Volume 19 Number 3 September 1995 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Guest Editor: Johann Eder

Concurrent Software with Redundancy

Self-Learning Fuzzy Rules

Scheduling Parallel Tasks

Object-Oriented Database Language

Hot Living Systems

Informational Calculus

Informatica 19 (1995) Number 3, pp. 291–421

1977

The Slovene Society Informatika, Ljubljana, Slovenia

Informatica

An International Journal of Computing and Informatics

Basic info about Informatica and back issues may be FTP'd from ftp.arnes.si in magazines/informatica ID: anonymous PASSWORD: <your mail address>
FTP archive may be also accessed with WWW (worldwide web) clients with
URL: `http://www2.ijs.si/~mezi/informatica.html`

Subscription Information: Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 61000 Ljubljana, Slovenia.
The subscription rate for 1995 (Volume 19) is
- DEM 50 (US$ 35) for institutions,
- DEM 25 (US$ 17) for individuals, and
- DEM 10 (US$ 7) for students

plus the mail charge DEM 10 (US$ 7).
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LaTeX Tech. Support: Borut Žnidar, DALCOM d.o.o., Stegne 27, 61000 Ljubljana, Slovenia.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Žibertova 1, 61000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Jožef Stefan Institute: Tel (+386) 61 1773 900, Fax (+386) 61 219 385, or use the bank account number 900–27620–5159/4 Ljubljanska banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

According to the opinion of the Ministry for Informing (number 23/216-92 of March 27, 1992), the scientific journal Informatica is a product of informative matter (point 13 of the tariff number 3), for which the tax of traffic amounts to 5%.

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences (Janez Peklenik)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Cur. Cont. & Comp. & Math. Sear., Engineering Index, INSPEC, Mathematical Reviews, Sociological Abstracts, Uncover, Zentralblatt für Mathematik und ihre Grenzgebiete.

The issuing of the Informatica journal is financially supported by the Ministry for Science and Technology, Slovenska 50, 61000 Ljubljana, Slovenia.

Reliability Optimization of Concurrent Software with Redundancy

Jie Wu and Kejun Yao
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

Keywords: concurrent systems, fault tolerance, fault-tree, software reliability

Edited by: Johann Eder
Received: February 22, 1995 **Revised:** July 30, 1995 **Accepted:** September 7, 1995

We consider in this paper an optimization model of concurrent software with redundancy. The software fault-tolerance technique used is conversation, a special mechanism for concurrent systems. Two optimization models are combinatorial model and fault-tree model which calculate system reliability based on the same method but under different fault assumptions. These two models are illustrated by examples.

1 Introduction

With the increasing complexity of computer systems and importance of performing intended functions in critical tasks, such as nuclear plant control, defense systems, and automated factories, the *dependability* of these systems becomes essential. *Dependability* [8] is defined as the ability of a system or product to deliver its intended level of services to its users, especially in the lights of failures or other incidents that impinge on its level of serve. Dependability is a high-level framework that encompasses several measures including reliability. *Fault prevention* and *fault tolerance* are two complementing strategies for achieving reliability. A fault-tolerant system is one that can continue to correctly perform its specified tasks in the presence of hardware or software faults.

Fault tolerance is usually achieved by means of redundancy [6]. Hardware fault tolerance and software fault tolerance in sequential systems have been extensively studied, however, software fault tolerance in concurrent systems has not been exploited much. The *conversation* [9] is a special software fault tolerance construct for concurrent systems. A conversation involves two or more processes, as it constitutes a boundary to prevent possible error propagation to eliminate the *domino effect* [9]. Identification of a set of conversations in a given systems has been discussed by Wu and Fernandez [11] using Petri nets and by Tyrrell and Carpenter [10] using CSP. The potential practical value was shown in a 3-year experiment conducted at the University of Newcastle during 1982-1984. However, there are still important practical problems that remain to be solved. For example, under a given cost, how to construct one or several conversation to achieve a maximum reliability?

The optimal structure for software reliability models was initially proposed by Belli and Jedrzejowice [2], which involves application of the existing reliability predictions, evaluation of the amount of resources needed to develop and implement a system, and calculation of the optimal sequential software structure. This scheme was extended later in Zhang's reliability model [12]. However, the software fault tolerance schemes considered in Belli's model and Zhang's model are for sequential systems only. Software fault tolerance schemes for concurrent systems, such as the one using conversation, have not been studied for optimization.

In this paper, we extend the optimization model proposed by Belli and Jedrzejowice by considering the following problem: Select a set of software versions to construct one or multiple conversation such that the system has a maximum reliability while the total system cost is bounded by a given value. Normally, different reliability results are obtained by using different fault assumptions. We study the *combinatorial model* and the *fault-*

tree model which are based on the same method but under different fault assumptions.

The paper is organized as following: Section 2 describes the mechanism of conversation. Section 3 studies the combinatorial model. Section 4 proposes the fault-tree model. Section 5 concludes the paper.

2 Conversation

The conversation is a language construct which is a generalization of the *recovery block scheme* [5] in a concurrent system. A conversation defines an *atomic action* for a set of communicating processes. Two conversations are either nested or independent. The boundaries of a conversation consist of a *recovery line*, a *test line* and two *side walls*. A recovery line is a set of recovery points which are established before any process communication. A test line is the correlated set of the acceptance tests of the interacting processes. Two *side walls* prevent the processes within a conversation from interacting with processes outside the conversation. A conversation is successful only if all its processes pass their acceptance tests at the test line. If any acceptance test fails, all the processes within the conversation go back to the recovery line, restore the previous state by replacing necessary variables with the ones saved in the recovery points, and retry using their alternate try versions. The structure of conversation is shown in Figure 1, where it is assumed that there are p_j parallel processes in one conversation that have the same number of versions. A version, a sequence of instructions, consists of two parts. The first part is a sequence of instructions called the *function segment*. The second part is the *test segment*. The role of the test segment is to test the result of various versions in sequence and to accept (or reject) them. The test segment is also subject to failures. The failure of the test segment means that it either judges the correct result of a version as incorrect or judges the incorrect result of a version correct. If the computation result in the ith version of any process is rejected then every process's initial state is restored, and its $(i+1)$th version is activated and performed.

A fault-tolerant concurrent system consists of a set of *special objects* which are conversations and a set of regular objects where no fault-tolerance mechanism is applied. Since the correct execution of the whole system depends on the correct execution of each object, the reliability model of the system could be defined as objects connected in series.

Basically, there are four different types of errors that can occur in a conversation structure [7]. A Type 1 error occurs when any process version produces an incorrect result, but the acceptance test labels the result as correct. As a Type 2 error, the final version produces a correct result, but the acceptance test erroneously determines that the result as incorrect. A Type 3 error occurs when the recovery mechanism can not successfully recover the input state of the previous version in preparation for executing another version, or it can not successfully execute the next version. Finally, a Type 4 error occurs when the last version produces an incorrect result even though the test segment correctly judges it. In our models, we don't consider Type 3 error. We assume that the recovery mechanism can recover the previous state successfully all the time. We focus on finding a set of versions such that the system has the maximum reliability under a given cost.

Different reliability results can be obtained by using different fault assumptions. Here, we study two models which are based on the same method but under two different fault assumptions. The first model is a *combinatorial model* which considers only Type 2 and Type 4 errors. The second model is a *fault-tree model* which concerns with only Type 1, Type 2 and Type 4 errors.

The combinatorial reliability model used in this paper is based on the following assumptions:

1. All software failures are independent.
2. The total system cost is equal to the sum of the costs of all the objects.
3. All the software version reliabilities and costs are known.
4. Reliabilities of test segments are known as the probability that the error is detected if it occurs.

The fault-tree model also use the above assumptions 1, 2 and 3. But the assumption 4 is different:

4′. Failure probability of each test segment has two parts: the probability that the test segment

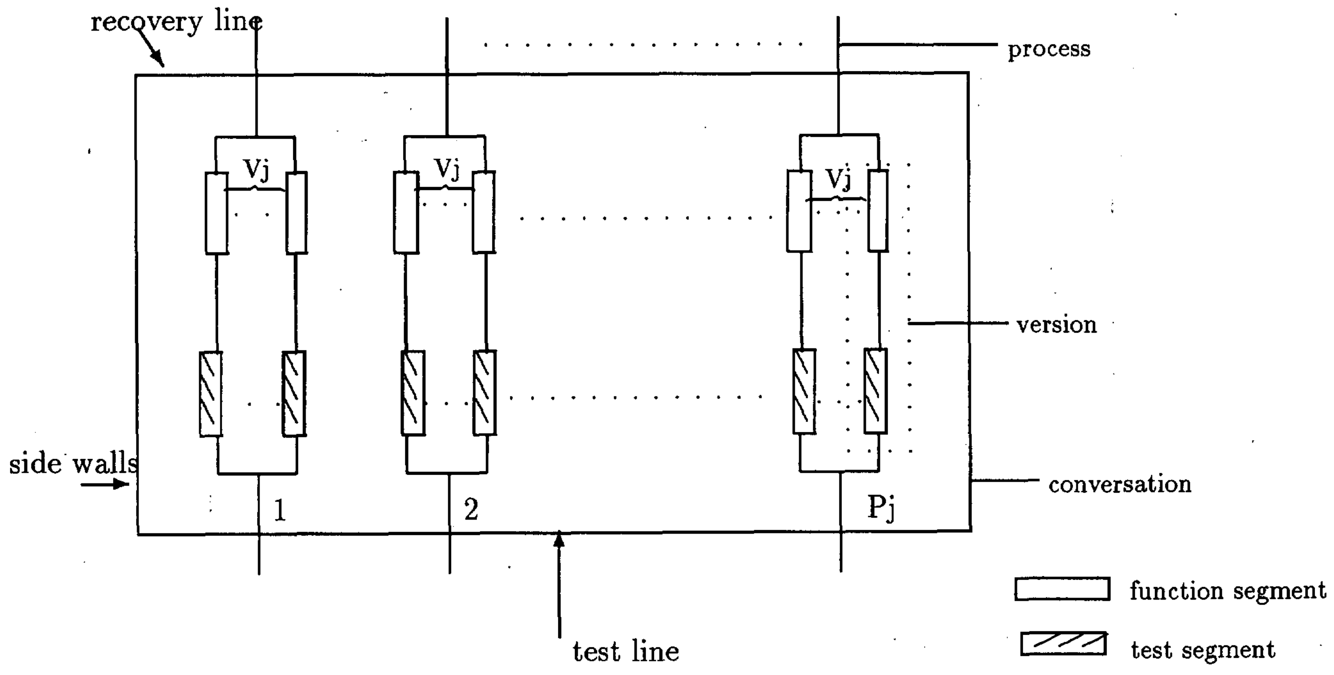


Figure 1: Structure of a conversation

judges a correct result as incorrect and the probability that it judges an incorrect result as correct.

The following notation is introduced:

q	Number of independent conversations (j is used as index for conversation)
v_j	Number of software versions in the jth conversation (i is used as index for version)
p_j	Number of processes in the jth conversation (k is used as index for process)
C_{ij}^k	Amount of cost needed for the ith version in the k process of the jth conversation
R_j	Reliability of the jth conversation
R_s	System reliability
C	Specified cost

3 Combinatorial Model

The following is notation used in the combinatorial model:

r_{ji}^k	Reliability of the function segment of the ith version in the kth process of the jth conversation
t_{ji}^k	Reliability of the test segment of the ith version in the kth process of the jth conversation (probability that an error is detected when it occurs)

Using the combinatorial model, we can derive the reliability of the jth conversation as follows:

$$R_j = \sum_{i=1}^{v_j} p_{ji} r_{ji} t_{ji} \tag{1}$$

$$r_{ji} = \prod_{k=1}^{p_j} r_{ji}^k \tag{2}$$

$$t_{ji} = \prod_{k=1}^{p_j} t_{ji}^k \tag{3}$$

$$p_{j(i+1)} = p_{ji}(1 - \sum_{k=1}^{p_j} r_{ji}^k t_{ji}^k) \tag{4}$$

where $p_{j1} = 1$, p_{ji} denotes the probability that the ith path in the jth conversation is executed. There is only one possible condition under which

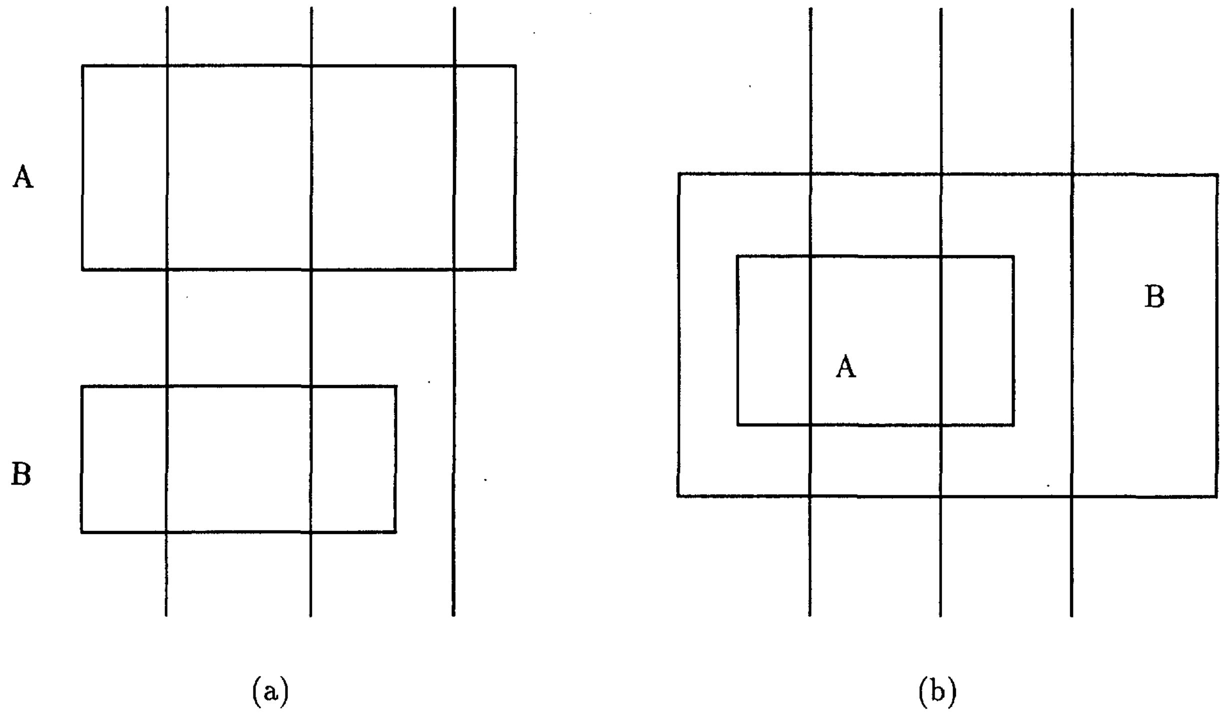


Figure 2: A pair of conversations, (a) independent, (b) nested

the $(i+1)$th version will not be activated. That happens when all the function segments in the ith versions of all the processes set their correct results, and test segments judge them as correct. (To simplify the calculation, we don't consider here the Type 1 fault condition which has a relative small possibility: the function segment fails and the test segment judges the incorrect result as correct.) r_{ji} is the reliability of the ith version of all the processes in the jth conversation, and t_{ji} is the possibility that all the test segments in the ith version of all the processes of the jth conversation judge their correct results as correct.

Usually, there are more than one conversation in a fault-tolerant system. In these case, for any pair of conversations, they are either *independent* or *nested.* As in Figure 2, conversations A and B are independent in Figure 2(a), while A is nested within B in Figure 2(b).

First, we examine systems with only independent conversations. In such a system, the system structure can be partitioned into a set of special objects (conversations) together with a set of regular objects (without using conversations). Each regular object can also be considered as a special conversation with one software version for all the participating processes. Therefore, the reliability model of a system with q conversations is equivalent to the one for serial system of q objects. The system reliability R_s can be expressed as follows:

$$R_s = \prod_{j=1}^{q} R_j \tag{5}$$

R_j is the reliability of the jth object (special or regular), $1 \le j \le q$.

When there are nested conversations in the system, to simplify the calculation, we first consider a case in which one conversation is nested within another conversation. In Figure 2(b), conversation A is nested within conversation B. Obviously, if conversation A fails, then the conversation B also fails (if conversation A fails, as part of con-

j	1	2
p_j	3	2
r^k_{ji}	0.85 0.85 0.85	0.88 0.88
c^k_{ji}	10 10 10	12 12
v^k_{ji}	0.8 0.8 0.8	0.7 0.7

Table 1: Failure probabilities and costs for a system with two independent conversations

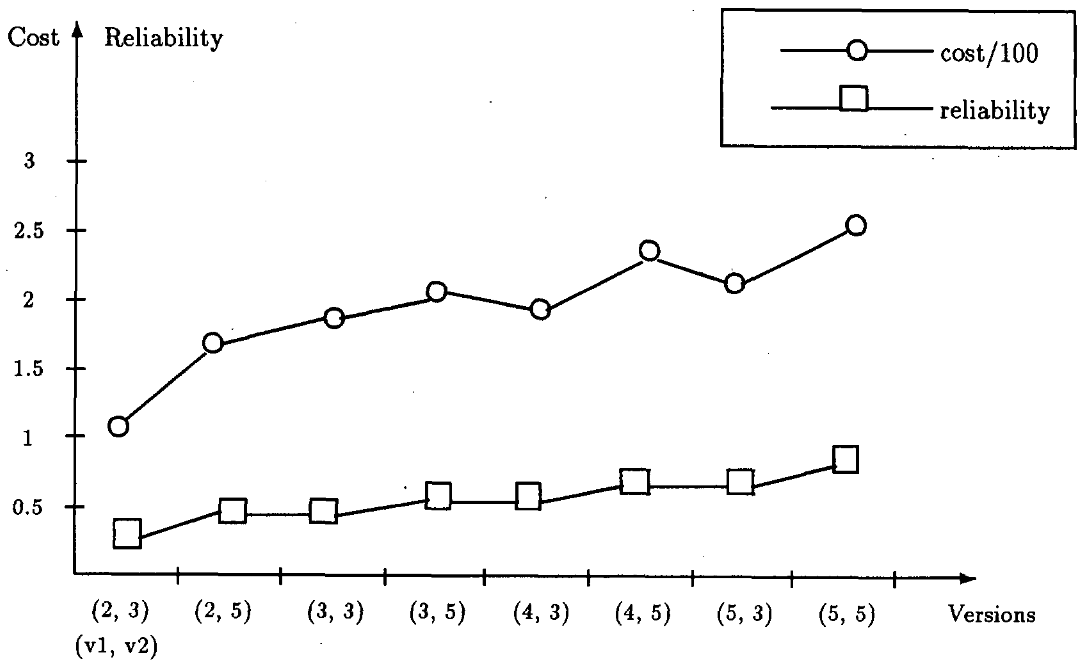


Figure 3: Costs/reliabilities of the combinatorial model

versation A, conversation B never gets the correct result). So the system reliability for this case is:

$$R_s = R_A R_{B'} \tag{6}$$

where R_A is the reliability of conversation A, and $R_{B'}$ is the reliability of the part included in conversation B but is excluded in conversation A. Equations(1)-(5) are still suitable in the reliability calculation of nested conversations; however, r_{ji}^k and t_{ji}^k in conversation B should be changed to $r_{ji}'^k$ and $t_{ji}'^k$ in calculating $R_{B'}$, which are:

$r_{ji}'^k$	Reliability of the function segment (which is not included in conversation A) in the ith version of the kth process in the jth conversation.
$t_{ji}'^k$	Reliability of the test segment (which is excluded in conversation A) in the ith version of the kth process in the jth conversation.

