM, Japan & Co.) v informacijski tehnologiji delajo mrzlično. V naslednjih desetih letih in vse do leta 2010 naj bi informacijska in tudi telekomunikacijska industrija doživela bistvene spremembe. Ključ informacijske tehnologije je elektronika, natančneje mikroelektronika. Ta narekuje število komponent, ki jih je mogoče povezati na silicijskem vezju, velikem kot noht na prstu. To število se približno podvoji vsakih 18 mesecev. V začetku šestdesetih let je bil na integriranem vezju (čipu) en sam aktivni element (tranzistor), leta 1970 pa že 100. Leta 1980 je število komponent na vezju narastlo že na 100000 (10^5) in leta 1992 na 10^7 (sto milijonov). V letu 2010 naj bi bila dosežena fizikalna meja 10^{10} komponent na eno vezje, tj. deset milijard.

Trend, o katerem govorimo, bo dozorel okoli leta 2010, ko bo na enem vezju več milijard komponent, zlasti na pomnilnih vezjih. Vezja z logiko po naročilu bodo vsebovala od sto milijonov do ene milijarde komponent. Število komponent na vezje bo omejevala ekonomija in ne fizikalne zmogljivosti. Cena ene produktne linije za kompleksna vezja se bo namreč gibala med eno do deset milijard dolarjev. Predvideva se, da bo takrat na svetu le pet ali največ deset takih linij. Časi načrtovanja tako kompleksnih vezij se bodo stalno skrajševali in bodo leta 2010 dosegli čas le še treh dni za najbolj kompleksno vezje.

Mikroelektronika, ki temelji na znanih masnih efektih trdne snovi, bo dosegla svoj višek v začetku naslednjega tisočletja. To bo mikroelektronika s tranzistorji v izmeri 400 krat 400 atomov, kar je lahko najmanjši še funkcionalni tranzistor z masnim efektom. Novo področje znanosti bodo elementi z enim elektronom in balistični tranzistorji, katerih vedénje opisujejo zakoni kvantne fizike za posamezne delce (partikle). Ti elementi bi lahko do leta 2010 izpodrinili konvencionalne naprave, vendar to ni povsem zanesljivo.

V zadnjem času se pojavlja prepričanje, da bo elektronika dozorela in da bo t.i. *fotonika* prevzela elektronske funkcije. Viden je že začetek trenda, ki omogoča krmiljenje svetlobe s svetlobo, kar je bistveno pri fotonskih logičnih funkcijah. Do leta 2010 se bo ta trend že dobro razvil in fotonske logične funkcije naj bi tako dosegle svojo polno veljavo.

Naprave, kot je t.i. SEED (Self-Electro-optic Effect Device), omogočajo krmiljenje svetlobe s svetlobo, čeprav le posredno. SEED namreč omogoča krmiljenje elektronov s svetlobo, ti elektroni pa krmilijo svetlobo. Vse to se dogaja na atomski ravni, v atomu. Ob tem se raziskujejo še druge možnosti fotonske logike, z novimi vlaknastimi logičnimi elementi, v katerih svetloba neposredno krmili svetlobo brez elektronskega posredovanja.

Dosežki na ravni zmogljivost-razsežnost

Kakšne so lahko posledice glavnih trendov fotonike? Zmogljivost prenosa svetlobnega vala merimo s produktom sistemskega števila bitov v časovni enoti in razdalje, ki jo signali lahko prepotujejo brez regeneracije oziroma ojačevanja. V zadnjem desetletju se je ta zmogljivost podvojevala vsako leto in se bo lahko podvojevala še naslednji dve desetletji, ko bodo dosežene znane fizikalne meje.

Čeprav je fotonska funkcionalnost transporta že dobro razvita pa kažejo podatki o trendih ne le napredovanje v naslednjih 20 letih, temveč tudi, da znaša zmogljivost današnjih najnaprednejših razvojnih sistemov le 0,1 % znanih fizikalnih omejitev tehnologije svetlobnih valov. Praktični sistemi, ki danes obratujejo, že prenašajo informacijo s hitrostjo 3,4 milijarde bitov na sekundo, kar ustreza hkratnemu prenosu 50000 telefonskih pozivov v enemu vlaknenem paru. Tisočkratna izboljšava bi tako pomenila bilijarde bitov na sekundo, to pa je zmogljivost, ki bo leta 2010 potrebna zaradi masivnih količinskih zahtev visoke kakovosti, neokrnjenega video signala in trirazsežnostnih slik.

Vlakno do končnega uporabnika, do njegovega domačega in službenega mesta, bo poslednja stopnja pri dovršitvi vseobsegajoče vlaknaste komunikacijske mreže. Ta razvojna stopnja bo utemeljena s potrebo, ki bo omogočala uporabo video storitev končnim uporabnikom, z uvedbo novega digitalnega formata, ki se imenuje širokopasovna integrirana storitvena digitalna mreža (angl. Broadband Integrated Services Digital Network, kratko BISDN).

V tem trenutku se kaže, da bo rast rezidentnih vlaknastih mrež zelo hitra. V 15 letih lahko vlaknasta mreža naraste na 100 milijonov dostopnih linij. Ta rast je odvisna od pospeševanja novih video storitev, ki bodo dodane telefonskim storitvam. Brez teh novih storitev bi se rast vlaknene tehnologije lahko zakasnila za celo desetletje. Tehnologija fotonskega prenosa postaja tudi neodvisna od specifičnih prenosnih hitrosti. To je posledica uporabe optičnih ojačevalnikov, ki krmilijo svetlobo s svetlobo, tako da ni več potrebe po elektroniki visoke hitrosti, s katero se regenerirajo signali v elektronskih relejnih postajah. In en sam ojačevalnik obvladuje signale velikega števila nosilnih frekvenc in svetlobnih »barv«. To pa je pomembno zlasti v sistemih, ki uporabljajo multipleksiranje valovnega deljenja pri povečevanju zmogljivosti, ko se različne svetlobne »barve« prenašajo po istem vlaknu in kjer je vsaka barva nosilec neodvisnega informacijskega toka.

Optični ojačevalnik bo najprej uporabljen v prekoceanskem kabelskem sistemu, v katerem bodo signali prečili oceane brez regeneracije. Ta sistem je v razvoju in bo uporabljen še pred koncem tega desetletja. Fotonske prenosne značilnosti bodo utemeljene z nastajajočim mednarodnim standardom SONET (Synchronous Optical Network). Ker SONET opredeljuje standarne mrežne povezave, bodo lahko telefonska podjetja in končni uporabniki imeli naprave različnih proizvajalcev brez problema njihove združljivosti. SONET bo omogočal zmogljivi prenos širokopasovnih storitev in bo poenostavil mreže.

Komponente govornega procesiranja

Oglejmo si nekatere trende nastajajoče informacijske tehnologije za procesiranje govora. Stopnja integracije silicijskih vezij za procesiranje govora se povečuje s 33% letno. Pri tem narašča hitrost posameznih komponent za 20% na leto. Ti trendi omogočajo uporabo čedalje manjšega števila komponent pri procesiranju specifičnega govornega razpoznavanja in sinteze. To zmanjševanje materialne kompleksnosti sovpada z izboljševanjem algoritmov za procesiranje govora. Konec letošnjega leta bo že mogoče zgraditi 100-besedni razpoznavnik v enem vezju, medtem ko je bilo še pred petimi leti potrebnih 20 do 30 čipov. Napredni algoritmi povečujejo funkcionalnost govornega procesiranja, in sicer od sintezatorjev do 100- ali celo 1000-besednih razpoznavnikov na enem vezju, skladno z dozorevanjem elektronske tehnologije.