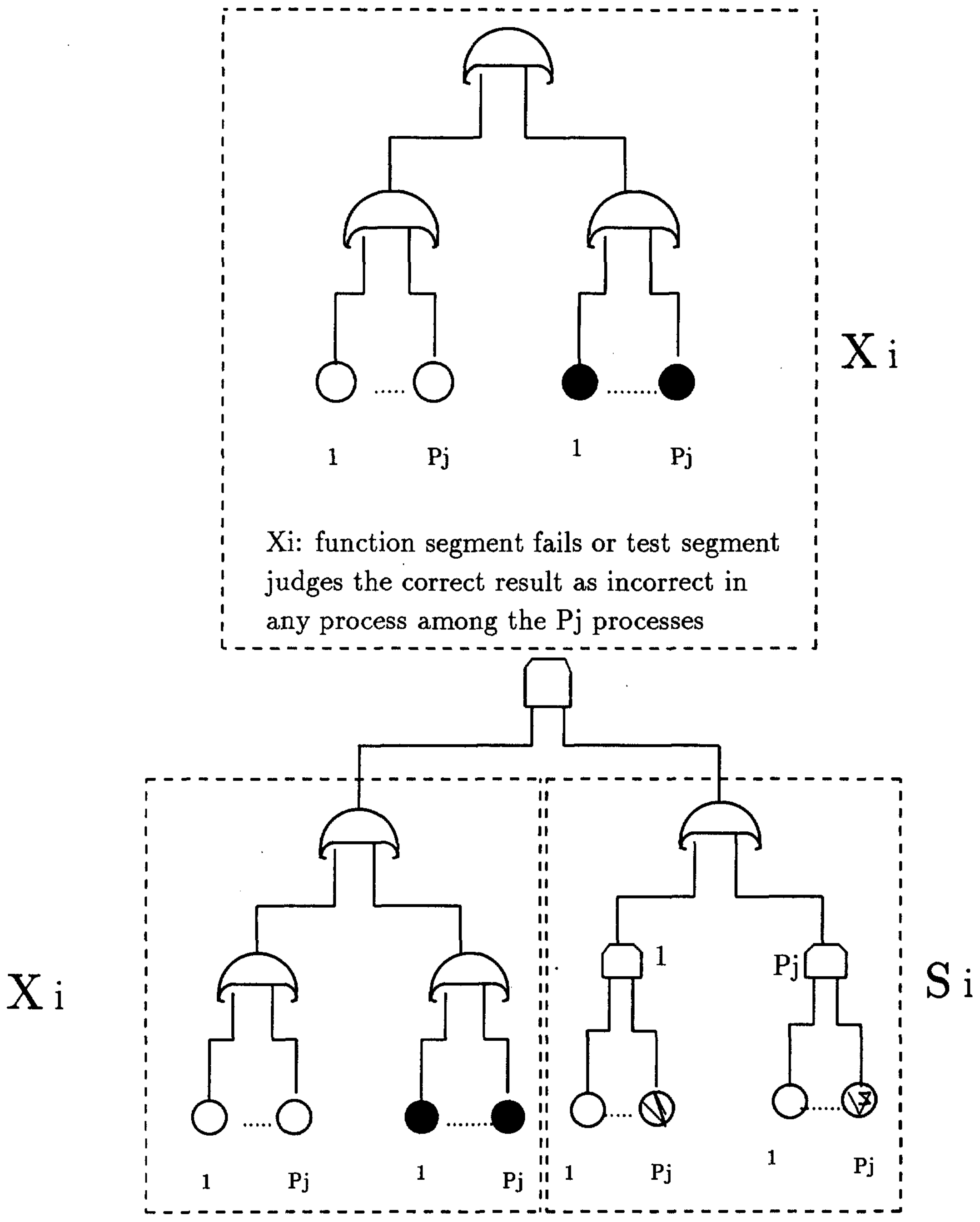
Hence, the software optimization problem becomes:

$$\max_{v_i} \prod_{j=1}^{q} \sum_{i=1}^{v_j} p_{ji} r_{ji} \tag{7}$$

subject to:

$$\sum_{j=1}^{q} \sum_{i=1}^{v_j} c_{ji} \le C \tag{8}$$

The optimal solution is obtained by solving the *nonlinear programming problem* defined in (7) and (8). However, there is a heuristic method [13] that solves this problem. As an example, we consider a system consists of two conversations. Component reliabilities and their costs are given in Table 1. For the sake of simplicity, we assume that reliabilities of function segments and testing segments of different processes in each conversation are identical. Also the cost of each path in the same conversation is identical. The results of different version combinations from (2, 2) (the first number represents the number of versions in the first conversation, the second number represents the number of versions in the second conversation) to (5, 5) are obtained from expression (7) and are shown in Figure 3 and Table 2. Assume that the specified cost is 200, we can obtain the optimal result from those sets of versions (shown in Table 2) which have the cost below 200. Obviously, the optimal structure is: v_1=4, v_2=3, R_s=0.653 as shown in Figure 4.


1
Pj
1
Pj
Xi
Xi: function segment fails or test segment judges the correct result as incorrect in any process among the Pj processes
Xi
1
Pj
1
Pj
1
Pj
1
Pj
Si

Xi: the ith version fails which activates the (i+1)th version.
Si: in the (i+1)th version, the function segment fails and the test segment judgs the incorrect result as correct which causes the system failure.

Figure 5: Fault-tree diagram of two types of failures in the jth conversation

(v1, v2)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(3, 2)	(3, 3)	(3, 4)	(3, 5)
R_s	0.387	0.465	0.494	0.548	0.488	0.574	0.622	0.689
C	108	132	156	180	138	162	186	210
(v1, v2)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(5, 2)	(5, 3)	(5, 4)	(5, 5)
R_s	0.554	0.653	0.706	0.783	0.608	0.716	0.776	0.859
C	168	192	216	240	198	222	246	270

Table 2: R_s and C under different conversation configurations in the combinatorial model

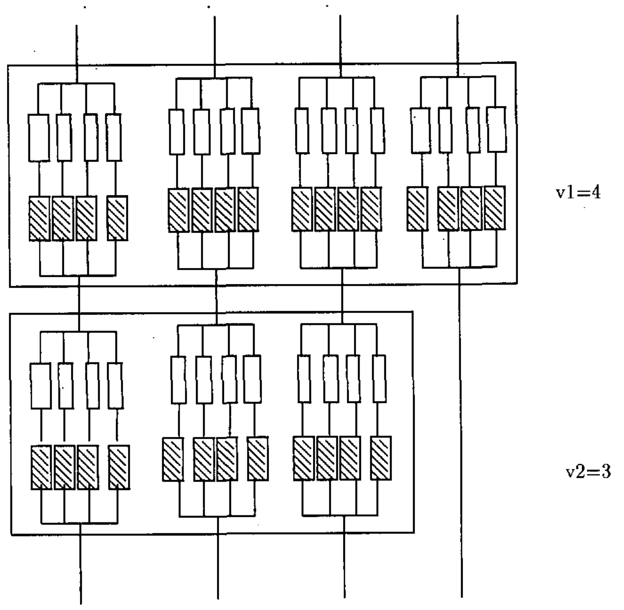


Figure 4: The optimal structure of the example in Table 1

4 Fault-Tree Model

The overall design goal of fault-tolerant software can be briefly stated as preparing for the unexpected. Unfortunately, even in principle, there is no way to anticipate all the problems that will be encountered. However, in designing fault-tolerant software, one can have a conception of the general classes of failures that can occur. Review of the fault-tolerant aspects of the software is facilitated by the use of the trees: failures that are covered are explicitly listed; uncovered failures do not appear. *Fault-tree* [4] is such a model used as means of identifying these classes of faults in a top-down fashion. The basic notation used in a fault-tree is listed in Figure 5. The idea is that a top event is broken into several low (or simple) events that are characterized as a "yes/no" occurrences. These low events are linked together by AND or OR gates. System probability can be calculated by combining probabilities of the lower events based on the types of linkage: addition in the case of an OR gate; multiplication in the case of an AND gate. Therefore, conceptually the combinatorial and the fault-tree models are the same. To facilitate the discussion, we illustrate the fault-tree model by including the Type 1 error.

Let the following symbols denote probabilities of the basic event:

e^k_{ji}	Probability that the functioning segment in the ith version of the kth process in the jth conversation has failed
m^k_{ji}	Probability that the test segment in the ith version process of the kth process in the jth conversation judges the incorrect results as incorrect
n^k_{ji}	Probability that the test segment in the ith version of the kth process in the jth conversation judges the incorrect results as correct
F_j	Failure probability of the jth conversation

The parameters such as j, v_j, i, p_j, k are the same as in the combinatorial model. There are two types of failure which will cause the system failure. The first one is that the function segment fails or the test segment judges the correct results as incorrect in each version. The illustration of this failure in fault-tree model is shown in Figure 5(a) named as X_i. The other one is that the function segment or test segment of the ith version fails which activates the $(i+1)$th version, then the function segment of the $(i+1)$th still fails, and the test segment of the $(i+1)$th version judges the incorrect result as correct. The fault-tree diagram of the $(i+1)$th failure is shown in Figure 5(b) named as S_i. Therefore, we can easily

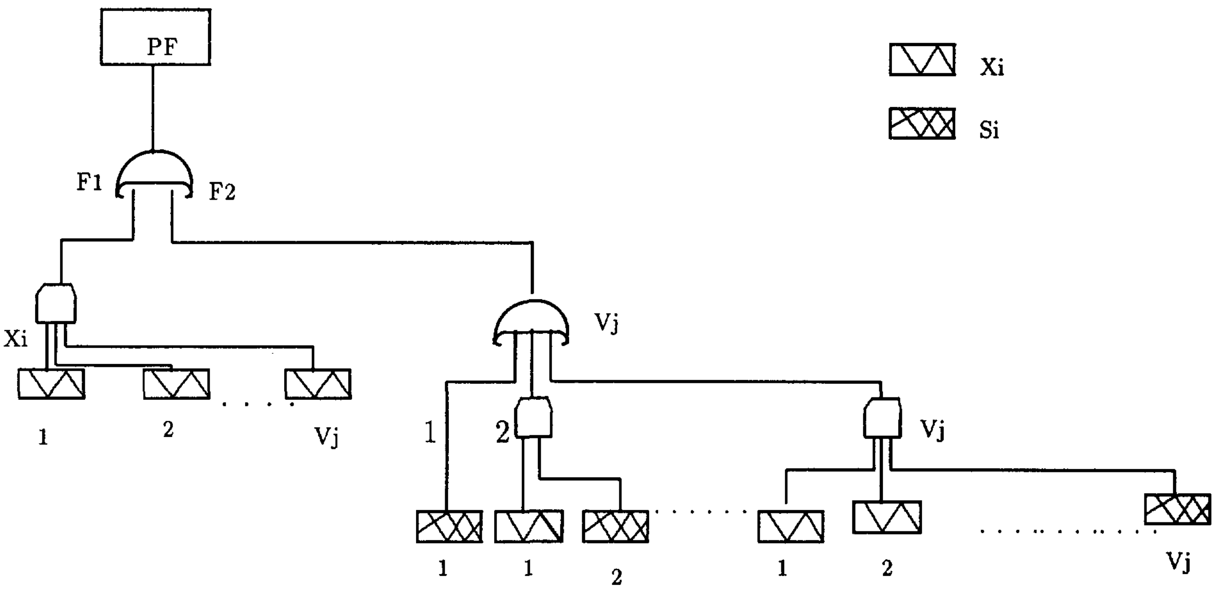


Figure 6: The fault-tree diagram of the v_j-version conversation

j	1	2
p_j	3	2
r^k_{ji}	0.15 0.15 0.15	0.12 0.12
c^k_{ji}	10 10 10	12 12
m^k_{ji}	0.18 0.18 0.18	0.22 0.22
n^k_{ji}	0.02 0.02 0.02	0.08 0.08

Table 3: Failure probabilities and costs for a system with independent conversations

obtain the fault-tree diagram of the v_jth version conversation as shown in Figure 6.

The failure probability of the system consists of two parts. $F1$ is caused by the X_i failure, which represents that the function segment fails or the test segment judges the correct result as incorrect in each version. $F2$ is caused by the failure of S_i, which represents that the function segment fails and the test segment judges the incorrect result as correct and the next version is not activated.

$$F_j = F1 + F2 - F1 * F2 \tag{9}$$

$F1$ can be obtained by the following equations:

$$F1 = \prod_{i=1}^{v_j} \tag{10}$$

$$X_i = E_i + M_i - E_i * M_i \tag{11}$$

$$E_i = \sum_{k=1}^{p_j} e^k_{ji} - \sum_{k=1}^{p_j}\sum_{l>k}^{p_j}(e^k_{ji} \cap e^l_{ji}) + \ldots + (-1)^{p_j-1}\prod_{k=1}^{p_j} e^k_{ji} \tag{12}$$

$$M_i = \sum_{k=1}^{p_j} m^k_{ji} - \sum_{k=1}^{p_j}\sum_{l>k}^{p_j}(m^k_{ji} \cap m^l_{ji}) + \ldots + (-1)^{p_j-1}\prod_{k=1}^{p_j} m^k_{ji} \tag{13}$$

$F2$ can be obtained by the following equations:

$$F2 = \sum_{i=1}^{v_j} Y_i - \sum_{i=1}^{v_j}\sum_{l>i}^{v_j}(Y_i \cap Y_l) + \ldots + (-1)^{v_j-1}\prod_{i=1}^{v_j} Y_i \tag{14}$$

$$Y_1 = S_1 \tag{15}$$

$$Y_i = S_i \prod_{l=1}^{i-1} X_l (i > 1) \tag{16}$$

$$S_i = \sum_{k=1}^{p_j} Z_{jk} - \sum_{k=1}^{p_j}\sum_{l>k}^{p_j}(Z_{ik} \cap Z_{il} + \ldots + (-1)^{p_j-1}) \prod_{k=1}^{p_j} Z_{ik} \tag{17}$$

$$Z_{ik} = e^k_{ji} * n^k_{ji} \tag{18}$$

$$R_j = 1 - F_j \tag{19}$$

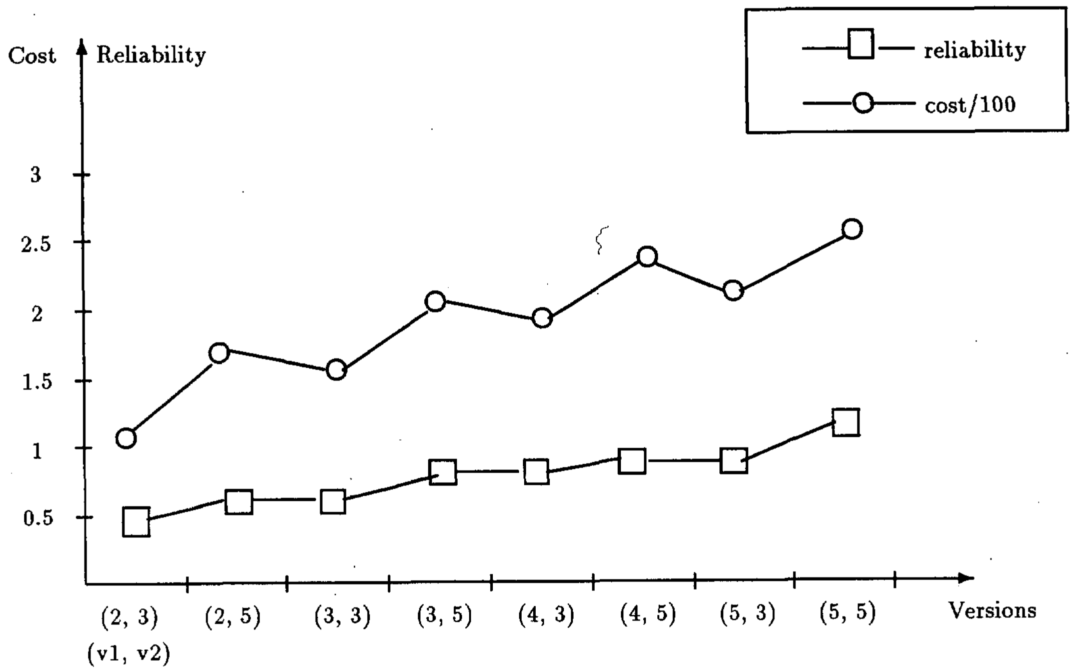


Figure 7: Costs/reliabilities of the fault-free model

(v1, v2)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(3, 2)	(3, 3)	(3, 4)	(3, 5)
R_s	0.424	0.493	0.531	0.603	0.521	0.606	0.652	0.741
C	108	132	156	180	138	162	186	210
(v1, v2)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(5, 2)	(5, 3)	(5, 4)	(5, 5)
R_s	0.585	0.680	0.731	0.831	0.674	0.801	0.863	0.980
C	168	192	216	240	198	222	246	270

Table 4: R_s and C under different conversation configurations in the fault-tree model

$$R_s = \prod_{j=1}^{q} R_j \tag{20}$$

Hence, the software optimization problem is:

$$\max_{v_j} \prod_{j=1}^{q} R_j \tag{21}$$

subject to:

$$\sum_{j=1}^{q} \sum_{i=1}^{v_j} C_{ji} \leq C \tag{22}$$

We use the same example as used in the combinatorial model and provide a set of parameters as shown in Table 3. According to the above equations, we can obtain reliabilities of different version combinations from (2, 2) to (5, 5) as shown in Table 4. As the specified cost is 200, we can get the optimal result: $v_1 = 4, v_2 = 3, R_s = 0.68$. The results are also shown in Figure 7.

5 Conclusion

In this paper, we have studied the reliability optimization problem in concurrent software with redundancy using heuristic methods. The models presented in this paper show that it is possible to formulate and solve a particular reliability optimization problem. The combinatorial model and the fault-tree model have been used, which are based on the same method, but are illustrated by using different fault assumptions. The combinatorial model is more convenient to calculate, while the fault-tree model can be clearly illustrated by the fault-tree diagram. The next steps in the development of reliability optimization models can lead toward incorporating the effect of correlated errors [3] on reliability models.

References

[1] Avizienis, A., The n-version approach to fault-tolerant software, *IEEE Transactions on Software Engineering*, Vol. 11, No. 12, Dec 1985, 1491-1501.

[2] Belli,F and P. Jedrzejowicz, An approach to the reliability optimization of software with redundancy, *IEEE Transactions on Software Engineering*, Vol.17, No. 3, Mar.1991, 310-312.

[3] Eckhardt, Jr., D. C. and L. D. Lee, A theoretical basis for the analysis of multiversion software subject to coincident errors, *IEEE Transactions on Software Engineering*, Vol. 11, No. 12, Dec. 1985, 1511-1517.

[4] Hecht, H. and M. Hecht, Fault-tolerant software. *Fault-tolerant Computing: Theory and Techniques*, Vol. 27, D.K. Pradham, ed. Prentice-Hall, 1986.

[5] Horning, J. J., H. C. Lauer, P. M. Melliar-Smith, and B. Randell, A program structure for error detection and recovery, *Proc. Conf. Operating System: Theoretical and Practical Aspects IRIA*, Apr. 1974, 177-193.

[6] Johnson, B.W., *Design and Analysis of Fault-Tolerant Digital Systems.* Addison-Wesley Publishing Company, Inc. 1987.

[7] Scott, K. R. and J. W. Gault, Fault-tolerant software reliability modeling, *IEEE Transactions on Software Engineering*, Vol. 13, No.5, May 1987.

[8] Laprie, J.C., Dependability: A unifying concept for reliable computing and fault-tolerance. *Dependability of Resilient Computers.* T. Anderson, ed, BSP Professional Books, 1989,1-28.

[9] Randell, B., System structure for fault tolerance. *IEEE Transactions on Software Engineering.* Vol. 1, No. 6, June, 1975, 220-232.

[10] Tyrrell, A. M. and G. F. Carpenter, CSP methods for identifying atomic actions in the design of fault tolerant concurrent systems, *IEEE Transactions on Software Engineering*, Vol. 21, No. 7, July 1995, 629-639.

[11] Wu, J. and E. B. Fernandez, Using Petri nets for the design of conversation boundaries in fault-tolerant software, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 10, Oct. 1994, 1106-1112.

[12] Wu, J., Zhang, M. and E. B. Fernandez, Design and Modeling of Hybrid Fault-Tolerant Software with Cost Constraints, accepted to appear in the *Journal of Systems and Software.*

[13] Zhang, M., Design and Modeling of Hybrid Fault-Tolerant Systems. *Master's Thesis.* Florida Atlantic University, Boca Raton, FL., 1992.

An Algorithm for Self-Learning and Self-Completing Fuzzy Control Rules

Xiaozhong Li and Shuo Bai
National Research Center for Intelligent Computing Systems(NCIC)
P. O. Box 2704, Beijing
Beijing 100080. P. R. C
Fax: 86-10-2541342, Tel:86-10-2534642
email:{lxz, bai}@tango.ncic.ac.cn
AND
Zemin Liu
Beijing University of Posts & Telecommunications

Keywords: self-learning, self-completing, fuzzy control

Edited by: Johann Eder
Received: December 12, 1994 **Revised:** June 30, 1995 **Accepted:** September 8, 1995

In this paper, we introduce a method of generating control rules for fuzzy logic controllers, called an algorithm of self-learning and self-completing (ASLSC). This method can learn to generate rules from the sample data with the unsupervised competitive learning (UCL) of the neural network. If a rule table generated is not full, the ASLSC method can fill those blanks in the rule table with proper rules. Besides, ASLSC can determine automatically the universes of discourse of fuzzy variables. The simulations on an inverted pendulum system and the experiments on temperature controlling of an industrial electric heating ring show that ASLSC is practicable and effective.

1 Introduction

Since E. H. Mamdani and S. Assilian first implemented a fuzzy controller in 1974 [5], fuzzy control technology has been widely used and developed because of its strong robustness and insensibility to environmental parameters and short settling time. However, the fuzzy control theory itself is not perfect, yet, and many aspects rely on the experience of an expert or a skilled operator, for example, (1) selection of a universe of discourse or an interval of a fuzzy variable; (2) determination of fuzzy control rules; (3) selection of membership functions. In order to do those tasks, continuous trial and tuning are necessary. Therefore, fuzzy control technology has been hindered to some degree from gaining wider application and dissemination. As to (3), generally, it is not a true knotty problem which will be encountered in designing fuzzy controllers(FCs), because in most cases it is enough to solve the problem with membership functions of fixed form, such as triangle-shaped and trapezoid-shaped functions. The real difficulties are the tasks (1) and (2).

As to (1), usually at first, one must determine the universes of discourse (intervals) of fuzzy variables empirically, and then begin to try and adjust it to get a "better" interval. There is not a common way to get it directly. As to (2), there has been much work on the choice of linguistic control rules in the literature [6]-[13][17]. But recently, some fusion techniques have attracted people's attention, such as NN fuzzy and GA fuzzy [1][4][16][18]. An early and typical work is the Bart Kosko's book "Neural Networks and Fuzzy Systems", which describes a method of adaptive FAM-rule generation system (abbreviated as MFAM in this paper) [1]. Our algorithm of self-learning and self-completing (ASLSC) to be introduced in this paper is mainly based on MFAM and the parametric functions method [14]. It is composed of three sub-algorithms; that is, an algorithm for generating universes of discourse (AGUD), and an algorithm for generating

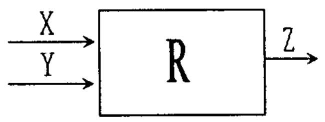


Figure 1: A diagram of a dual control system

control rules (AGCR), and an algorithm for completing control rules (ACCR). Each sub-algorithm may be used as an independent one. So we may introduce them separately. Because AGCR is an important postulate in this paper, we will first discuss it in Section 2. Section 3 will introduce ACCR, and Section 4 will introduce AGUD and ASLSC. Section 5 will report the simulation results from an inverted pendulum and Section 6 will report the experimental results from an industrial electric heating ring. Finally Section 7 will discuss some other related problems.

2 AGCR

The basic idea of AGCR is to make the division of the input and output spaces crisp, and cluster all sample data with a kohonen network, and count the number of weight vectors in each crisp subspace, and the middle rule among the incompatible rules is selected. Before describling the AGCR algorithm, we make some technical assumptions. Suppose that the control system is a dual control system(the plural is similar) as shown in figure 1 in which R represents the fuzzy controller with two input variables X and Y and an output variable Z, the rules have the following form:

if X is A_i and Y is B_i, then Z is C_i,
or $(A_i, B_i; C_i)$ for short

(1)

where X, Y, Z have dual meanings on notation, and they are viewed as the universes of discourse and fuzzy variables. Fuzzy variable X assumes fuzzy-set values $A_1, A_2, \ldots, A_m$, and fuzzy variable Y assumes fuzzy-set values $B_1, B_2, \ldots, B_n$, and fuzzy variable Z assumes fuzzy-set values $C_1, C_2, \ldots, C_p$, where A_i, B_j, C_k are linguistic values, such as positive large, negative small,etc.. For clarification, we often assume $m = n = p = 7$

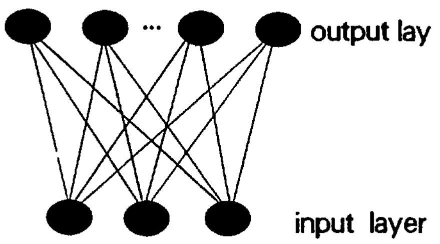


Figure 2: An adaptive clustering network

and the linguistic values set consists the following seven fuzzy sets:

$$\begin{gathered}
A_1 = B_1 = C_1 = NegativeLarge(NL) \\
A_2 = B_2 = C_2 = NegativeMiddle(NM) \\
A_3 = B_3 = C_3 = NegativeSmall(NS) \\
A_4 = B_4 = C_4 = Zero(ZE) \\
A_5 = B_5 = C_5 = PositiveSmall(PS) \\
A_6 = B_6 = C_6 = PositiveMiddle(PM) \\
A_7 = B_7 = C_7 = PositiveLarge(PL)
\end{gathered}$$

Suppose the input space X and Y are the real intervals $[X_l, X_r]$ and $[Y_l, Y_r]$, respectively, and the output space Z is the real interval $[Z_l, Z_r]$. We also assume we have a known sample data set: $\{s(i) = (x_i, y_i, z_i) | x_i \in [X_l, X_r], y_i \in [Y_l, Y_r], z_i \in [Z_l, Z_r], i = 1, 2, \ldots, N\}$ where N is the total number of all sample data and $N \gg m \times n \times p$.

To get the fuzzy control rule table from the sample data set, at first, a two-layer network which has 3 input nodes and $m \times n \times p$ output nodes must be made, as shown in figure 2. It is a kind of kohonen's adaptive clustering network, whose purpose is to cluster the sample data set, so the number of input nodes is equal to that of components of a sample datum, i.e. 3 here. Because a cluster corresponds to a fuzzy control rule and the most of rules is $m \times n \times p$, the number of output nodes is certainly $m \times n \times p$. On the other hand, like [1], the membership functions here are assumed to have fixed shape. For instance, we assume in this paper they have the forms of symmetrical triangle-shaped functions shown in figure 3. One of the advantages of choosing such membership functions is that, to any point in the universe of discourse, a fuzzy set always exists to make the membership degree of the point belonging to the fuzzy set greater than or equal to 0.5.

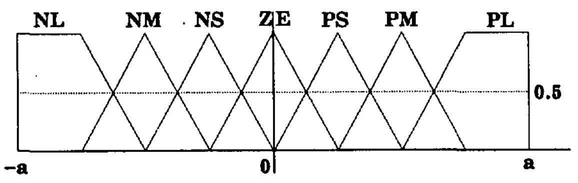


Figure 3: Membership functions have the forms of symmetrical triangle-shaped functions

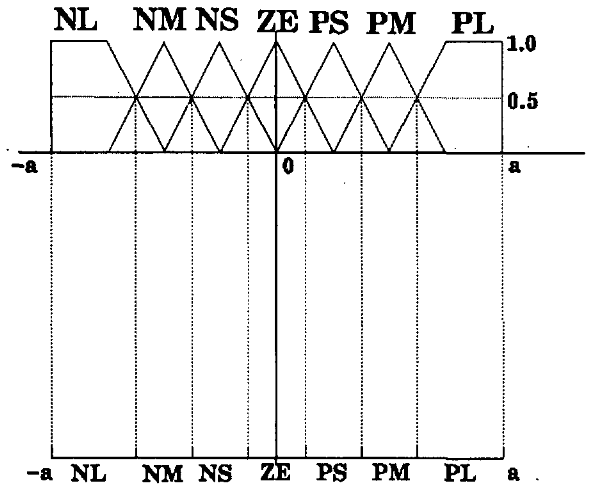


Figure 4: According to the "0.5 partition principle" to partition [-a, a] into 7 intervals

With the preparation above, we describe AGCR as follows:

- Step 1. Partitioning the universes of discourse (or the input and ouput spaces)

  This step is to make the division of the input and output spaces crisp. We present a principle of partitioning that the point whose membership degree belonging to a fuzzy set is greater than or equal to 0.5 should be included into the corresponding interval. For instance, to the case of figure 3, the diagram of partitioning is shown in figure 4. We call it the "0.5 partition principle".

- Step 2. Training the samples set with unsupervised competitive learning(UCL)

  The purpose of this step is to cluster the sample data set. Here we don't use differential competitive learning(DCL) which is always used in [1], because we have proved that UCL possesses faster speed in computing and occupies less memory than DCL [15].

  - (1) Initialize the weight vectors

    $$\mathbf{w}_i(0) = \mathbf{s}(i), i = 1, 2, \ldots, m \times n \times p.$$

  - (2) For additional random sample $\mathbf{s}(t)$, find the closest ("winning") weight vector $\mathbf{w}_j(t)$:

    $$\|\mathbf{w}_j(t) - \mathbf{s}(t)\| = \min_i \|\mathbf{w}_i(t) - \mathbf{s}(t)\| \quad (2)$$

    where $\|\mathbf{w}\|^2 = w_1^2 + w_2^2 + w_3^2$ gives the squared Euclidean norm of $\mathbf{w}$.

  - (3) Update the winning weight vector(s)

    $$\begin{cases} \mathbf{w}_j(t+1) & = \mathbf{w}_j(t) + c_t[\mathbf{s}(t) - \mathbf{w}_j(t)] \\ \mathbf{w}_i(t+1) & = \mathbf{w}_i(t), \quad \text{if } i \neq j \end{cases} \quad (3)$$

    where $\{c_t\}$ defines a slowly decreasing sequence of learning coefficients. For instance, $c_t = 0.1[1 - (t/1000)]$ for 1000 samples $\mathbf{s}(t)$.

- Step 3. Generating rules

  After step 2, all sample data have been clustered to $m \times n \times p$ clusters whose centers are those $m \times n \times p$ weight vectors respectively. This step is to count the number of the weight vectors in each crisp sub-space and generate rules with a weight preliminarily.

  - (1) Initialization

    $p_{ijk} = 0.q = 1, i = 1, 2, \ldots, m.j = 1, 2, \ldots, n.k = 1, 2, \ldots, p.$

  - (2) To all weight vectors $\mathbf{w}_q(s), q = 1, 2, \ldots, m \times n \times p, \mathbf{w}_q = (w_q^1, w_q^2, w_q^3)$,

    $$\begin{cases} p_{ijk} = p_{ijk} + 1, & \text{if } w_q^1 \in A_i, w_q^2 \in B_j, \\ & w_q^3 \in C_k \\ p_{ijk} = p_{ijk}, & \text{else} \end{cases} \quad (4)$$

    where p_{ijk} indicates the number or the frequency of all the weight vectors $\mathbf{w}_q(s)(q = 1, 2, \ldots, m \times n \times p)$ falling in the cuboid composed of (A_i, B_j, C_k) in R^3. We double up on notation and view A_i, B_j and C_k as fuzzy variables as well as intervals.

- (3) If $p_{ijk} \neq 0$, then rule $(A_i, B_j; C_k)$ exists, and A_i, B_j and C_k are three fuzzy sets. The weight of this rule is

$$\frac{p_{ijk}}{T}, \tag{5}$$

where

$$T = \sum_{i,j,k} p_{ijk} \tag{6}$$

The rule $(A_i, B_j; C_k)$ can also be considered as a mapping [14]:

$$\varphi: \mathcal{X} \times \mathcal{Y} \to \mathcal{Z} \tag{7}$$

where $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ is the linguistic values sets for the variables X, Y and Z, respectively, and $\mathcal{X} = \{A_1, A_2, \ldots, A_m\}$, $\mathcal{Y} = \{B_1, B_2, \ldots, B_n\}$ and $\mathcal{Z} = \{C_1, C_2, \ldots, C_p\}$. So the mapping (7) can be represented as $\varphi(A_i, B_j) = C_{f(i,j)}$, where f is a function of the following form

$$f: \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \to \{1, 2, \ldots, p\} \tag{8}$$

If $f(i,j)$ is many-valued, it means that incompatible rules exist.

- Step 4. Optimizing rules

This step is to select one optimal rule among the incompatible rules. Because in this method all the last rules will be displayed in a rule table on the computer screen (see figure 5), the incompatible rules are not permitted to exist. Only one rule must be selected from a group of incompatible rules. If a group of incompatible rules contains α rules and if α is even (often to be 2, 4 and 6), then one rule which has a bigger weight (or frequency) should be selected from the two "middle" rules, and if α is odd (often to be 3, 5 and 7), then the "middle" rule should be selected. We explain it with formulas:

Suppose there are α incompatible rules whose antecedent is A_i and B_j. It means $f(i,j)$ is α-valued and assume

$$f(i,j) = \{f(i,j)(1), f(i,j)(2), \ldots, f(i,j)(\alpha)\} \tag{9}$$

where $1 \leq f(i,j)(1) < f(i,j)(2) < \ldots < f(i,j)(\alpha) \leq p$.

θ\θ	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PL	PS	PS	ZE
NM	PL	PL	PL	PM	ZE	ZE	NS
NS	PL	PL	PM	PS	ZE	NS	NM
ZE	PL	PM	PS	ZE		NM	NL
PS	PZ	ZE	ZE	NS	NM	NL	NL
PM	ZE	ZE	NS	NM	NL	NL	NL
PL	ZE	ZE	NS	NL	NL	NL	NL

Table 1: An example of rule table

If α is odd, then $f(i,j)(\frac{\alpha+1}{2})$ is the middle value of (9). So $\varphi(A_i, B_j) = C_{f(i,j)(\frac{\alpha+1}{2})}$ is the only one that should be selected.

If α is even, then there are two middle values in (9), and they are $f(i,j)(\frac{\alpha}{2})$, $f(i,j)(\frac{\alpha}{2}+1)$. Let $k_1 = f(i,j)(\frac{\alpha}{2})$, $k_2 = f(i,j)(\frac{\alpha}{2}+1)$. Then the two middle rules are $(A_i, B_j; C_{k_1})$ and $(A_i, B_j; C_{k_2})$. Their corresponding frequencies are p_{ijk_1} and p_{ijk_2}. If $p_{ijk_1} \geq p_{ijk_2}$, then $(A_i, B_j; C_{k_1})$ is the only one that should be selected, else $(A_i, B_j; C_{k_2})$ is the only one that should be selected.

We call the technique above the "selecting the middle rule (SMR)".

3 ACCR

In the simulation results of AGCR to be introduced later, it always occurs that some rules are absent in the last rule table. For example, Table 1 misses a rule, i.e., (PS, ZE; 00), where "00" represents this rule is absent or lost.

Absence of a rule will affect controlling. How can this problem be solved? PENG Xian-Tu [14] had proposed a method of generating rules by functions, such as by linear functions, but a linear function must be a monotonic function. It means that the monotonicity exists in a rule table. We also find this hidden law in many other cases, and we will use it to solve the above problem. This method is called ACCR which always assumes the rule table is monotonic.

For example, returning to table 1, if we regard NL, NM, NS, ZE, PS, PM and PL as 1, 2, 3, 4, 5, 6 and 7, respectively, we can find the monotonic tendency decreasing from the upper-left corner to the lower-right corner, and in a row from left to right, and in a column from up to down. Then, for the blank in table 1, we can guess it might be NS.

Hence, in the case that some rule is absent, we should observe the rules on it's left and right and up and down, and fill a proper rule according to the monotonicity law. This is the very idea of the "self-completing".

For example, if the linguistic values sets $\mathcal{X} = \mathcal{Y} = \mathcal{Z} = \{NL, NM, NS, ZE, PS, PM, PL\}$, then the rule table has at most 49 rules. ¿From (8), any rule may be represented as $\varphi(A_i, B_j) = C_{f(i,j)}$, where f is a function of the form

$$f : \{1, 2, \ldots, 7\} \times \{1, 2, \ldots, 7\} \rightarrow \{1, 2, \ldots, 7\}.$$

Additionally, we assume $f(i, j) = 0$, if the rule whose antecedent is A_i and B_j is absent. Then an special ACCR may be summed up as the following. We view ACCR as a method which makes a rule table complete, not only a specific algorithm.