Na temelju teh trendov je že mogoče napovedati od govorca neodvisno avtomatično govorno razpoznavanje in sintezo. To bo imelo za posledico vrsto storitev, podprtih z inteligentnimi stroji, ki bodo govorili in poslušali kot ljudje. V letu 2010 naj naj bi bilo mogoče prenašati edinstvene osebne govorne značilnosti v različnih jezikih, z uporabo prirejenih govornih prevajalnikov. Govor iz enega jezika se bo

avtomatično in v realnem času prevajal v drugi jezik in bo lahko sinteziran (»poumetljen«) z glasovnimi značilnostmi izvirnega govorca. Tudi ta prevod bo potekal v realnem času.

Rast video storitev

Potrebe po razvedrilu naraščajo in razvija se pomembno tržišče vlaknastih mrež, v katerih so tudi vlakna do končnih uporabnikov. Razvedrilo je tako vir prihodkov, ki je potreben za pokritje stroškov naraščajoče fotonske mreže. Pri tem se zajemajo tudi podatki o informaciji, ki naj bi se pošiljala po mreži. Visokodefinicijska televizija (High-definition TV, kratko HDTV) potrebuje podatkovno zmogljivost 40 do 150 megabitov na sekundo in je tudi glavna privlačnost za razširitev širokopasovne mreže v domove.

Nekateri trendi kažejo, da bo zabavni video postal interaktiven in dosegljiv na zahtevo. Uporabnik bo lahko izbiral med velikim številom video programov preko seznama za izbiranje in to v poljubnem času dneva in noči, ko bo užival dani program od začetka do konca. Nekateri filmi in programi v živo pa bodo prirejeni za interakcijo z gledalcem. Ti tehnološki trendi kažejo, da bo postala »virtualna realnost« glavni hit okoli leta 2010 in bo temeljila na široko razpredeni telekomunikacijski in razvedrilni mreži. To bo omogočalo ljudem občutenje prostora in dogodja ter tudi izbranega dogodka v vseh razsežnostih posredno in na daljavo. Ljudje bodo lahko navzoči na sestankih brez potovanja, čeprav seveda telenavzočnost ne bo odpravila vsakega potovanja. Nekdo bo lahko sedel za svojo mizo in spremljal hkrati več sestankov na oddaljenih lokacijah, z možnostjo izbiranja svoje navzočnosti zdaj na tem, zdaj na onem sestanku.

Virtualna realnost (navidezna stvarnost) bo omogočala »izbiro« novega doma ali počitniške lokacije brez potovanja od doma. Posameznik bo lahko zgolj z zasukom glave videl in užival različne vidike partikularne scene, občutil pa bo lahko tudi teksturo (zgrajenost) zidne tapete od daleč, z uporabo elektronske rokavice.

Zmogljivost računalniškega procesiranja

Mikroračunalniški sistemi podvojijo svojo procesno zmogljivost vsako leto, veliki računalniki (mainframes) pa vsako tretje leto. Procesna zmogljivost se meri z enoto MIPS (milijon ukazov na sekundo). Mikroračunalniški sistemi uporabljajo čedalje več procesorjev. Npr. sistem AT&T 5ESS Switch je povečal število mikroprocesorjev z 2500 v letu 1987 na 5000 v letu 1991. Podjetje Thinking Machines prodaja superračunalnike z masivnim paralelizmom, kjer bo v paralelni sistem povezanih do 64000 visokozmogljivih, 64-bitnih mikroprocesorjev tipa RISC. Zmogljivosti teh sistemov se bodo merile v gigaflopsih. Pred dvajsetimi leti so ljudje reševali svoje probleme z velikimi računalniki v računalniških centrih. Danes spremljajo mikroračunalniki človeka povsod, kjer se ustrezni problemi pojavljajo. Po dvajsetih letih od danes bo človek še vedno odvisen od mikroračunalnika, ki bo elektronski ali fotonski in povezan v zmogljivo mrežo.