- Let $f(1, 1) = 7, f(4, 4) = 4, f(7, 7) = 1$.
- For $\forall i, j, 1 \leq i \leq 4, 2 \leq j \leq 3, f(i, j) = 0$, if $f(i, j-1) = f(i, j+1) > 0$, then $f(i, j) = f(i, j+1) = f(i, j-1)$, else if $f(i, j+1) > 0$, then let $f(i, j) = \min((f(i, j+1) + 1), 7)$.
- For $\forall i, j, 2 \leq i \leq 3, 1 \leq j \leq 4, f(i, j) = 0$, if $f(i-1, j) = f(i+1, j) > 0$, then $f(i, j) = f(i+1, j) = f(i-1, j)$, else if $f(i+1, j) > 0$, then let $f(i, j) = \min((f(i+1, j) + 1), 7)$.
- For $\forall i, j, 4 \leq i \leq 7, 5 \leq j \leq 6, f(i, j) = 0$, if $f(i, j-1) = f(i, j+1) > 0$, then $f(i, j) = f(i, j+1) = f(i, j-1)$, else if $f(i, j-1) > 0$, then let $f(i, j) = \max((f(i, j-1) - 1), 1)$.
- For $\forall i, j, 5 \leq i \leq 6, 4 \leq j \leq 7, f(i, j) = 0$, if $f(i-1, j) = f(i+1, j) > 0$, then $f(i, j) = f(i+1, j) = f(i-1, j)$, else if $f(i-1, j) > 0$, then let $f(i, j) = \max((f(i-1, j) - 1), 1)$.
- For $\forall i, j, 1 \leq i \leq 3, 5 \leq j \leq 6, f(i, j) = 0$, if $f(i, j-1) > f(i, j+1) > 0$, then $f(i, j) = f(i, j-1) - 1$, if $f(i, j-1) = f(i, j+1) > 0$, then $f(i, j) = f(i, j-1) = f(i, j+1)$.
- For $\forall i, j, 2 \leq i \leq 3, 5 \leq j \leq 7, f(i, j) = 0$, if $f(i-1, j) > f(i+1, j) > 0$, then $f(i, j) = f(i-1, j) - 1$, if $f(i-1, j) = f(i+1, j) > 0$, then $f(i, j) = f(i-1, j) = f(i+1, j)$.
- For $\forall i, j, 5 \leq i \leq 7, 2 \leq j \leq 3, f(i, j) = 0$, if $f(i, j-1) > f(i, j+1) > 0$, then $f(i, j) = f(i, j-1) - 1$, if $f(i, j-1) = f(i, j+1) > 0$, then $f(i, j) = f(i, j-1) = f(i, j+1)$.
- For $\forall i, j, 5 \leq i \leq 6, 1 \leq j \leq 3, f(i, j) = 0$, if $f(i-1, j) > f(i+1, j) > 0$, then $f(i, j) = f(i-1, j) - 1$, if $f(i-1, j) = f(i+1, j) > 0$, then $f(i, j) = f(i-1, j) = f(i+1, j)$.
- For $\forall i, j, 1 \leq i \leq 7, 1 \leq j \leq 7$, if $f(i, j) = 1$, for $\forall i', j', i \leq i' \leq 7, j \leq j' \leq 7$, then $f(i', j') = 1$.
- For $\forall i, j, 1 \leq i \leq 7, 1 \leq j \leq 7$, if $f(i, j) = 7$, for $\forall i', j', 1 \leq i' \leq i, 1 \leq j' \leq j$, then $f(i', j') = 7$.

4 AGUD and ASLSC

In many methods, including the MFAM, it is necessary to know each universe of discourse of each fuzzy variable beforehand. However this is difficult, because we do not have a theorical method to advise us how to determine the universes of discourse of fuzzy variables, which are generally determined by experience and debugging [2]. Therefore, it will be very significant if a learning system can automatically determine the universes of discourse. We will present the following formulas to try to solve this problem, although it is also an experimental method.

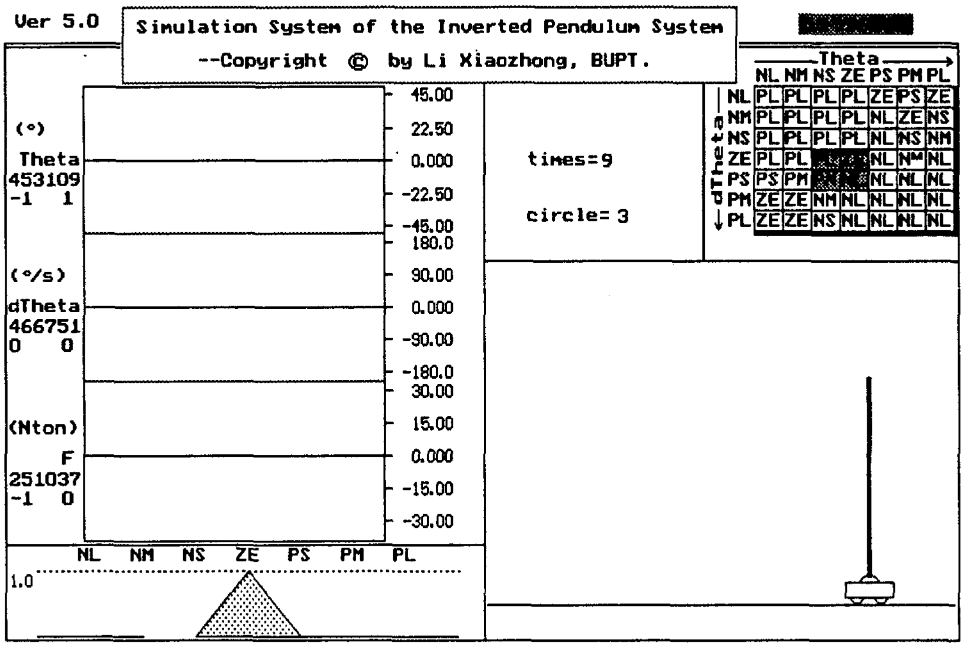
$$\left\{ \begin{array}{ll} X_r & = \max_{1 \leq i \leq N} |x_i| \\ X_l & = -X_r \\ Y_r & = \max_{1 \leq i \leq N} |y_i| \\ Y_l & = -Y_r \\ Z_r & = \frac{4}{3} \max_{1 \leq i \leq N} |z_i| \\ Z_l & = -Z_r \end{array} \right. \tag{10}$$

where $\{(x_i, y_i, z_i), i = 1, 2, \ldots, N\}$ is the sample data set.

It must be noted that (10) is not the only one as an AGUD, and in fact AGUD may be other forms of formulas. For example,

$$X_r = (X_1 + X_2 + X_3)/3, \quad X_l = -X_r \tag{11}$$

where
$X_1 = \max_{1 \leq i \leq N} |x_i|$, $X_2 = \max_{1 \leq i \leq N} \{|x_i| : |x_i| < X_1\}$, $X_3 = \max_{1 \leq i \leq N} \{|x_i| : |x_i| < X_2\}$, and so do Y and Z. Obviously, the new formulas have some effect of "filter" and they are robust.



Ver 5.0

Simulation System of the Inverted Pendulum System
--Copyright © by Li Xiaozhong, BUPT.

(°)
Theta
453109
-1 1

(°/s)
dTheta
466751
0 0

(Nton)
F
251037
-1 0

times=9

circle= 3

dTheta ↓ \ Theta →	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PL	ZE	PS	ZE
NM	PL	PL	PL	PL	NL	ZE	NS
NS	PL	PL	PL	PL	NL	NS	NM
ZE	PL	PL	[illegible]	[illegible]	NL	NM	NL
PS	PS	PM	[illegible]	[illegible]	NL	NL	NL
PM	ZE	ZE	NM	NL	NL	NL	NL
PL	ZE	ZE	NS	NL	NL	NL	NL

Figure 5: Simulation system of an inverted pendulum

ASLSC is a sequential composition of AGUD, AGCR and ACCR.

5 Simulation results

We have done some simulations with the algorithms above on an inverted pendulum system [3] shown in figure 5, a classic experiment target in control theory. The simulations have two purposes: one is to verify whether the rule table generated by AGCR+ACCR coincides with the original rule table with the same universes of discourse, the other is to explain that ASLSC is very effective to obtain the universes of discourse and fuzzy control rules if we know only the sample data of a control system.

At first, we must design a fuzzy controller which will help us to obtain a sample data set which contains 1000 pendulum trajectory data. The details of the fuzzy controller may refer to [1]. The defuzzification strategy we employ is the usual mini-max-center of gravity method [1][2]. These product-space training vectors $(\theta, \dot{\theta}, f)$ are points in R^3. The universes of discourse of $\theta, \dot{\theta}$ and f are the following intervals, respectively:

$$\begin{aligned} \theta &\in [-0.57, 0.57], \text{ the unit is radian} \\ \dot{\theta} &\in [-3.14, 3.14], \text{ the unit is radian/s} \\ f &\in [-30, 30], \text{ the unit is newton} \end{aligned}$$

Fuzzy variables assume only seven fuzzy-set values NL, NM, NS, ZE, PS, PM and PL, so there are 343 possible rules.

Additionally, all original rules of this controller include 49 rules which are displayed in table 2.

This controller can control the inverted pendulum very well. It restores the pendulum to equilibrium as we knock it over to the right and to the left. At the same time, input-output sample data reads automatically to a training data file.

Simulation 1

By knocking the pendulum randomly, we get a sample data file "data.1". With the same uni-

$\dot\theta\backslash\theta$	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PL	PS	PS	ZE
NM	PL	PL	PL	PM	ZE	ZE	NS
NS	PL	PL	PM	PS	ZE	NS	NM
ZE	PL	PM	PS	ZE	NS	NM	NL
PS	PS	ZE	ZE	NS	NM	NL	NL
PM	ZE	ZE	NS	NM	NL	NL	NL
PL	ZE	ZE	NS	NL	NL	NL	NL

Table 2: The original fuzzy rule table

$p_{117} = 3$	$p_{217} = 6$	$p_{317} = 5$	$p_{417} = 1$
$p_{127} = 3$	$p_{227} = 6$	$p_{327} = 8$	$p_{426} = 7$
$p_{137} = 2$	$p_{237} = 6$	$p_{326} = 1$	$p_{436} = 4$
$p_{147} = 2$	$p_{247} = 1$	$p_{336} = 2$	$p_{435} = 8$
$p_{156} = 1$	$p_{246} = 3$	$p_{346} = 4$	$p_{434} = 5$
$p_{156} = 3$	$p_{255} = 9$	$p_{345} = 4$	$p_{444} = 145$
$p_{155} = 10$	$p_{254} = 10$	$p_{344} = 7$	$p_{454} = 8$
$p_{164} = 2$	$p_{274} = 1$	$p_{355} = 5$	$p_{453} = 3$
		$p_{354} = 6$	$p_{452} = 3$
		$p_{364} = 1$	$p_{462} = 2$

Table 3: A clustering result in simulation 1

verses of discourse and membership functions, we use AGCR to generate rules:

First, by training a two-layer kohonen network with the samples of file "data.1", we get such a clustering result (Table 3):

Table 3 describes all rules and their frequencies, and certainly includes incompatible rules, (12) presents three such rules,

$$\begin{cases} p_{346} = 4 \\ p_{345} = 4 \\ p_{344} = 7 \end{cases} \tag{12}$$

where (12) repesents three rules: (NS, ZE; PM), (NS, ZE; PS) and (NS, ZE; ZE), and their frequencies are 4, 4 and 7, respectively.

Then, according to the SMR, we get the following rule table(Table 4) which does not contain incompatible rules:

$p_{524} = 9$	$p_{624} = 7$	$p_{733} = 1$
$p_{534} = 2$	$p_{623} = 4$	$p_{732} = 2$
$p_{552} = 1$	$p_{633} = 2$	$p_{742} = 2$
$p_{562} = 1$	$p_{661} = 4$	$p_{741} = 1$
$p_{561} = 3$	$p_{671} = 3$	$p_{751} = 2$
$p_{571} = 2$		$p_{761} = 2$

$\dot\theta\backslash\theta$	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PL			
NM	PL	PL	PL	PM	ZE	ZE	
NS	PL	PL	PM	PS	ZE	NS	NM
ZE	PL	PM	PS	ZE			NM
PS	PS	ZE	ZE	NS	NM		NL
PM	ZE		NS	NM	NL	NL	NL
PL		ZE			NL	NL	

Table 4:

$\dot\theta\backslash\theta$	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PL	PS	PS	ZE
NM	PL	PL	PL	PM	ZE	ZE	NS
NS	PL	PL	PM	PS	ZE	NS	NM
ZE	PL	PM	PS	ZE	NS	NM	NM
PS	PS	ZE	ZE	NS	NM	NL	NL
PM	ZE	ZE	NS	NM	NL	NL	NL
PL	ZE	ZE	NS	NL	NL	NL	NL

Table 5:

Last, using ACCR, we get a complete rule table(Table 5) which almost coincides with the original table(Table 2). The small difference between Table 2 and Table 5 lies in the position of line 5 and column 8, i.e., NL in Table 2 and NM in Table 5. This simulation result shows that AGCR can recover the original rules. In other words, if we know the variables and their universes of discourse in advance and have an expert or a skilled operator of a control system, we can generate or "recover" the fuzzy control rules in his mind in the process of his operation.

The purpose of the next two typical simulations is to verify the effectiveness of ASLSC in the situation of not knowing the universe of discourses in advance.

Simulation 2

We knock the pendulum heavily so that it swings in a large scope, then we get a sample file "data.2".

(1) From AGUD, we get the following intervals:

$$\begin{aligned} \theta &\in [-0.783862, 0.783862], \\ \dot\theta &\in [-5.217616, 5.217616], \\ f &\in [-30, 30], \end{aligned}$$

It can be seen that some sample points have gone beyond the original intervals [-0.57, 0.57] and [-3.14, 3.14].

$\dot{\theta}\backslash\theta$	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PL	PM	PS	ZE
NM	PL	PL	PL	PM	PS	ZE	ZE
NS	PL	PL	PL	PS	ZE	NS	NM
ZE	PL	PL	PM	ZE	NS	NM	NL
PS	PM	PS	ZE	ZE	NM	NL	NL
PM	PS	ZE	NM	NL	NL	NL	NL
PL	ZE	NS	NM	NL	NL	NL	NL

Table 6:

$\dot{\theta}\backslash\theta$	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PM	PS	ZE	ZE
NM	PL	PL	PM	PM	PS	ZE	ZE
NS	PL	PM	PM	PS	ZE	ZE	NS
ZE	PM	PS	ZE	ZE	ZE	NS	NM
PS	PS	ZE	ZE	NS	NM	NM	NL
PM	ZE	ZE	NS	NM	NM	NL	NL
PL	ZE	ZE	NS	NM	NL	NL	NL

Table 7:

(2) From AGCR and APCR, we get the following rule table 6.

Simulation 3

We knock the pendulum slightly so that it swings in a small scope, then we get a sample file "data.3".

(1) From AGUD, we get the following intervals:

$$\begin{aligned}\theta &\in [-0.312234, 0.312234],\\ \dot{\theta} &\in [-2.553612, 2.553612],\\ f &\in [-30, 30],\end{aligned}$$

(2) From AGCR and ACCR, we get the last rule table 7.

On the two simulations above the two sample data files are very different, so the last rule tables are different. It is obvious that table 6 is more suitable for controlling the pendulum in a large scope, and table 7 is more suitable for controlling the pendulum in a small scope. We cannot judge which rule table is better only according to which is closer to the original table 2. But we can judge by control curves. Figure 6 shows three control curves from the initial state $(\theta, \dot{\theta}) = (-10°, 0)$, where the curve A is obtained from the original rule table 2, and the curve B is from table 6 and the curve C is from table 7. Obviously, as viewed from the rapid rise and the small overshoot and the short settling time, C is better than B, and

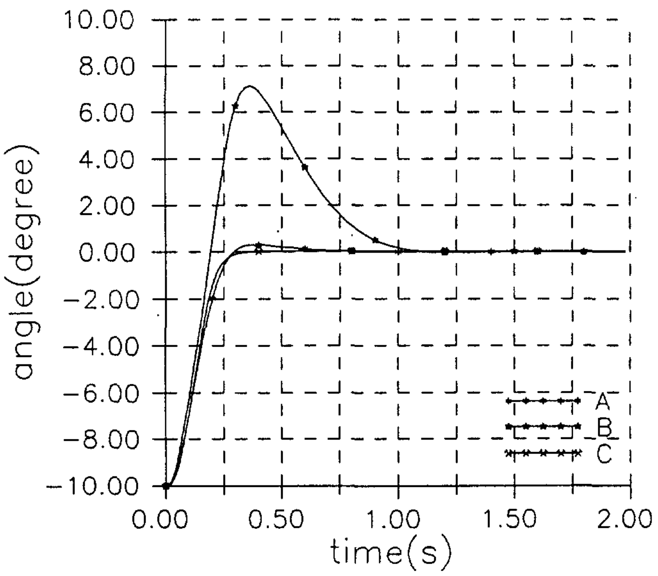


Figure 6: Comparisons among the three control curves. A, the control curve of the original table 2. B, the control curve of table 6. C, the control curve of table 7. It is obvious that C is closer to A than B.

C is very close to A. The reason is that some samples in the simulation 2 have gone beyond the original intervals, and this enlarges the variation ranges of the state variables. Even if the pendulum is near the zero point, it will be controlled by a larger force, such as (NS, ZE; PM) in table 6. Consequently, it will increase its overshoot. However, simulation 3 is exactly contrary to the variation ranges of θ and $\dot{\theta}$, therefore, the relevant intervals get smaller. Thus, even if the pendulum is at the end of the interval of θ or $\dot{\theta}$, it will be controlled by a smaller force, such as (PL, ZE; NM) and (NS, ZE; ZE) and (NL, ZE; PM) in table 7. None of their controlling force reaches NL or PL.

By means of the previous simulations, it is explained that ASLSC is very effective to produce fuzzy control rules. If we don't know the universes of discourse of fuzzy variables before learning, then different sample data files will cause different universes of discourse and different rule tables, but they all are able to control the system well.

$\Delta T \backslash \Delta T$	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PL	PS	PS	ZE
NM	PL	PL	PL	PM	ZE	ZE	NS
NS	PL	PL	PM	PS	NS	NM	NL
ZE	PL	PM	PS	ZE	NS	NM	NL
PS	PM	PS	PS	NS	NM	NL	NL
PM	ZE	ZE	NS	NM	NL	NL	NL
PL	ZE	ZE	NS	NL	NL	NL	NL

Table 8: The original rule table

6 Application

An introduction to the control system

The application described here is a control system for an industry heating ring. The sensor is an industrial thermocouple of E type. The controller system is a 286 microcomputer which contain an A/D-D/A converter and a set of fuzzy controlling software. Additionally, this control system also includes an isolated transmitter and a silicon controlled rectifier (SCR), as seen in figure 5.

The signal picked by the thermocouple is transmitted to the isolated transmitter which receives a weak signal and outputs a standard computer signal (0-5V). Then this analog signal is converted to a digital signal by the A/D converter. After the computer receives the digital signal, the program installed in the computer will process the signal and output a control signal. Next this output signal will be converted to an analog signal and used to control the heat generating power of the heating ring to keep the temperature at a desired level.

Results of applying the AGCR+ACCR

The method with which the sample data file is produced is the same as that of the above simulations, and the original rule table is table 8.

In this table, $\Delta T = T - T_0$, T_0 is the set value, $\Delta \dot{T} = \Delta T_2 - \Delta T_1 = T_2 - T_1$. Any one rule has the form of the following:

if ΔT is A and $\Delta \dot{T}$ is B, then ΔU is C.

where

$A, B, C \in \{NL, NM, NS, ZE, PS, PM, PL\}$ and ΔU is the process-input-change.

The universes of discourse of $\Delta T, \Delta \dot{T}, \Delta U$ are the following intervals, respectively:

$$[\Delta T_l, \Delta T_r] = [-10.0, 10.0]$$

ΔT	$\Delta \dot{T}$	ΔU
-0.280000	-0.360000	0.791367
1.160000	2.360000	-3.750000
1.160000	0.680000	-1.673780
-1.542857	-1.862857	3.217691
-0.480000	1.005714	-1.204858
-2.533333	-1.853333	1.640484
-4.040000	-2.097143	3.750000
...	...	...
-2.133333	1.906667	-1.908245

Table 9: Some sample data

$\Delta T \backslash \Delta T$	NL	NM	NS	ZE	PS	PM	PL
NL	PL		PL	PM	PS		ZE
NM	PL	PL	PL	PM		ZE	
NS	PL	PL	PM	PS	ZE	NS	NS
ZE	PL	PM	PS	ZE	NS	NM	NL
PS	PM	PS	ZE	NE	NM		
PM	PS	ZE	NS	NM	NL		
PL	ZE	NS	NM	NL			

Table 10: The rule table from AGCR

$$[\Delta \dot{T}_l, \Delta \dot{T}_r] = [-3.0, 3.0]$$

$$[\Delta U_l, \Delta U_r] = [-5.0, 5.0]$$

Assume $T_0 = 170°$. We let the temperature rise about $10°C$ above the set value. Then we use table 8 to control the temperature, and at the same time input-output sample data reads automatically to a training data file. Table 9 lists some sample data.