Računalništvo bo odvisno od koncepta mikroračunalniških skupin (clusters), s katerimi je že danes mogoče doseči 10000 MIPS. Pred dvajsetimi leti so veliki računalniki s težavo dosegli en MIPS.

Mrežna arhitektura

Trendi v smeri zmogljivosti procesiranja na temelju porazdeljenesti mikroprocesorjev (masivni paralelizem) so povzročili naraščanje sistemske inteligence na področjih umetne inteligence, telekomunikacij in informacijskih mrež. Prav mrežna inteligenca naj bi omogočila dosegljivost razpoložljivih mrežnih virov na zahtevo in to v širokem območju potreb. Mrežni uporabniki naj bi vstopali v sistem z natanko opredeljeno količino pasovne širine, kot jo za določene namene potrebujejo, npr. za sestankovanje na daljavo. Storitve na zahtevo bodo v mreži zagotovljene vnaprej; npr. zahteva za dovolj širok spekter zahtev bo vzpostavljena vnaprej. Ta mehanizem bo omogočal uporabniku in storitvenemu sistemu aktiviranje potreb na zahtevo, doma ali na delovnem mestu in v poljubnem času. Z visokozmogljivimi sistemi prihodnosti bo mogoče najti storitveno zmogljivost in vključevati mrežna vozlišča z uporabo softvera in na zahtevo.

Mrežna inteligenca bo nudila široko izbiro adaptivnih in logičnih storitev, ki bodo prilagojene potrebam uporabnikov in se bodo aktivirale s pro-

gramskimi ukazi v inteligentnih preklopnikih in drugih mrežnih elementih.

Mrežna arhitektura mora ustrezati vrsti multimedijskih aplikacij, kot so zvok, podatek in slika. Osnova teh sistemov bodo izpopolnjeni in široko razširjeni digitalni formati tipa ISDN, BISDN in SONET. In mrežne video aplikacije bodo pomembno vplivale na razvoj mrežne arhitekture.

Ozkopasovna ISDN bo dozorela v naslednjih dveh desetletjih. Z njo se bo spremenil način dela in igre. ISDN se le počasi uveljavlja — podobno kot se je telefon — ker je potrebno spremeniti navade, ki so v veljavi še danes. Današnje telefonske storitve bodo v glavnem zamenjane z ISDN. Visokokakovosten zvok bo le ena od mnogih možnosti ISDN. Visokokakovosten videotelefon bo druga možnost. Podatkovne baze za doma in delovno mesto bodo prispevale k kakovosti in učinkovitosti dela in življenja. Okoli leta 2010 bo širokopasovna ISDN že v intenzivni uporabi. Tako bo omogočena hitra obdelava distribuiranih podatkov in mrežno superračunalništvo. Močan je tudi trend uvajanja brezžičnih mrež. Vse več ljudi bo vzpostavljalo mobilne zveze prek osebnih komunikacijskih mrež (Personal Communication Network, kratko PCN).

(Se nadaljuje)

A.P. Železnikar

Pojasnilo k članku
Trends in Computer Progress

V časopisu Informatica, Vol. 16, No. 1, March 1992, je bil objavljen članek M. Gams, B. Hribovšek: Trends in Computer Progress. V poglavju 6 »PC world« in 7 »Summary« najdemo trditve, da s koprocesorji opremljeni najboljši PC računalniki že dosegajo okoli 40 MIPS. Zaradi morebitne nejasnosti podrobneje pojasnimo te trditve.

Po podatkih proizvajalcev dosegajo najboljši PC s koprocesorji (npr. Number Smasher 860) danes celo okoli 80 MFLOPS z enojno dolžino realnih števil. Brez specialnih koprocesorjev pa boljši današnji »standardni« PC (486) dosegajo le kakšnih 10 MIPS pri 33 ali 40 MHz. Ti »standardni« PC bodo morali preskočiti tehnološko mejo, da bodo po napovedih v obdobju 1995-2000 dosegli 100 MIPS.