With the AGCR, we get the following table 10:

Applying ACCR to table 10, we get the rule table 11:

It can be seen that table 11 is much similar to the original rule table 8. This shows again that

$\Delta T \backslash \Delta T$	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PM	PS	PS	ZE
NM	PL	PL	PL	PM	PS	ZE	NS
NS	PL	PL	PM	PS	ZE	NS	NS
ZE	PL	PM	PS	ZE	NS	NM	NL
PS	PM	PS	ZE	NS	NM	NL	NL
PM	PS	ZE	NS	NM	NL	NL	NL
PL	ZE	NS	NM	NL	NL	NL	NL

Table 11: The rule table from AGCR+ACCR

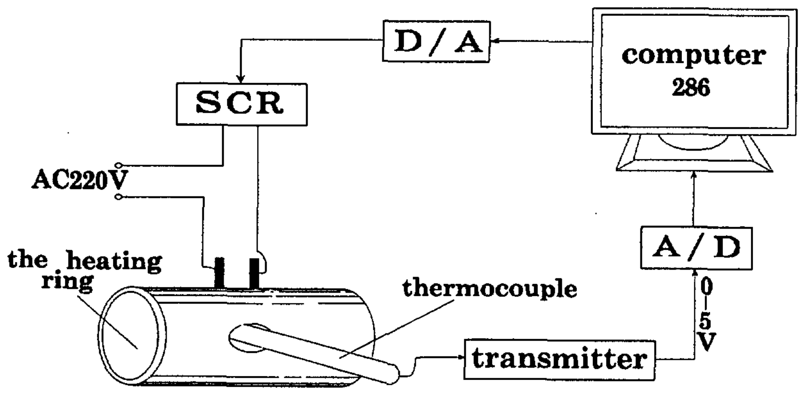


Figure 7: Fuzzy control system for temperature

$\Delta T\backslash\Delta T$	NL	NM	NS	ZE	PS	PM	PL
NL	PL	PL	PL	PL	PM	ZE	ZE
NM	PL	PL	PL	PL	NS	ZE	ZE
NS	PL	PL	PL	PM	NM	NS	ZE
ZE	PL	PL	PM	ZE	NM	NL	NL
PS	PL	PL	PM	NS	NL	NL	NL
PM	PS	PS	ZE	NM	NL	NL	NL
PL	ZE	ZE	NS	NL	NL	NL	NL

Table 12:

AGCR plus ACCR is effective for learning and recovering the original rules.

Result of applying the AGUD+AGCR +ACCR

At first we apply the AGUD to the sample data file which has generated tables 10 and 11, and we get the following universes of discourse,

$$[\Delta T_l, \Delta T_r] = [-16.6, 16.6]$$

$$[\Delta\dot{T}_l, \Delta\dot{T}_r] = [-2.48, 2.48]$$

$$[\Delta U_l, \Delta U_r] = [-2.6, 2.6]$$

Then with the AGCR+ACCR, we have table 12.

When we use this rule table to control the temperature, we find the temperature accuracy is satisfactory, since the error is within $\pm 1°C$. However, if we compare table 12 to table 8, we find the control force appears bigger than that in table 8. For example, some place, where NS or PS stays in table 8, has become the place where NM or PM stays in table 12. The main reason is that the interval of ΔT is enlarged, and the interval of ΔU is shrunk. The further analysis is the same as that in Section 5. This experiment shows again that ASLSC may be used to generate the universes of discourse and fuzzy control rules for a fuzzy control system if we have some sample data in advance.

7 Discussions

This paper has presented a compositional algorithm,i.e. ASLSC, and the core of the ASLSC is AGCR. Although ASLSC is developed from the MFAM, it is different from MFAM. At first, the methods of partitioning intervals are different. MFAM always uses the uninform partitioning method, but ASLSC partitions the intervals by "the 0.5 partition principle". From many simulation experiments,

we have found this principle can help us to greatly increase the accuracy of the last learning result, especially for the membership functions as shown in the figure 3 [15]. Finally, the incompatible rules are forbidden to exist in the last result of ASLSC, but they are permitted to exist with a weight in MFAM [1]. So, the two results of the two methods are different.

For a group of incompatible rules, our method

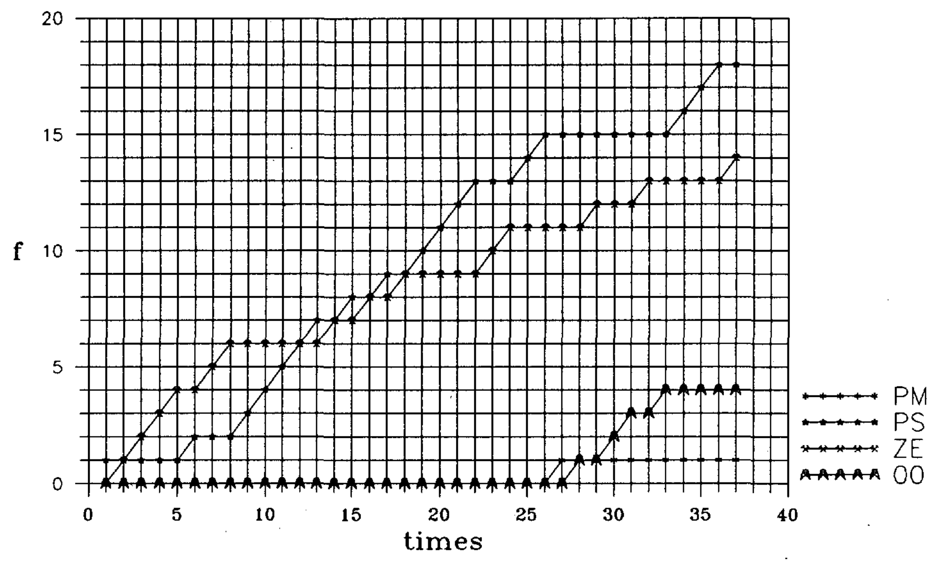


Figure 8: The frequency trajectories of the four possible rules: (ZE, NS; PM), (ZE, NS; PS), (ZE, NS; ZE) and (ZE, NS; 00)

is to "select the middle rule" which is explained in Section 2. Why don't we choose the rule which has the biggest weight or frequency? Intuitively, we consider the appearance of incompatible rules results from some perturbance. For example, we just simply define FAM-cell [1] boundaries in the input-output product space $X \times Y \times Z$ with the clear partition method. But in fact, FAM cells, which correspond to the cuboids mentioned in Section 2, should have overlaped, since contiguous quantizing fuzzy sets A_i and A_{i+1}, and B_j and B_{j+1}, and C_k and C_{k+1} overlap. So the FAM cells are fuzzy subsets of $X \times Y \times Z$, and their boundaries should be fuzzy but not clear. Now, when we replace the fuzzy boundaries with the clear boundaries, of course, it will cause some perturbance. Obviously, this kind of perturbance diffuses in all directions and diffusion probabilities are the same. Thus forms the incompatible rules, and certainly, the centroidal rule of a group of incompatible rules is the "core". As for the frequency of the rule, we think it is not the most important piece of data, since it has strong randomness. So, we always choose the middle rule. Additionally, it is also an experimental conclusion [15]. For example, in simulation 1 on inverted pendulum reported above, equation (12) is a group of incompatible rules, if we choose the rule of the biggest frequency, then (NS, ZE; ZE) is selected. According to table 2, it is wrong. However, if we choose the middle rule, then (NS, ZE; PS) is chosen and it is right.

For the AGCR, although we think it is a good method of recovering the original rules, we are not sure that it can generate the very same result in each time. Because randomness exists in random sampling every time, it also exists in the last learning result. Of course, the influence of this randomness is not very serious. For instance, the rule (ZE, NS; PS) is the most mutable, and most of the others are not mutable. To overcome such randomness, we have created a statistical method [15], which requires many times of leaning, and counting the number of times that each possible rule presents itself. Only the most-frequent rules are decided to be the last rules. For example, in 37 times learning, (ZE, NS; PS) is present 18 times, and (ZE, NS; ZE) is present 14 times, and (ZE, NS; PM) is present 1 time, and (ZE, NS; 00.is present 4 times. Therefore (ZE, NS; PS) should be chosen. Figure 8 describes the frequency trajectories of the four rules.

ACCR does not play the most important role in this paper. ACCR is different from PENG Xian-Tu's function method [14]. Because ACCR is only for completing a rule table which has just lost some rules but not for making a whole rule table. But the function method may make a whole rule table. Obviously, ACCR may have other forms. As for AGUD, it is still an experimental result. Of course it may have other versions. We hope ASLSC will be able to generate membership functions in future, and we believe some better results on ASLSC would be obtained after further research and application.

References

[1] Bart Kosko, (1990)*Neural Networks and Fuzzy Systems*, Prentice Hall, 1990.

[2] C. C. Lee, Fuzzy logic in control systems: Fuzzy logic controller -Part I, (1990)*IEEE Trans. Syst. Man Cybern*, Vol. 20. no. 2, pp 404-418.

[3] A. G. Barto, R. S. Sutton and C. W. Anderson, (1983) Neurolike adaptive elements that can solve difficult learning problem. *IEEE Trans. Syst. Man, Cybern*, vol. SMC-13, No. 5, pp 834-846.

[4] Jyh-Shing R. Jang, (1992) Self-learning fuzzy controllers based on Temporal Back Propagation, *IEEE TRANS. on Neural Networks*, vol. 3, No. 5.

[5] E. H. Mamdani and S. Assilian, (1975) An experiment in linguistic synthesis with a fuzzy logic controller, *Int. J. Man Mach. Studies*, vol 7, no. 1, pp. 1-13.

[6] M. Sugeno and K. Murakami, (1984) Fuzzy parking control of model car, *23rd IEEE Conf. on Decision and Control*, Las Vegas.

[7] M. Sugeno and M. Nishida, (1985) Fuzzy control of model car, *Fuzzy Sets Syst.*, vol. 16, pp. 103-113.

[8] M. Sugeno and K. Murakami, (1985) An experimental study on fuzzy parking control using a model car, *Industrial Applications of Fuzzy Control*, M. Sugeno, Ed. Amsterdam:North-Holland, pp. 125-138.

[9] M. Sugeno and G. T. Kang. (1988) Structure identification of fuzzy model, *Fuzzy Sets Syst.* , vol. 28, no. 1, pp. 15-33.

[10] T. Takagi and M. Sugeno, (1983) Derivation of fuzzy control rules from human operator's control actions, *Proc. of the IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, Marseilles, France, July, pp. 55-60.

[11] E. Czogala and W Pedrycz, (1981) On identification in fuzzy systems and its applications in control problems, *Fuzzy sets Syst.* , vol. 6, no. 1, pp. 73-83.

[12] E. Czogala and W Pedrycz, (1982) Fuzzy rule generation for fuzzy control, *Cybern. Syst.*, vol. 13, no. 3, pp. 275-293.

[13] C. W. Xu and Y. Zailu, (1983) Fuzzy model identification and self-learning for dynamic systems. *IEEE trans. Syst. Man Cybern.* vol. SMC-17, no. 4, pp. 683-689.

[14] PENG Xian-Tu, (1990) Generating rules for fuzzy logic controllers by functions, *Fuzzy Sets and Systems* Vol.36. pp.83-89.

[15] Xiaozhong Li, (1994) *Researches on a self-learning and adaptive fuzzy systems and fuzzy neural networks.* Ph. D thesis, Beijing Univ. of Post & Telecomms.

[16] H. Takagi, (1993) Fusion techniques of fuzzy systems and neural networks, and fuzzy systems and genetic algorithms, *SPIE Proc. of Technical Conference on Applications of Fuzzy Logic Technology, SPIE's Symposium on Optical Tools for Manufacturing and Advanced AUtomation*, Vol. 2061 pp.402-413, Boston, MA.

[17] L. X. Wang and J. M. Mendel, (1992) Generating fuzzy rules by learning from examples, *IEEE Trans. on Syst. Man Cybern.* , 22-6, pp.1414-1427.

[18] L. X. Wang and J. M. Mendel, (1992) Fuzzy basis functions, universal approximation, and orthogonal least-square learning, *IEEE Trans. on Neural Networks*, Vol. 3. No. 5, pp. 807-814.

Performance Bounds on Scheduling Parallel Tasks with Setup Time on Hypercube Systems

Jiann-Fu Lin and Sao-Jie Chen
Department of Management
Takming Junior College of Commerce
Taipei, Taiwan, R.O.C.

Keywords: heuristic algorithm, scheduling, parallel task, setup time, performance bound

Edited by: Jerzy R. Nawrocki
Received: August 15, 1994 **Revised:** May 5, 1995 **Accepted:** August 27, 1995

This paper investigates the problem of scheduling "parallel tasks" with consideration of setup time on a d-dimensional hypercube system, where scheduled tasks are independent such that each task can simultaneously require a subcube for its processing under the constraint that the dimension of a required subcube cannot be greater than the maximum parallelism dimension of a task. Whenever there is a switching of schedule from a task to another one, setup time is necessary. The objective of this problem is to find a schedule with minimum finish time, such a scheduling problem is NP-hard. Therefore, in this paper, we will propose two heuristic algorithms for this kind of problem and derive their performance bounds, respectively.

1 Introduction

The problems of scheduling tasks on multiprocessor systems have been extensively studied for a long time. Conventional approaches [1, 7, 12] were proposed for the problem of scheduling each task on only one processor at a time, which is referred to as the "conventional task scheduling problem". Finding a minimum finish time nonpreemptive schedule for this type of problem on an m-processor system, where $m \geq 2$, is *NP-complete* [12]. Given a list of tasks with indices 1, 2, ..., n, the *list scheduling* (LS) algorithm proposed by Graham[7] assigns tasks in the order that they appear on the list. For any given task set, the performance bound of LS was shown as $(\frac{S_{LS}}{S_O}) \leq (2 - \frac{1}{m})$, where S_O and S_{LS} denote the finish times of an optimal schedule and that of a list schedule, respectively. Another heuristic studied by Graham [7] is the *largest processing time first* (LPT) scheduling algorithm, which states that tasks are selected to assign according to the nonincreasing order of their processing times, and whenever a task is selected, it will be assigned to the first free processor that it meets. If S_O is the finish time of an optimal schedule and S_{LPT} is that of an LPT schedule for any given task set then it was shown that $(\frac{S_{LPT}}{S_O}) \leq (\frac{4}{3} - \frac{1}{3m})$. Note that the LPT scheduling algorithm is a special case of the LS algorithm, where tasks in the LPT list are ordered nonincreasingly by their processing times.

Afterwards, the above "conventional tasks scheduling problem" has been modified to generate the "multiprocessor tasks scheduling problem" [2, 3, 8], where tasks can be processed on a different number of processors and are thus called "multiprocessor tasks". The latter problem considers a task set $T = \{T_1, T_2, ..., T_n\}$ which contains n independent multiprocessor tasks to be processed on an m-processor system. Each multiprocessor task T_i in T is associated with a fixed number p_i to denote the number of processors required. For this problem type, a schedule is feasible if each task T_i can be exactly processed by p_i processors simultaneously, and the performance of a feasible schedule is measured by its finish time such that a feasible schedule is called an optimal schedule if it has the minimum finish time. For this problem type, the complexity of obtaining a schedule with minimum finish time is *NP-hard* [3]. In solving the multiprocessor task scheduling problem, polynomial time algorithms were proposed by Blazewicz, Drabowski, and Weglarz [3]. But these algorithms were only optimal for two constrained cases of independent multiprocessor task schedulings: one case assumed that all multiprocessor tasks require one-unit of processing time; the other one was that multiprocessor tasks can have arbitrary processing time, but they need to be preemptive. Algorithms proposed by Chen and Lai [4, 5], and Zhu and Ahuja [16, 17] for scheduling tasks on a d-dimensional hypercube system can be seen as a special case of the multiprocessor tasks scheduling problem. All their proposed algorithms, i.e., *the Largest Dimension Largest Processing Time* (LDLPT) [4] and the *Largest Dimension First* (LDF) [17] algorithms, have a same performance bound of $(2 - \frac{1}{2^d})$.

Recently, the problem of parallel tasks scheduling [6, 9, 13, 14] has been introduced. This problem is a little bit different from that of multiprocessor tasks scheduling. In the parallel tasks scheduling problem, a set of n parallel tasks $\mathbf{T} = \{T_1, T_2, ..., T_n\}$ are to be scheduled on an m-processors system, and each task T_i has its *maximum degree of parallelism* Δ_i and computation requirement t_i. This implies that a task T_i may be scheduled to process on up to Δ_i processors and this degree of parallelism, once decided for T_i, will not be altered during its processing. Suppose that a task T_i is scheduled to process on δ_i processors, $1 \leq \delta_i \leq \Delta_i$, then d_i is called the scheduled parallelism of T_i and the processing time required by T_i will be $\frac{t_i}{\delta_i}$. For this problem type, a schedule is feasible if the scheduled parallelism δ_i of each parallel task T_i is no greater than its maximum degree of parallelism Δ_i. To find an optimal schedule for such parallel tasks scheduling problem is *NP-hard* [6], here the optimal schedule means a feasible schedule with minimum finish time. Under the linear speedup assumption, Wang and Cheng [13] showed that the performance bound of Graham's list scheduling algorithm applied to this problem is $(\Delta + \frac{m-\Delta}{m})$, where m is the number of processor in a system and $\Delta = \max\{\Delta_i | i = 1, 2, ..., n\}$. Lately, Wang and Cheng [14] proposed an *Earliest Completion Time* (ECT) algorithm for scheduling parallel tasks and derived a performance bound of $(3 - \frac{2}{m})$. Recently, Lin and Chen [9] proposed an *Modified Largest Dimension First* (MLDF) algorithm for scheduling parallel tasks with communication cost on a d-dimensional hypercube system, which has a performance bound of $(2 + \ln 2^d - \frac{1}{2^d})$.

In this paper, the problem of nonpreemptively scheduling independent parallel tasks on a d-dimensional hypercube with setup time in concerns will be investigated. Setup time for the processing of any parallel task on a hypercube system consist of the arrangement time, such as constructing a subcube structure, memory arrangement, loading parallel task. That is, if task T_i is a task prior to task T_j, then the start time of processing task T_j is not earlier than the finish time of task T_i. In this paper, the setup time will be considered as a constant. This problem is *NP-hard*, because scheduling independent nonpreemptive parallel tasks without considering setup time, a special case of this problem, was already known to be *NP-hard* [6]. The rest of this paper is organized as follows. In Section 2, a maximum-parallel- dimension procedure, which is used to decide the scheduled parallelism of each parallel task, is proposed and the worst performance bound of the *Largest Scheduled Dimension Firsti* (LSDF) algorithm using the above maximum-parallel-dimension procedure is analyzed. Section 3 presents another parallel dimension deciding procedure, called the co-relation procedure, and derived the worst performance bound of the LSDF algorithm using this co-relation procedure. Finally, some concluding remarks are given in Section 4.

2 Maximum Parallel Dimension Policy

Consider a set of n independent parallel tasks $\mathbf{T} = \{T_1, T_2, ..., T_n\}$ to be processed on a d-dimensional hypercube in which each task T_i is associated with a maximum dimension of parallelism Δ_i which is similar to the maximum degree of parallelism but must be power of 2, and a computation requirement t_i. A task T_i may be processed on up to a Δ_i-dimensional subcube and this parallelism dimension, once decided for T_i, will not be altered during its processing. If a task T_i is scheduled to run on a d_i-dimensional subcube, for $0 \leq d_i \leq \Delta_i \leq d$, then d_i is called the scheduled dimension of T_i. For this problem type, a schedule is feasible if the scheduled dimension d_i of each task T_i is no greater than its maximum dimension of parallelism Δ_i. Under the ideal assumption, previous parallel tasks and no overhead is required. That is, the setup time caused by switching from processing a task to processing another task needs not be included in their performance analysis. But in fact, the overhead of switching is high and must be taken into consideration.

In this paper, the setup time is assumed to be sequence independent [10, 11, 15] and constant [15], i.e., the time taken to switch a processor to a new task is independent of the task last processed and is a constant s. Thus, the total processing time $P(x, t_i)$ required for processing a task T_i is equal to $(\frac{t_i}{2^x} + s)$, where $1 \leq x \leq \Delta_i$. By simple calculus, we then have:
$\frac{\partial P(x,t_i)}{\partial x} = -(\frac{\ln 2}{2^x})t_i$ and $\frac{\partial^2 P(x,t_i)}{\partial x^2} = \frac{(\ln 2)^2}{2^x}t_i$.

Since $\frac{\partial P(x,t_i)}{\partial x} < 0$ and $\frac{\partial^2 P(x,t_i)}{\partial x^2} > 0$ for $1 \leq x \leq d$, $P(x, t_i)$ is a decreasing function which reaches its minimum value at Δ_i. Based on the above result, an intuitive strategy letting the scheduled dimension d_i of task T_i be assigned to its maximum dimension of parallelism Δ_i and we will have a minimum processing time $P(d_i, t_i) = (\frac{t_i}{2^{\Delta_i}} + s)$.

```
Procedure maximum-parallel-dimension
Begin
    Assign $\Delta_i$ to the scheduled dimension $d_i$ of
    each task $T_i$, for $i = 1, 2, ..., n$.
End.
```

With the above scheduled dimension d_i for each parallel task T_i, a nonpreemptive scheduling algorithm, called the *Largest Scheduled Dimension First* (LSDF) algorithm for the parallel tasks scheduling problem is proposed. The major policy of LSDF, similar to the Blazewicz's algorithm [3] or the LDF algorithm [17], is that the larger scheduled dimension a task has, the earlier it will be assigned.

```
Algorithm LSDF
Begin
    Call the maximum-parallel-dimension procedure;
    Arrange parallel tasks in nonincreasing order
    of their scheduled dimension;
    Assign parallel tasks to subcubes according
    to this nonincreasing order;
End.
```

Lemma 1: [8, 9] The number of free processors in a hypercube system will always be a multiple of the number of processors required by a task which is the next to be scheduled in the LSDF algorithm.

Lemma 2: [8, 9] If a task T_i, $1 \leq i \leq n$, is finished at time S_H, then there will be no processor idle before $(S_H - P(d_i, t_i))$, where S_H and t_i denote the finish time of a task set **T** scheduled by LSDF and the computation requirement of task T_i, respectively.

Theorem 1: $S_H \leq (2 - 1/2^d)S_O + (1 - 1/2^d)ns$, where S_H and S_O denote the finish times of the LSDF using the maximum-parallel-dimension procedure schedule and that of an optimal schedule of **T**, respectively.

Proof: By Lemma 1 and Lemma 2, we can get:

$$(2^d \times S_H) = \sum_{i=1}^{n}[2^{\Delta_i} \times P(\Delta_i, t_i)] + (2^d - 2^{\Delta_j})P(\Delta_j, t_j)$$

$$(2^d \times S_H) = \sum_{i=1}^{n}[2^{\Delta_i}(\frac{t_i}{2^{\Delta_i}} + s)] + (2^d - 2^{\Delta_j})(\frac{t_j}{2^{\Delta_j}} + s)$$

$$S_H \leq \frac{\sum_{i=1}^{n}[2^{\Delta_i}(\frac{t_i}{2^{\Delta_i}} + s)] + (2^d - 1)(\frac{t_j}{2^{\Delta_j}} + s)}{2^d}$$

It is obvious that $\frac{\sum_{i=1}^{n}(t_i+s)}{2^d} \leq S_O$ and $(\frac{t_j}{2^{\Delta_j}} + s) \leq S_O$. We then have:

$$S_H \leq (2 - \frac{1}{2^d})S_O + \frac{\sum_{i=1}^{n}(2^{\Delta_i} - 1)s}{2^d}$$

$$S_H \leq (2 - \frac{1}{2^d})S_O + (1 - \frac{1}{2^d})ns$$

The performance bound derived in Theorem 1 is not tight, in the following, an example will be given to show that the worst case performance bound of LSDF using the maximum-parallel-dimension procedure is at least $(2 - \frac{1}{2^d})S_O + (2^d - 2 - \frac{1}{2^d})s$.

Example 1: Let $t_i = t$ and $\Delta_i = d$ for $i = 1, 2, ..., 2^d - 1$, and $t_{2^d} = t$ and $\Delta_{2^d} = 1$. Figs. 1 (a) and (b) illustrate that the finish times of an LSDF using the maximum- parallel-dimension procedure schedule and that of an optimal schedule are $(2t + 2^d s - \frac{t}{2^d})$ and $(t + s)$, respectively. Thus, $S_H = (2 - \frac{1}{2^d})S_O + (2^d - 2 - \frac{1}{2^d})s = (2 - \frac{1}{2^d})S_O + (1 - \frac{2}{2^d} - \frac{1}{2^{2d}})ns \approx (2 - \frac{1}{2^d})S_O + (1 - \frac{1}{2^d})ns$, where $n = 2^d$.

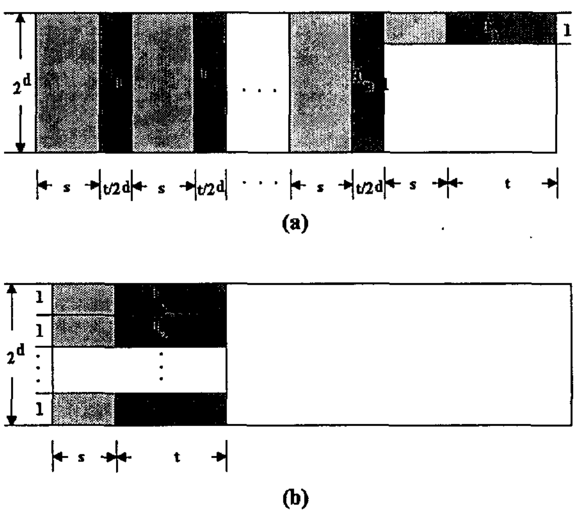


Figure 1: (a) An LSPF schedule using the maximum-parallel-dimension procedure and (b) an optimal schedule.

3 Co-relation Policy

The performance of a task T_i is usually defined in term of a *speedup factor* $S(d_i, t_i)$ and an *efficiency factor* $E(d_i, t_i)$ as follows.

$$S(d_i, t_i) = \frac{P(0, t_i)}{P(d_i, t_i)} = \frac{t_i + s}{(\frac{t_i}{2^{d_i}}) + s} \quad (1)$$

$$E(d_i, t_i) = \frac{S(d_i, t_i)}{2^{d_i}} = \frac{t_i + s}{t_i + (2^{d_i})s} \quad (2)$$

From equations (1) and (2), we can observe that the larger dimension of a subcube allocated to a task, the higher speedup of a task will be obtained but the less efficiency will also be incurred. There will be a need to make a trade-off between speedup and efficiency of a task. It should be pointed out that the required setup time s also plays an important role in equations (1) and (2). Note that the speedup and the efficiency of task T_i will be dominated by the setup time s and little affected by the scheduled dimension d_i of task T_i if the setup time is quite large. On the other hand, the speedup and the efficiency of task T_i will have little influence on the setup time s and much on the scheduled dimension d_i of task T_i if the setup time s is small enough. Therefore, either the required setup time s or the scheduled dimension d_i of task T_i have their respective effect to the speedup and the efficiency of task T_i. To determine the scheduled dimension d_i of a task T_i, our strategy is to find the largest possible dimension x_i of T_i such that the total reduction of time obtained from processing T_i on an $(x_i + 1)$-dimension subcube and on an x_i dimension subcube should be less than k times of the required setup time s, where k is an arbitrarily chosen positive constant. The smaller k is chosen, the more speed and less efficiency a task T_i will have. From the above strategy, we have:

$$\begin{aligned} & P(x_i, t_i) - P(x_i + 1, t_i) < ks \\ & \Rightarrow [(\frac{t_i}{2^{x_i}} + s) - (\frac{t_i}{2^{x_i+1}} + s)] < ks \\ & \Rightarrow (\frac{t_i}{2^{x_i+1}}) < ks \end{aligned} \quad (3)$$

<u>**Procedure co-relation**</u>
Begin

Let the scheduled dimension d_i of task T_i be $\max\{0, \min\{\lfloor \log_2(\frac{t_i}{ks}) - 1 \rfloor, \Delta_i\}\}$, for $i = 1, 2, ..., n$.

End.

In application, we have simply to replace the statement "Call the maximum- parallel-dimension procedure;" in LSDF algorithm by the statement "Call the co-relation procedure;".

Lemma 3: $P(d_j, t_j) < (2k + 1)P(\Delta_j, t_j)$, where d_i is the scheduled dimension of a task T_i determined by the co-relation procedure.

Proof: $P(d_i, t_i) = (\frac{t_j}{2^{d_j}} + s)$

$$\Rightarrow P(d_i, t_i) = (\frac{\frac{t_j}{2^{d_j}} + s}{\frac{t_j}{2^{\Delta_j}} + s})(\frac{t_j}{2^{\Delta_j}} + s)$$

Consider the following two conditions about $\frac{t_j}{2^{d_j}}$.

(i) If $0 \le d_i < \Delta_i$, then according to the equation (3), we know that $\frac{t_i}{2^{d_i+1}} < ks$. Then,

$$P(d_i,t_i) \le (\frac{2ks+s}{\frac{t_j}{2^{\Delta_j}}+s})(\frac{t_j}{2^{\Delta_j}}+s)$$

$$\Rightarrow P(d_i,t_i) \le (2k+1)(\frac{t_j}{2^{\Delta_j}}+s)$$

That is,

$$P(d_i,t_i) \le (2k+1)P(\Delta_j,t_j) \qquad (4)$$

(ii) If $d_i = \Delta_i$, then it is obvious that

$$P(d_i,t_i) = P(\Delta_j,t_j) \qquad (5)$$

From the above two conditions (4) and (5), we conclude that $P(d_i,t_i) \le (2k+1)P(\Delta_j,t_j)$.

Theorem 2: The worst case analysis of LSDF using the co-relation procedure is bounded by $(2+2k+\frac{1}{k})-(\frac{2k+1}{2^d})$.

Proof: By Lemma 1 and Lemma 2, we can get:

$$(2^d\times S_H) = \sum_{i=1}^{n}[2^{d_i}\times P(d_i,t_i)]+(2^d-2^{d_j})P(d_j,t_j)$$

$$\Rightarrow (2^d\times S_H) = \sum_{i=1}^{n}[2^{d_i}(\frac{t_i}{2^{d_i}}+s)]+(2^d-2^{d_j})P(d_j,t_j)$$

$$\Rightarrow S_H \le \frac{\sum_{i=1}^{n}[2^{d_i}(\frac{t_i}{2^{d_i}}+s)]+(2^d-1)P(d_j,t_j)}{2^d}$$

By Lemma 3, we have

$$\begin{aligned} S_H &\le \frac{\sum_{i=1}^{n}(t_i+s)}{2^d}+\frac{\sum_{i=1}^{n}(2^{d_i}-1)s}{2^d} \\ &\quad +(1-\frac{1}{2^d})(2k+1)P(\Delta_j,t_j) \\ \Rightarrow S_H &\le \frac{\sum_{i=1}^{n}(t_i+s)}{2^d}+\frac{\sum_{i=1}^{n}[(2^{d_i}-1)(\frac{t_i}{k\times 2^{d_i}})]}{2^d} \\ &\quad +(1-\frac{1}{2^d})P(d_j,t_j), \end{aligned}$$

because $ks \le \frac{t_i}{2^{d_i}} < 2ks$.

It is obvious that $\frac{\sum_{i=1}^{n}(t_i+s)}{2^d} \le S_O$ and $P(\Delta_j,t_j) \le S_O$, we have:

$$S_H \le (1+\frac{1}{k})S_O+(1-\frac{1}{2^d})(2k+1)S_O$$

$$\Rightarrow \frac{S_H}{S_O} \le (2+2k+\frac{1}{k})-(\frac{2k+1}{2^d})$$

Corollary 1: The least upper bound of the derived performance bound of the LSDF using the co-relation procedure is bounded by $[2+2(2-\frac{2}{2^d})^{\frac{-1}{2}}+(2-\frac{2}{2^d})^{\frac{1}{2}}-\frac{2(2-\frac{2}{2^d})^{\frac{-1}{2}}+1}{2^d}]$.

Proof: Let $F(k) = (2+2k+\frac{1}{k})-(\frac{2k+1}{2^d})$. By simple calculus, we have:
$F'(k) = 2-\frac{1}{k^2}-\frac{2}{2^d}$ and $F''(k) = \frac{2}{k^3}$.

Since $F'(k) = 0$ when $k = (2-\frac{2}{2^d})^{\frac{-1}{2}}$ and $F''(k) > 0$.

Then, $F(k)$ reaches its minimum value at $k = (2-\frac{2}{2^d})^{\frac{-1}{2}}$.

Thus, we have:

$$\begin{aligned} F(k) \ \ge\ & [2+2(2-\frac{2}{2^d})^{\frac{-1}{2}}+(2-\frac{2}{2^d})^{\frac{-1}{2}} \\ & -\frac{2(2-\frac{2}{2^d})^{\frac{-1}{2}}+1}{2^d}]. \end{aligned}$$

The above derived performance bound of LSDF using the co-relation procedure is not tight.

Corollary 2: The behaviors of the co-relation procedure and the maximum- parallel-dimension procedure will be the same, if the chosen constant k is rarely small, say $k < \frac{\tau}{2^{d+1}}$, where $\tau = \min\{t_i | i = 1,2,...,n\}$.

A special case of the above problem occurs when the setup time is negligible. In such a case, the scheduled dimension of each task T_i determined by the above two procedures, the maximum-parallel-dimension procedure and the co-relation procedure, will be the same. The worst performance bound of LSPF will be reduced to $(2-\frac{1}{2^d})$ no matter what the maximum-parallel-dimension procedure or the co-relation procedure is used.

Corollary 3: In case that the setup time is negligible, say $s = \varepsilon/n$, the worst performance bound of LSPF will approach to $(2-\frac{1}{2^d})$.

4 Conclusions

In this paper, we have investigated an LSDF algorithm for the problem of nonpreemptively scheduling independent parallel tasks with setup time.

We showed that each task is processed with its maximum dimension is not always the best choice and derived that $S_H \leq [(2 - \frac{1}{2^d})S_O + (1 - \frac{1}{2^d})ns]$, where S_H, S_O and n are the finish time of the LSDF using the maximum-parallel-dimension procedure schedule, the finish time of an optimal schedule, and the number of tasks, respectively. Thus, another scheduled dimension decision procedure, the co-relation procedure, is proposed and showed that the performance bound of the LSDF using this procedure is $[(2+2k+\frac{1}{k}) - \frac{2k+1}{2^d}]$, but this derived performance bound is not tight. The behaviors of the maximum-parallel- dimension procedure and the co-relation procedure will be identical if k is very small. Besides, in case that the setup time can be ignored, the above two scheduled dimension decision procedures will be the same and a tight performance bound of $(2 - \frac{1}{2^d})$ can also be derived.

References

[1] Baker, R. K., "Introduction to Sequencing and Scheduling," New York: Wiley, 1974.

[2] Blazewicz, J., Drabowski, M., and Weglarz, J., "Scheduling Independent 2-processor Tasks to Minimize Schedule Length," *Information Processing Letter*, vol. 18, no. 5, pp. 267-273, 1984.

[3] Blazewicz, J., Drabowski, M., and Weglarz, J., "Scheduling Multiprocessor Tasks to Minimize Schedule Length," *IEEE Trans. on Computers*, vol. C-35, no. 5, pp. 389-393, 1986.

[4] Chen, G. I. and Lai, T. H., "Scheduling Independent Jobs on Hypercubes," *Proc. 5th Symp. Theoretical Aspects of Computer Science*, pp. 273–280, 1988.

[5] Chen, G. I. and Lai, T. H., " Preemptive Scheduling of Independent Jobs on a Hypercube," *Information Processing Letters*, 28, vol. 28, no. 4, pp. 201- 206, 1988.

[6] Du, J. and Leung, J. Y., "Complexity of Scheduling Parallel Task Systems," *SIAM J. Discrete Math.*, vol. 2, pp. 473–487, 1989.

[7] Graham, R. L., "Bounds on Multiprocessing Timing Anomalies," *SIAM J. Appl. Math.*, vol. 17, pp. 416–429, 1969.

[8] Lin, J. F. and Chen, S. J., "Scheduling Algorithm for Nonpreemptive Multiprocessor Tasks," *Computers and Mathematics with Applications*, vol. 28, no. 4, pp. 85–92, 1994.

[9] Lin, J. F. and Chen, S. J., "Scheduling Parallel Tasks on Hypercubes," *Electronics Letters*, vol. 30, no. 11, pp. 841–842, 1994.

[10] Monma, C. L. and Potts, C. N., "On the Complexity of Scheduling with Batch Setup Times," *Operations Researches*, vol. 37, pp. 798–804, 1989.

[11] Monma, C. L. and Potts, C. N., "Analysis of Heuristics for Preemptive Parallel Machine Scheduling with Batch Setup Times," *Operations Researches*, vol. 41, no. 5, pp. 981–992, 1993.

[12] Ullman, J. D., "NP-complete Scheduling Problem," *Journal of Computer and System Sciences*, vol. 10, pp. 384–393, 1975.

[13] Wang, Q. and Cheng, K. H., "List Scheduling of Parallel Tasks," *Information Processing Letters*, vol. 37, no. 5, pp. 291–297, 1991.

[14] Wang, Q. and Cheng, K. H., "A Heuristic of Scheduling Parallel Tasks and Its Analysis," *SIAM J. Comput.*, vol. 21, no. 2, pp. 281–294, 1992.

[15] Wonginger, G. J. and Yu, Z., " A heuristic for Preemptive Scheduling with Setup Times," *Computing*, vol. 49, pp. 151–158, 1992.

[16] Zhu, Y. and Ahuja, M., "An O(n log n) Feasibility Algorithm for Preemptive Scheduling of n Independent Jobs on a Hypercube," *Information Processing Letters*, vol. 35, no. 1, pp. 7–11, 1990.

[17] Zhu, Y. and Ahuja, M., "On Job Scheduling on a Hypercube," *IEEE Trans. on Computers*, vol. 4, no. 1, pp. 62–69, 1993.