Na koncu poudarimo, da je previdnost pri primerjanju vedno na mestu, saj se iz konteksta iztrgan stavek lahko narobe tolmači.

Matjaž Gams

Ustanovitev Slovenskega društva za umetno inteligenco

Ustanovna skupščina Slovenskega društva za umetno inteligenco (angl. Slovenian Artificial Intelligence Society, SLAIS) je bila na Institutu Jožef Stefan v Ljubljani, dne 11. 5. 1992. Na njej je bilo prisotnih 32 udeležencev, število članov pa že presega 50. Na ustanovni skupščini so bili sprejeti statut in člani Izvršnega odbora. Društvo bo intenzivno navezovalo stike s slovenskimi in mednarodnimi organizacijami in skrbelo za razširjanje področja umetne inteligence. Članarine ni, vseeno pa bodo člani deležni podobnih ugodnosti kot pri drugih društvih. Pristopna izjava in dodatne informacije so dostopne na naslovu: Laboratorij za umetno inteligenco, Institut »Jožef Stefan«, Jamova 39, Ljubljana.

K sodelovanju vabimo vse posameznike, ki jih kakorkoli zanima področje umetne inteligence in vsa slovenska in mednarodna društva in združenja.

Strokovna področja društva so:
- —metode za predstavitev znanja in avtomatsko sklepanje
- —avtomatsko učenje
- —avtomatizirano modeliranje
- —nevronske mreže
- —odločanje z metodami umetne inteligence
- —metode kombinatorične optimizacije
- —logično programiranje
- —kvalitativno modeliranje
- —genetski algoritmi
- —simbolično programiranje
- —ekspertni sistemi ni sistemi znanja, jeziki in orodja umetne inteligence, naravni jezik: jezikovne in govorne tehnologije, umetni vid

V izvršni odbor društva so bili izvoljeni:

Marko Bohanec, Ivan Bratko, Bojan Cestnik, Bogdan Filipič, Matjaž Gams, Nikola Guid, Anton Jezernik, Neda Karba, Igor Kononenko, Nada

Lavrač, Vladislav Rajkovič, Ivan Rozman, Peter Tancig in Jure Zupan.

Predsednik društva je Ivan Bratko, podpredsednik Matjaž Gams in tajnik Bogdan Filipič. Društvo se bo včlanilo v ECCAI.

R.A. de Beaugrande in W.U. Dressler: *Uvod v besediloslovje,* Park, Ljubljana 1992.

Naslova originalov, ki sta izšla sočasno, tj. »Einführung in die Textlinguistik« in "Introduction to Text Linguistics" pomenita uvod v lingvistiko besedila. Ta distinkcija je bistvena, ker lahko besediloslovje pomeni tudi še kaj drugega kot zgolj lingvistiko. Vsebino dela in to distinkcijo bomo pojasnili v nadaljevanju.

Knjigo sestavlja 10 poglavij ter seznami literature, kratic, imen in pomembnejših izrazov. V predgovoru dela najdemo kar nekaj zanimivih stališč, ki si jih velja zapomniti.

Faktorji, ki določajo dejavnosti v okviru »normalne znanosti«, so prej konvencije med znanstveniki kot pa očitne značilnosti raziskovanih predmetov. V modelih in teorijah se ni moč izogniti poenostavitvam in idealizacijam, ki so začasne in nezaželene. Formalizacija je le predstavitev in ne razlaga, sredstvo in ne cilj. Analizi formalnih struktur se vselej ne posreči odkritje narave in funkcije predmeta v širšem kontekstu. Jezikoslovje teži k naravoslovni, logični in matematični strogosti. Verjetnostni modeli so primernejši od deterministčnih. Dinamične strukture (zakonitosti, strategije, motivacije, preference, primeri) so plodnejše od statičnih opisov (pravil, zakonov). Znanost sistemizira nejasnosti svojih predmetov in jih ne ignorira. To delo načrtuje modele in pojmovne okvire za splošno spoznavno in komunikacijsko vedo.