HLO: A Higher-Order Deductive Object-Oriented Database Language

Mengchi Liu
Department of Computer Science
University of Regina
Regina, Saskatchewan
Canada S4S 0A2

Keywords: deductive databases, object-oriented databases, fixpoint semantics, least model semantics

Edited by: Xindong Wu

Received: February 13, 1995 **Revised:** June 19, 1995 **Accepted:** July 14, 1995

This paper proposes a higher-order deductive object-oriented database language called HLO. This language is capable of directly and naturally supporting object-oriented features such as object identity, complex objects, classes, and multiple inheritance. It supports the separation of schema and instance but treats schema as meta data so that schema and inheritance information can be represented and queried in the same way as ordinary data. The language is given a firm logical foundation in the form of the Herbrand-like semantics. The main novelty of the language is the incorporation of the well-definedness and well-typedness constraints in its semantics. A number of interesting semantic properties are investigated. The intended semantics of an HLO program is given by the least and justified model if it can be obtained by a bottom-up least fixpoint computation. The necessary condition to guarantee the existence of the intended semantics and the decidability of checking the necessary condition are also discussed.

1 Introduction

Deductive and object-oriented databases are two important extensions of the traditional database technology. Deductive databases extend the expressive power of the traditional databases by means of deduction and recursion. Languages of this kind include Datalog [8], LDL [23] and CORAL [24]. Object-oriented databases extend the modeling power of the traditional databases by using concepts such as object identity, complex objects, classes, and inheritance. Languages of this kind include Iris [11], Orion [15], O_2 [17]. The integration of deductive and object-oriented databases has received considerable attentions over the past few years. A number of deductive object-oriented database languages have been proposed to combine the best of the two approaches, such as COL [1], IQL [2, 4], LOGRES [7], O-logic [20], revised O-logic [14], C-Logic [9], OIL [25], F-logic [13], LLO [19], LOL [6], L&O [21], Datalog/method [3], and Gulog [10].

To deal with complex objects naturally and directly, proper notions are needed for schema and inheritance, which normally lead to higher-order logic [13]. Unfortunately, higher-order unification problem is undecidable [12]. Most of existing deductive object-oriented database languages make non-natural restrictions to avoid higher-order logic in some way. Languages such as COL, OIL, IQL, LOGRES separate schema from instance and use non-logical languages for schema and logical languages for instance. Languages such as F-logic, mix the notions of class and object non-naturally to get around high-order logic.

This paper proposes a novel typed deductive object-oriented database language called HLO, which stands for Higher-order Language for Objects. The rationale behind HLO is that certain kinds of higher-order features are natural requirements of deductive object-oriented databases and should be supported directly and naturally. The higher-order unification problem can be effectively handled by a bottom-up least fixpoint computation in which only ground substitutions are used as in [16].

HLO is capable of naturally and directly supporting object-oriented features such as object identity, complex objects, classes, and multiple inheritance. It supports the separation of schema and instance but treats schema as meta data so that schema and inheritance information can be represented and queried in the same way as ordinary data. The language is given a firm logical foundation in the form of the Herbrand-like semantics. The intended semantics of an HLO program is given by the least and justified model if it can be obtained by a bottom-up least fixpoint computation. The necessary condition to guarantee the existence of the intended semantics of a program and the decidability of checking the necessary condition are also discussed.

Compared with the related approaches, the main novelty of the language is the incorporation of the well-definedness and well-typedness constraints in its semantics. Because of these constraints, a number of well-known semantic properties held for traditional logic programs fail for HLO programs. We argue that these constraints are fundamental to the semantics of higher-order deductive object-oriented database languages.

This paper is organized as follows. Section 2 informally introduces the language by several motivating examples. Section 3 describes the formal syntax and Herbrand-like semantics of HLO. Section 4 discusses how to compare different interpretations and models and defines the least model. Section 5 focuses on the fixpoint semantics, and Section 6 summarizes and points out further research issues.

2 Informal Presentation

There are three different kinds of concepts in HLO which function at three different levels: a unique metaclass *class*, classes, and objects. The metaclass *class* denotes the set of classes. Classes denote sets of objects. One purpose of using three levels instead of two is to make the language as self-descriptive as possible, another is to be able to operate on classes and instances in a uniform fashion. For reasons of simplicity, we do not deal with metametaclasses here.

Instances of a concept must be declared using *instance-of atoms*. For example,

$$person : class$$
$$smith : person$$

specify that *person* is a class and *smith* is an object of the class *person*.

Concepts in HLO can have properties. Two kinds of properties are distinguished: *definitional* and *factual*. The terms were first introduced in TAXIS [22]. The metaclass *class* can only have definitional properties, classes can have definitional properties and factual properties, while objects can only have factual properties.

Definitional properties of a concept define attributes as partial mappings applicable to its instances, and the range of the attributes. If an attribute always maps an instance of the concept to an instance of the range, then it is called a *single-valued* attribute; if an attribute always maps an instance to a set of instances of the range, then it is called a *set-valued* attribute.

In HLO, *definitional atoms* are used to define definitional properties of concepts. For example,

$$person[spouse \Rightarrow person,$$
$$friends \Rrightarrow person]$$

defines a single-valued attribute *spouse* and a set-valued attribute *friends* of *person*.

Factual properties of a concept give the attribute values of the concept, and are specified using *factual atoms*. For example,

$$smith[spouse \rightarrow mary,$$
$$friends \twoheadrightarrow \{bob, pam\}]$$

specifies that the value of the *spouse* attribute of *smith* is *mary* and the value of the *friends* attribute of *smith* includes *bob* and *pam*. If nothing else about the *friends* attribute of *smith* exists, then the value of the *friends* attribute is the set consisting of *bob* and *pam*.

If the factual properties of a concept are constrained by the definitional properties of its class, that is, the attribute values of the concept are in the ranges of the attributes of its class, then they are said to be *well-typed*. For the above example, factual properties *spouse* and *friends* of *mary* are well-typed if *mary, bob* and *pam* are instances of the class *person*.

A class may have subclasses, and a subclass may have more than one superclass. Instances of a subclass are also instances of superclasses. A subclass (multiply) inherits definitional properties from its superclasses, but may refine some of the inherited properties and introduce extra definitional properties local to itself. In HLO, such subclass relationship between classes may be represented by a user-defined set-valued attribute *isa* as follows.

$$\begin{aligned}
&class[isa \Rightarrow\!\!\!\Rightarrow class]\\
&student : class\\
&\qquad [isa \twoheadrightarrow \{person\},\\
&\qquad\ takes \Rightarrow\!\!\!\Rightarrow course]\\
&employee : class\\
&\qquad [isa \twoheadrightarrow \{person\},\\
&\qquad\ salary \Rightarrow integer]\\
&workStudent : class\\
&\qquad [isa \twoheadrightarrow \{student, employee\}]
\end{aligned}$$

The inheritance may then be represented by the following higher-order rules:

$$\begin{aligned}
&O : D \Leftarrow C[isa \twoheadrightarrow \{D\}], O : C.\\
&C[A \Rightarrow Q] \Leftarrow C[isa \twoheadrightarrow \{D\}],\\
&\qquad\qquad\qquad D[A \Rightarrow Q]\\
&C[A \Rightarrow\!\!\!\Rightarrow Q] \Leftarrow C[isa \twoheadrightarrow \{D\}],\\
&\qquad\qquad\qquad\ D[A \Rightarrow\!\!\!\Rightarrow Q].
\end{aligned}$$

These rules say that if C is a subclass of D then every instance of C is also an instance of D and every definitional property of D is also a definitional property of C. Note that here $\{D\}$ does not mean a singleton set, instead it means a set of which the variable D stands for an element. In some way, it is similar to a set grouping variable of LDL [5], but can appear in the body of a rule. Based on the above examples, we can infer $workStudent[takes \Rightarrow\!\!\!\Rightarrow course, salary \Rightarrow integer]$. If we have $mary : student$, then we can infer $mary : person$.

Rules of HLO are used to insert concepts into its classes, to conditionally define definitional properties of a class or metaclass, or to infer factual properties of objects or classes. Following are several further examples.

$$\begin{aligned}
&mary : student\\
&person[likes \Rightarrow\!\!\!\Rightarrow person] \Leftarrow X : person\\
&mary[likes \twoheadrightarrow \{X\}] \Leftarrow\\
&\qquad\qquad\qquad smith[likes \twoheadrightarrow \{X\}\\
&X[ancestors \twoheadrightarrow \{Y\}] \Leftarrow\\
&\qquad\qquad\qquad X[parents \twoheadrightarrow \{Y\}]\\
&X[ancestors \twoheadrightarrow \{Y\}] \Leftarrow\\
&\qquad\qquad\qquad X[ancestors \twoheadrightarrow \{Z\}],\\
&\qquad\qquad\qquad Z[parents \twoheadrightarrow \{Y\}]\\
&X[descendants \twoheadrightarrow \{Y\}] \Leftarrow\\
&\qquad\qquad\qquad Y[ancestors \twoheadrightarrow \{X\}]
\end{aligned}$$

The first rule inserts object *mary* into class *student*. The second says if there exists an instance in the class *person*, then *person* has a set-valued attribute *likes*. The third says *mary* likes everyone that *smith* likes. The next two rules define the traditional *ancestor* relation. The last rule defines the reverse relation of *ancestor*.

Given a program of HLO which is just a set of rules, queries can be used to ask for the information about objects, classes, metaclass and their properties. Following are several examples.

$$\begin{aligned}
&?\!-\ X : person\\
&?\!-\ mary[likes \twoheadrightarrow \{X\}]\\
&?\!-\ smith : C[ancestors \twoheadrightarrow \{X\}]\\
&?\!-\ C[isa \twoheadrightarrow \{person\}]\\
&?\!-\ student[A_1 \Rightarrow C_1, A_2 \Rightarrow\!\!\!\Rightarrow C_2]\\
&?\!-\ C : class[A \twoheadrightarrow \{D\}]
\end{aligned}$$

The first query asks about the instances of the class *person*. The second requests people that *mary* likes. The third inquires about the classes as well as the ancestors of *smith*. The fourth asks about the subclasses of *person*. The fifth requests single-valued and set-valued definitional properties of the class *student*. The last inquires about the classes and their set-valued factual properties.

3 Formal Presentation

This section defines the formal syntax and semantics of the HLO language. The syntax is concerned with valid programs and queries admitted by the grammar of HLO. The semantics is concerned with the meanings attached to the valid programs and the symbols they contain.

3.1 Syntax

This subsection introduces the formal syntax of the HLO language, i.e., its alphabet, terms, atoms, rules, programs, and queries.

Definition 1 The *alphabet* of HLO consists of the following pairwise disjoint sets of symbols:

(1) a possibly infinite set $\mathcal{O}$ of object symbols,
(2) a possibly infinite set $\mathcal{C}$ of class symbols,
(3) a singleton set $\mathcal{M}$ containing the metaclass symbol *class*,
(4) a possibly infinite set $\mathcal{A}$ of attribute symbols,
(5) an infinite set $\mathcal{V}_\mathcal{O}$ of object variables,
(6) an infinite set $\mathcal{V}_\mathcal{C}$ of class variables,
(7) an infinite set $\mathcal{V}_\mathcal{A}$ of attribute variables,
(8) a set of improper symbols such as parentheses, arrows, etc.

Definition 2 The *terms* are defined as follows:

(1) an object symbols from $\mathcal{O}$ or an object variable from $\mathcal{V}_\mathcal{O}$ is an *object term*,
(2) a class symbol from $\mathcal{C}$ or a class variable from $\mathcal{V}_\mathcal{C}$ is a *class term*,
(3) *class* is a *metaclass term*,
(4) an attribute symbol from $\mathcal{A}$ or an attribute variable from $\mathcal{V}_\mathcal{A}$ is an *attribute term*.

A term is *ground* if it has no variables. An object is a ground object term. A class is a ground class term. An attribute is a ground attribute term.

Definition 3 The *atoms* are defined as follows:

(1) Let P be a class term and Q an object term. Then $class : class$, $P : class$ and $Q : P$ are *simple* atoms, called *instance-of* atoms.
(2) Let P and Q both be class terms or both be metaclass terms, and A be an attribute term. Then $P[A \Rightarrow Q]$ and $P[A \Rrightarrow Q]$ are *simple* atoms, called *definitional* atoms.
(3) Let P, Q, Q_1, ..., Q_n all be object terms, or all be class terms, and A be an attribute term. Then $P[A \rightarrow Q]$ and $P[A \twoheadrightarrow \{Q_1, ..., Q_n\}]$ are *simple* atoms, called *factual* atoms.
(4) Let $P : Q$, $P[A_1 \rightarrow P_1]$, ..., $P[A_l \twoheadrightarrow \{Q_{l,1}, ..., Q_{l,l_k}\}]$, ..., $P[A_m \Rightarrow P_m]$, ..., $P[A_n \Rrightarrow P_n]$, ..., be simple atoms, where $0 \leq l \leq m \leq n$. Then $P[A_1 \rightarrow P_1, ..., A_l \twoheadrightarrow \{Q_{l,1}, ..., Q_{l,l_k}\}, ..., A_m \Rightarrow P_m, ..., A_n \Rrightarrow P_n, ...]$ and $P : Q[A_1 \rightarrow P_1, ..., A_l \twoheadrightarrow \{Q_{l,1} ..., Q_{l,l_k}\}, ..., A_m \Rightarrow P_m, ..., A_n \Rrightarrow P_n, ...]$ are *composite* atoms, while $P : Q$, $P[A_1 \rightarrow P_1]$, ..., $P[A_l \twoheadrightarrow \{Q_{l,1} ..., Q_{l,l_k}\}]$, ..., $P[A_m \Rightarrow P_m]$, ..., $P[A_n \Rrightarrow P_n]$, ..., are called *constituent* atoms of the corresponding composite atoms.

An atom is *ground* if it has no variables. For a ground definitional atom $c[f \Rightarrow d]$, f is called a *single-valued* attribute of c. For a ground definitional atom $c[s \Rrightarrow d]$, s is called a *set-valued* attribute of c. Syntactically and semantically, it is allowed for an attribute symbol to be both single-valued and set-valued. But the specific arrows uniquely determine its kind in the context.

Definition 4 A *rule* is of the form

$$A \Leftarrow A_1, ..., A_n$$

where A, $A_1, .., A_n, n \geq 0$ are atoms. A is called the *head*, and $A_1, ..., A_n$ is called the *body* of the rule.

A rule is *safe* if all variables in the head are also in the body. A *fact* is a rule with empty body.

Definition 5 A *program* P is a finite set of safe rules.

Definition 6 A *query* is a sequence of atoms starting with ?-.

We can view classes as unary relations and attributes as binary relations of mathematical logic. Since class and attribute variables are used in the language, it is therefore a higher-order language.

3.2 Semantics

This subsection defines the Herbrand-like interpretations and models. These Herbrand-like interpretations bear important properties of the traditional Herbrand interpretations.

Definition 7 The *universe* U of HLO is defined as follows:

$$U_O = \mathcal{O},$$
$$U_C = \mathcal{C},$$
$$U_M = \{class\},$$
$$U_A = \mathcal{A},$$
$$U = U_O \cup U_C \cup U_M \cup U_A.$$

Definition 8 The *Herbrand base* B of HLO is the set of all ground simple atoms formed from object, class, metaclass and attribute symbols in the universe.

Note that the Herbrand base does not contain any composite atoms.

Definition 9 Let S be a subset of B. A ground instance-of atom $p : q$ is *well-defined* in S if both p and q are *class* or $q : class \in S$. A ground definitional atom $u[f \Rightarrow v]$ or $u[s \Rrightarrow v]$ is *well-defined* in S if both $u : class \in S$ and $v : class \in S$.

Definition 10 Let S be a subset of B. A ground factual atom $p[f \rightarrow q]$ is *well-typed* in S if there exist u and v such that $p : u \in S$, $u[f \Rightarrow v] \in S$ are well-defined in S, and for all such u and v, $q : v \in S$. A ground factual atom $u[s \twoheadrightarrow q]$ is *well-typed* in S if there exist u and v such that $p : u \in S, u[s \Rrightarrow v] \in S$, and for all such u and v, $q_i : v \in S$ for each $q_i \in q$.

By requiring well-typedness, we can therefore use the definitional properties of classes to constrain the factual properties of their instances. However, the well-typedness constraint cannot guarantee that attributes are treated as mappings in S since an object may have more than one different value for an attribute.

Definition 11 A subset of B is *consistent* if it does not contain a pair of factual object atom $p[a \rightarrow q_1]$ and $p[a \rightarrow q_2]$, or $p[a \twoheadrightarrow q_1]$ and $p[a \twoheadrightarrow q_2]$ such that $q_1 \neq q_2$.

Definition 12 An *interpretation* I of a program is a consistent subset of B.

Definition 13 A *ground substitution* θ is a finite mapping from the variables to U. It is extended to terms, set of terms and atoms as follows:

(1) if $p \in \mathcal{O} \cup \mathcal{C} \cup \mathcal{M} \cup \mathcal{A}$, then $p\theta = p$,
(2) $\{X_1, ..., X_n\}\theta = \{X_1\theta, ..., X_n\theta\}$,
(3) if E is an atom, then $E\theta$ results from E by applying θ to every term, set of terms in E.

Definition 14 Given in interpretation I, the notion of satisfaction, denoted $\models$, is defined as follows:

(1) For each ground instance-of atom $p : q$, $I \models p : q$ iff $p : q$ is well-defined in I and $p : q \in I$.
(2) For each ground definitional atom ψ of the form $p[a \Rightarrow q]$ or $p[a \Rrightarrow q]$, $I \models \psi$ iff ψ is well-defined in I and $\psi \in I$.
(3) For each ground factual atom $p[f \rightarrow q]$, $I \models p[f \rightarrow q]$, iff $p[f \rightarrow q]$ is well-typed in I, and $p[f \rightarrow q] \in I$.
(4) For each ground factual atom $p[s \twoheadrightarrow q]$, $I \models p[s \twoheadrightarrow q]$, iff $p[s \twoheadrightarrow q]$ is well-typed in I and there exists a $p[s \twoheadrightarrow q'] \in I$ such that $q \subseteq q'$.
(5) For ground composite atoms ψ, $I \models \psi$ iff for each constituent atom φ of ψ, $I \models \varphi$
(6) Let r be a rule of the form $A \Leftarrow A_1, ..., A_n$. Then $I \models r$ iff for each ground substitution θ such that $I \models A_1\theta$,...,$I \models A_n\theta$ implies $I \models A\theta$.
(7) For each program P, $I \models P$ iff for each rule $r \in P$, $I \models r$.

Definition 15 A *model* M of P is an interpretation of P which satisfies P.

Example 1 Consider the following program:

$class : class[isa \Rrightarrow class]$
$person : class$
$student : class[isa \twoheadrightarrow \{person\}]$
$mary : student$
$O : D \Leftarrow O : C, C[isa \twoheadrightarrow \{D\}]$

An interpretation I_0 for this program is

$$\begin{aligned} I_0 = \{&class : class,\\ &class[isa \Rrightarrow class],\\ &person : class,\\ &student : class,\\ &student[isa \twoheadrightarrow \{person\}],\\ &mary : student,\\ &mary : person\} \end{aligned}$$

Indeed, the interpretation I_0 is a model of the program by the definition. The following interpretations I_1 and I_2 are two additional models:

$$\begin{aligned} I_1 = \{&class : class,\\ &class[isa \Rrightarrow class],\\ &person : class,\\ &person[parents \Rrightarrow person],\\ &student : class,\\ &student[isa \twoheadrightarrow \{person\}],\\ &mary : student,\\ &mary : person,\\ &smith : person\} \end{aligned}$$

$$\begin{aligned} I_2 = \{&class : class,\\ &class[isa \Rrightarrow class],\\ &person : class,\\ &student : class, \end{aligned}$$

$student[isa \twoheadrightarrow \{person\}],$
$student[classmates \Rrightarrow student],$
$mary : student,$
$mary : person,$
$john : student,$
$john : person,$
$john[classmates \twoheadrightarrow \{mary\}]\}$

In I_1, *person* has an extra definitional property *parents* and an extra instance *smith*, whereas in I_2, *student* has an extra definitional property *classmates* and an extra instance *john* with factual property value $\{mary\}$.