Prvo poglavje (temeljni pojmi) uvaja verbalno in brez posebne pomenske globine sedem kriterijev besedilnosti: *kohezijo, koherenco, namernost, sprejemljivost, informativnost, situacijskost in medbesedilnost;* k tem doda še tri konstitutivna in regulativna načela: *učinkovitost, vtisnost in ustreznost.*

Drugo poglavje opisuje razvoj besediloslovja: retoriko, stilistiko, proučevanje literature, antropologijo, tagmeniko, sociologijo in analizo diskurza, členitev po aktualnosti, opisno strukturalno jezikoslovje, transformacijsko slovnico, Petöfijevo teorijo, Dijkove besedilne slovnice in Mel'čukov model smisel-besedilo.

Tretje poglavje (proceduralni pristop) obravnava med drugim pragmatiko, virtualne in aktualne sisteme, kibernetično regulacijo in faze oblikovanja besedila: *načrtovanje, ideacijo, razvijanje, izražanje, slovnično sintezo, linearizacijo in sosedstvo;* faze sprejemanja besedila: *slovnična analiza, priklicevanje pojmov in načrtov;* modele: *umetna inteligenca, kognitivna psihologija in tipi operacij.*

Četrto poglavje je namenjeno koheziji, ki upošteva funkcijo sintakse, površino in spomin ter tesno povezane vzorce: *fraza, delni stavek in stavek.* Nadalje obravnava ponavljano uporabo vzorcev, kot so rekurzivna pojavitev, paralelizem in parafraza; vzorce zgoščevanja: za-oblika, anafora, katafora, elipsa; odnose: dobiček in izguba med zgoščenostjo in jasnostjo; junkcijo: konjunkcija, disjunkcija, kontrajunkcija in subordinacija; in naposled še modalnost, perspektivo in intonacijo.

Osrednji motiv petega poglavja je koherenca: pomen versus smisel; nedoločnost, dvoumnost in večpomenskost; kontinuiteta smisla; besedilni svetovi; determinirajoče, tipično in naključno védenje; razstavljanje, proceduralna semantika, aktiviranje; epizodični in semantični spomin; okviri, sheme, načrti in scenariji; dedovanje; operatorji; model besedilnega sveta, inferiranje in referenca.

Šesto poglavje ima naslov »Namernost in sprejemljivost« in obravnava: namernost (intencionalnost), zmanjšana kohezija in koherenca; teorija govornih dejanj, prformativi; Griceove konverzacijske maksime: *sodelovanje, količina, relevantnost in način;* dejanje in diskurzno dejanje; načrti, cilji, scenariji; interaktivno načrtovanje (projektiranje), spremljanje, posredovanje; sprejemljivost in slovnična pravilnost.

Informativnost (informativity) je téma sedmega poglavja in označuje le stopnjo novosti ali nepričakovanosti (protiinformiranja) neke informacije za sprejemnika. Obravnavajo se tile predmeti: pozornost (attention); klasična teorija

informacije; veriga Markova; stopnje informativnosti; diskontinuitete in diskrepance; motivacija, usmerjenost, jakost povezave; klasifikacija pričakovanj: *dejanski svet, dejstva in prepričanja, normalne urejevalne strategije; organiziranost jezika: površina, besedilne zvrsti; neposredni kontekst, zanikanje, določnost; časopisni članek in sonet, motivacija nepričakovanosti.*

V okviru situacijskosti v osmem poglavju so opisani: modeli situacij, posredovanje in očitna dejstva, spremljanje vs. usmerjanje, dominance, opažanje, normalne urejevalne strategije, pogostnost, vpadljivost, eksofora itd., in naposled strategije za spremljanje in usmerjanje situacije.

Medbesedilnost v devetem poglavju označuje odvisnost tvorjenja oziroma sprejemanja besedila v odvisnosti od seznanjenosti udeležencev v komunikaciji z drugimi besedili. Tu imamo besedilne zvrsti vs. jezikovna tipologija; funkcionalne definicije: *deskriptivna, narativna in argumentativna besedila;* literarna in poetična besedila; znanstvena in didaktična besedila; organiziranost konverzacije; problemi in spremenljivke; modeli diskurznega sveta; obnavljanje besedilne vsebine; abstrahiranje sledov, konstruiranje in rekonstruiranje; mentalne slike in prizori; interakcija med védenjem, predstavljenim v besedilu in shranjenim védenjem o svetu; besedilnost in eksperimenti v zvezi s spominjanjem.

Zadnje, deseto poglavje nosi naslov »Raziskovanje in interdisciplinarna aplikacija«. V tem okviru so npr. kognitivna znanost; spretnosti racionalnega človekovega obnašanja; jezik in spoznavanje; definiranje inteligence; besedila kot prenosniki znanosti; sociologija, antropologija, psihiatrija in svetovalna psihologija; pisanje, branje in berljivost; proučevanje literature: deavtomatizacija; generativna poetika; literarna kritika kot zniževanje vrednosti; poučevanje prevajanja: dobesedno in prosto prevajanje; kontrastivna lingvistika; semiotika, znanost o računalništvu in umetna inteligenca; razumevanje razumevanja.

Besediloslovje je le ena med disciplinami, ki so v okviru nastajanja in razumevanja besedila; besedilo je samo po sebi jezikovni (npr. literarni) artefakt (začetna, izhodiščna informacijska formula), ki v sprejemni entiteti sproži njeno razumevanje v obliki informacijskega sistema razumevanja. Na primeru informacijske formalizacije Heideggrovega besedila [IAB1, IAB2] smo spoznali, kako postanejo besede in besedne skupine informirajoče entitete in kako se začetna formula razvija, informacijsko nastaja, paralelizira, komponira in dekomponira, skratka narašča z informacijsko širitvijo razumevnega sveta.

Heideggrov sistem razumevanja skozi tu-bit je seveda splošen in ne velja le za besedilne primere. Ta sistem je v začetni obliki opisan kot transformacija iz literarne besedilne oblike v informacijsko formulsko obliko, ki je kot informacijska operandna entiteta informacijsko nastajajoča. Je torej lahko nekakšen model sprejemnika. Pri tem gre za dvojno umestitev (embedding). Najprej se sistem razumevanja umesti v sprejemni entiteti, potem pa ta vmeščeni sistem razumevanja začne razumevati dano besedilo skladno s strukturo in organizacijo (in seveda nastajanjem) informacijskega razumevnega sistema.

Prevod omenjene knjige v slovenščino je tako dober povod, da začnemo o kompleksnosti nekega sistema razumevanja — ne nujno skozi Heideggrovo bit — razmišljati in raziskovati tudi pri nas. Informacijski okvir je seveda osnovna stopnja formalizacije, v katero se vmeščajo konkretne sheme in scenariji razumevanja. Ob tem mora biti zagotovljeno informiranje nastopajočih entitet.

Literatura

[IAB1] Železnikar, A.P., An Informational Approach of Being-there as Understanding I, Informatica **16** (1992) *1*, 9-26.

[IAB2] Železnikar, A.P., An Informational Approach of Being-there as Understanding II, Informatica **16** (1992) *2*, 29-58.

Anton P. Železnikar

Informatica

Informatica *is published four times a year in Winter, Spring, Summer and Autumn by the Slovene Society Informatika, Tržaška cesta 2, 61000 Ljubljana, Slovenia.*

Volume 16 Number 2 June 1992 YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

CONTENTS