As above example shows, a program may have many different interpretations and models, which means that it can be interpreted differently. What is the exact semantics of a program then? We discuss this issue in the following sections.

4 Comparing Interpretations

In this section, we study how different interpretations and models of a program can be compared. We first introduce the following notion.

Definition 16 The *sub-atom* relation between ground simple atoms of B, denoted $\preceq$, is defined as follows.

(1) $p : q \preceq p : q$,
(2) $p[a \Rightarrow q] \preceq p[a \Rightarrow q]$,
(3) $p[a \Rrightarrow q] \preceq p[a \Rrightarrow q]$,
(4) $p[a \rightarrow q] \preceq p[a \rightarrow q]$,
(5) $p[s \twoheadrightarrow q_1] \preceq p[s \twoheadrightarrow q_2]$, if $q_1 \subseteq q_2$.

Proposition 1 The sub-atom relation is a partial order.

Proof. Direct from the definition □

Definition 17 Let P be a program, I_1 and I_2 two interpretations of P. Then I_1 is a *sub-interpretation* of I_2, denoted $I_1 \sqsubseteq I_2$, iff for every atom $\psi_1 \in I_1$, there exists an atom $\psi_2 \in I_2$ such that $\psi_1 \preceq \psi_2$.

For the interpretations I_0, I_1, I_2 of Example 1 in the last section, we have $I_0 \sqsubseteq I_1$, $I_0 \sqsubseteq I_2$ but neither $I_1 \sqsubseteq I_2$ nor $I_2 \sqsubseteq I_1$.

Proposition 2 The sub-interpretation relation between interpretations of a program is a partial order.

Proof. Direct from the definition and Proposition 1. □

Definition 18 A model M of P is the *least* iff for each model N of P, $M \sqsubseteq N$.

Definition 19 Let P be a program and I_1 and I_2 be two interpretations of P. The *intersection* I of I_1 and I_2, denoted $I = I_1 \sqcap I_2$, is defined as follows:

(1) if $p : q \in I_1$ and $p : q \in I_2$, then $p : q \in I$,
(2) if $p[a \Rightarrow q] \in I_1$ and $p[a \Rightarrow q] \in I_2$, then $p[a \Rightarrow q] \in I$,
(3) if $p[a \Rrightarrow q] \in I_1$ and $p[a \Rrightarrow q] \in I_2$, then $p[a \Rrightarrow q] \in I$,
(4) if $p[f \rightarrow q] \in I_1$ and $p[f \rightarrow q] \in I_2$, then $p[f \rightarrow q] \in I$,
(5) if $p[s \twoheadrightarrow q_1] \in I_1$ and $p[s \twoheadrightarrow q_2] \in I_2$, then $p[s \twoheadrightarrow q] \in I$ where $q = q_1 \cap q_2$.

Note here sub-interpretation relation is not just a simple subset relation. Similarly, interpretation intersection is not just a simple set intersection. Consider again the interpretations I_0, I_1, I_2 of Example 1 in the last section, we have $I_0 = I_1 \sqcap I_2$ based on the above definition.

The intersection of interpretations of a program has the following properties.

Proposition 3 Let I_1 and I_2 be two interpretations of P and $I = I_1 \sqcap I_2$. Then

(1) I is an interpretation.
(2) $I \sqsubseteq I_1$ and $I \sqsubseteq I_2$.

Proof. (1) Since I_1 and I_2 are consistent, I is therefore consistent.

(2) Direct from the definition. □

Proposition 4 The relation $\sqcap$ over interpretations of a given program is commutative, idempotent and associative, i.e., $I_1 \sqcap I_2 = I_2 \sqcap I_1$, $I_1 \sqcap I_1 = I_1$, and $I_1 \sqcap (I_2 \sqcap I_3) = (I_1 \sqcap I_2) \sqcap I_3$ for any interpretations I_1, I_2 and I_3.

Proof. Direct from the definition. □

Proposition 5 Let P be a program and $\{M_i\}_{i \in N}$ a non-empty set of all models of P. If the intersection $M_P = \sqcap_{i \in N} M_i$ is also a model for P, then M_P is the least model.

Proof. Based on Propositions 3 and 4, M_P is a sub-interpretation of every model of P. Since it is also a model of P, it is the least model of P. □

For the interpretations I_0, I_1, I_2 of Example 1, the least model is I_0.

It is well-known that every Horn clause program has models and the intersection of all models is also a model [18]. This is not necessarily true for an HLO program. An HLO program may not have models due to the consistency constraint, see Example 10 in the next section. Even if it has models, the model intersection property does not hold due to the well-definedness and well-typedness constraints.

Example 2 Look at the following program with only one fact:

$$mary[spouse \rightarrow smith]$$

The following interpretations I_1 and I_2 are models of the program:

$$\begin{aligned} I_1 = \{&person : class,\\ &person[spouse \Rightarrow person],\\ &mary : person,\\ &smith : person,\\ &mary[spouse \rightarrow smith]\}\\ I_2 = \{&employee : class,\\ &employee[spouse \Rightarrow employee],\\ &mary : employee,\\ &smith : employee,\\ &mary[spouse \rightarrow smith]\} \end{aligned}$$

Their intersection is $\{mary[spouse \rightarrow smith]\}$ which is not a model since the ground factual atom $mary[spouse \rightarrow smith]$ is not well-typed in I.

5 Least-Fixpoint Semantics

We proceed to show that if an HLO program has the least model and this model can be constructed by a bottom-up least fixpoint computation, then we can use it as the intended semantics of the program.

Definition 20 Let P be a program and I an interpretation, the operator T_P over I is defined as follows.

$$\begin{aligned} T_P(I) = \{\psi\theta \mid\ & A \Leftarrow A_1, ..., A_n \in P,\ \psi \text{ is a}\\ & \text{constituent atom of } A, \text{ there exists}\\ & \text{a ground substitution } \theta \text{ such that}\\ & I \models A_i\theta,\ 1 \le i \le n, \text{ and } \psi\theta \text{ is}\\ & \text{either well-typed in } I \text{ if } \psi \text{ is a}\\ & \text{factual atom or well-defined in } I\\ & \text{if } \psi \text{ is an instance-of atom or a}\\ & \text{definitional atom}\} \end{aligned}$$

Here T_P is similar to the traditional immediate consequence operator but it incorporates the well-definedness and well-typedness constraints. It turns out that several important properties held for traditional logic programs fail for HLO programs.

Definition 21 Let P be a program and S be a set of interpretations of P. The operator T_P is *monotonic* on S if for interpretations $I \in S$ and $J \in S$, $I \sqsubseteq J$ implies that $T_P(I) \subseteq T_P(J)$.

In general, T_P is not monotonic because of the well-typedness constraint.

Example 3 Consider the following program with just one fact and interpretations I and J.

$$john[spouse \rightarrow mary]$$

$$\begin{aligned} I = \{&class : class,\\ &person : class,\\ &person[spouse \Rightarrow person],\\ &john : person,\\ &mary : person\}\\ J = \{&class : class,\\ &person : class,\\ &employee : class,\\ &person[spouse \Rightarrow person],\\ &person[spouse \Rightarrow employee],\\ &john : person,\\ &mary : person\} \end{aligned}$$

Then $I \subseteq J$. But $T_P(I) = \{john[spouse \rightarrow mary]\} \not\subseteq T_P(J) = \emptyset$.

Besides, $T_P(I)$ may not be an interpretation.

Example 4 Consider the following program P:

$$\begin{aligned} &class : class\\ &person : class[friends \Rightarrow\!\!\!\!\Rightarrow person] \end{aligned}$$

$$mary[friends \twoheadrightarrow \{X\}] \Leftarrow smith[s \twoheadrightarrow \{X\}]$$

Let $I = \{class : class,$
$person : class,$
$person[friends \Rightarrow\!\!\!\!\Rightarrow person],$
$mary : person,$
$smith : person,$
$john : person,$
$bob : person,$
$smith[friends \twoheadrightarrow \{john, bob\}]\}$

be an interpretation. Then we have

$$\begin{aligned}T_P(I) = \{&class : class,\\ &person : class,\\ &person[friends \Rightarrow\!\!\!\!\Rightarrow person],\\ &mary[friends \twoheadrightarrow \{bob\}],\\ &mary[friends \twoheadrightarrow \{john\}]\}\end{aligned}$$

which is not consistent and therefore not an interpretation.

Definition 22 Let S be a set of ground simple atoms. The *compact* operator C is defined as a partial mapping from B to B as follows:

$$\begin{aligned}C(S) = &\{p : q \in S\} \cup\\ &\{p[a \Rightarrow q] \in S\} \cup\\ &\{p[a \Rightarrow\!\!\!\!\Rightarrow q] \in S\} \cup\\ &\{p[a \rightarrow q] \in S\} \cup\\ &\{p[s \twoheadrightarrow q] \mid q = \cup\{q_i \mid p[s \twoheadrightarrow q_i] \in S\}\}\end{aligned}$$

if this set is consistent; otherwise C is undefined.

Here C compacts a collection of set-valued simple atoms into a single set-valued simple atom and checks the consistency of single-valued simple atoms.

Continuing with the example above, we have

$$\begin{aligned}C(T_P(I)) = \{&class : class, person, class,\\ &person[friends \Rightarrow\!\!\!\!\Rightarrow person],\\ &mary[friends \twoheadrightarrow \{bob, john\}]\}\end{aligned}$$

which is an interpretation.

Proposition 6 The operator C has the following properties.

(1) if I is an interpretation, then $C(I) = I$;
(2) if $J \subseteq K$ and C is defined on both J and K, then $C(J) \sqsubseteq C(K)$.

Proof. Direct from the definition. □

A set S of simple atoms is *relevant to* an atom ψ iff for each $\psi' \in S$, $\psi' \preceq \psi$.

For example, $\{b : p\}$ and $\{b[s \rightarrow \{a\}], b[s \rightarrow \{b\}]\}$ are relevant to $b : p$ and $b[s \rightarrow \{a, b, c\}]$ respectively.

Definition 23 Let T_P and C be two operators as defined above and I an interpretation. Then I is a *prefixpoint* of T_P and C if $C(T_P(I)) \sqsubseteq I$; I is a *fixpoint* of T_P and C if $C(T_P(I)) = I$; and I is the *least fixpoint* of T_P and C iff for each fixpoint N of T_P and C, if $N \sqsubseteq I$, then $I = N$.

The following important property holds for HLO programs, which links the notion of model of P to that of prefixpoint of T_P and C.

Proposition 7 Let M be an interpretation of the program P. Then M is a model of P iff M is a prefixpoint of T_P and C.

Proof. Assume M is a model. Let $\psi \in C(T_P(M))$. If $\psi \in T_P(M)$, then there exists a rule $A \Leftarrow A_1, ..., A_n$ and a ground substitution θ such that $M \models A_1\theta, ..., M \models A_n\theta$, ψ is a ground constituent atom of $A\theta$ and ψ is either well-typed or well-defined. Because M is a model of P, $M \models \psi$. There exists a $\psi' \in M$ such that $\psi \preceq \psi'$. Thus, $C(T_P(M)) \sqsubseteq M$. If $\psi \notin T_P(M)$, then ψ is of the form $p[s \twoheadrightarrow q]$. There exists a subset S of $T_P(M)$ relevant to ψ such that ψ is the result of applying C to S. For each $p[s \twoheadrightarrow q_i] \in S$, there exist a rule $A \Leftarrow A_1, ..., A_n$ and a ground substitution θ such that $M \models A_1\theta, ..., M \models A_n\theta$ and $p[s \twoheadrightarrow q_i]$ is a ground constituent atom of $A\theta$. Since M is a model of P, $M \models p[s \twoheadrightarrow q_i]$. Then there exists a set q' such that $q_i \subseteq q'$ and $p[s \twoheadrightarrow q'] \in M$. Obviously, $q \subseteq q'$. Let ψ' be $p[s \twoheadrightarrow q']$. Then $\psi \preceq \psi'$. Again, $C(T_P(M)) \sqsubseteq M$.

Suppose $C(T_P(M)) \sqsubseteq M$. We prove M is a model. If M is not a model of P, then there exist a rule $A \Leftarrow A_1, ..., A_n$ and a ground substitution θ such that $M \models A_1\theta, ..., M \models A_n\theta$, but $M \not\models A\theta$. Then there exists a constituent atom ψ of A such that $M \not\models \psi\theta$. On the other hand, $\psi\theta \in T_P(M)$. Assume $\psi\theta \in C(T_P(M))$ as well. Since $C(T_P(M)) \sqsubseteq M$, there exists a $\psi' \in M$ such that $\psi\theta \preceq \psi'$. Thus $M \models \psi\theta$, a contradiction. Assume $\psi\theta \notin C(T_P(M))$. Then there exists a $\psi' \in C(T_P(M))$ such that $\psi\theta \preceq \psi'$. Since $C(T_P(M)) \sqsubseteq M$, there exists a $\psi'' \in M$ such that

$\psi' \preceq \psi''$. We have $M \models \psi'$ as well as $M \models \psi\theta$, also a contradiction. □

Due to the well-typedness and well-definedness constraints, the reverse of the above proposition is not necessarily true for HLO programs.

Example 5 Consider the following program:

$$person : class$$

We have $T_P(\emptyset) = \emptyset$ and thus $C(T_P(\emptyset)) = \emptyset \sqsubseteq \emptyset$. Clearly, $\emptyset$ is not a model of the program.

Definition 24 A model M of P is *justified* if for each $\psi \in M$, there exists a subset R of rules of P such that ψ is the result of applying the compact operator C to the following set:

$$\{\psi'\theta \mid A \Leftarrow A_1, ..., A_n \in R, \psi' \text{ is a constituent atom of } A, \text{ there exists a ground substitution } \theta \text{ such that } M \models A_i\theta, 1 \le i \le n, \text{ and } \psi\theta \preceq \psi\}$$

Proposition 8 A model M of P is justified iff $M \sqsubseteq C(T_P(M))$.

Proof. Suppose $M \sqsubseteq C(T_P(M))$: We prove M is justified. For each $\psi \in M$, there is a $\psi' \in C(T_P(M))$ such that $\psi \preceq \psi'$. Then there exist a subset R' of P and a corresponding set S' of ground atoms such that ψ' is the result of applying C to S'. Hence, there exist a subset R of R' and a corresponding subset S of S' such that ψ is the result of applying C to S. Therefore, M is justified.

Now Suppose M is justified. Let $\psi \in M$. By definition there exists a set S of ground atoms such that ψ is the result of applying C to S. For each $\varphi \in S$, $M \models \varphi$ since M is a model. So φ is well-typed or well-defined depending on whether or not it is a factual atom. Hence, $\varphi \in T_P(M)$. Therefore, there exists a $\psi' \in C(T_P(M))$ such that $\psi \preceq \psi'$. □

The next proposition relates a justified model to a fixpoint of T_P and C.

Proposition 9 Let P be a program and M a model of P. Then M is the least and justified model of P iff M is the least fixpoint of T_P and C.

Proof. Straightforward from Propositions 7 and 8. □

Note that the least model of a HLO program is not necessarily justified and a justified model is not necessarily the least.

Example 6 Consider the program of Example 5 again:

$$person : class$$

It has the least model

$$\{person : class, class : class\}$$

which is not justified since $class : class$ cannot be inferred from the program.

Example 7 Now consider the program that consists of the two rules:

$$class : class \Leftarrow person : person$$
$$person : class \Leftarrow class : class$$

It has a justified model $\{class : class, person : class\}$, but the least model is $\emptyset$.

Definition 25 The powers of the operator T_P are defined using the compact operator as follows:

$$T_P \uparrow 0 = \emptyset$$
$$T_P \uparrow n = C(T_P(T_P \uparrow n-1)) \text{ if } C \text{ is defined}$$
$$T_P \uparrow \omega = C(\cup_{n=0}^{\infty} T_P \uparrow n) \quad \text{ if } C \text{ is defined}$$

if C is not defined, then $T_P \uparrow n$ and $T_P \uparrow \omega$ are undefined.

Unlike traditional logic program, $T_P \uparrow \omega$ may not be a model of P.

Example 8 Consider the following program:

$$class : class$$
$$person : class[spouse \Rightarrow person]$$
$$employee : class$$
$$john : person[spouse \rightarrow mary]$$
$$mary : person$$
$$person[spouse \Rightarrow employee] \Leftarrow mary : person$$

The powers of T_P are computed as follows:

$$T_P \uparrow 0 = \emptyset$$
$$T_P \uparrow 1 = \{class : class\}$$
$$T_P \uparrow 2 = \{person : class, employee : class\} \cup T_P \uparrow 1$$
$$T_P \uparrow 3 = \{person[spouse \Rightarrow person], john : person,$$

$$\begin{aligned}
& \qquad mary : person\} \\
& \qquad \cup\, T_P \uparrow 2 \\
T_P \uparrow 4 &= \{person[spouse \Rightarrow employee], \\
& \qquad john[spouse \rightarrow mary]\} \\
& \qquad \cup\, T_P \uparrow 3 \\
T_P \uparrow 5 &= T_P \uparrow 4 \\
& \qquad - \{john[spouse \rightarrow mary]\} \\
T_P \uparrow \omega &= T_P \uparrow 4
\end{aligned}$$

However, $T_P \uparrow \omega$ is not a model of P since the fact $john[spouse \rightarrow mary]$ cannot be satisfied by it.

Proposition 10 Let P be a program. Suppose $T_P \uparrow \omega$ is a model of P. Then

(1) $T_P(T_P \uparrow i) \subseteq T_P(T_P \uparrow i+1)$ for $i \in N$.
(2) T_P is monotonic on $\{I\}_{I \sqsubseteq T_P \uparrow \omega}$.
(3) For any model N of P, $T_P(T_P \uparrow i) \subseteq T_P(N)$ for natural number i.

Proof. (1) Let $\psi \in T_P(T_P \uparrow i) - T_P(T_P \uparrow i+1)$. Then there exists a rule $A \Leftarrow A_1, ..., A_n$ and a ground substitution θ such that $T_P \uparrow i \models A_1\theta$, ..., $T_P \uparrow i \models A_n\theta$ and ψ is a constituent atom of $A\theta$. Since $\psi \notin T_P(T_P \uparrow i+1)$, either there exists A_i for some i such that $T_P \uparrow i+1 \not\models A_i\theta$ or ψ is neither well-defined nor well-typed in $T_P \uparrow i+1$.

For the first case, let $\psi_1 \doteq A_i\theta$. Then $\psi_1 \in T_P(T_P \uparrow i) - T_P(T_P \uparrow i+1)$. By repeating the above process we can find a sequence ψ, ψ_1, ..., ψ_n for some finite n such that ψ_n is a ground fact and is neither well-defined nor well-typed in $T_P \uparrow i+1$ which lead to the second case.

For the second case, there exists a $\psi' \in T_P \uparrow \omega$ such that $\psi \preceq \psi'$. Therefore ψ' is neither well-defined nor well-typed in $T_P \uparrow \omega$. This means that there exists a rule which cannot be satisfied by $T_P \uparrow \omega$, a contradiction.

(2) Direct from (1) and Proposition 6 (2).
(3) Straightforward from (1). □

Now we present one of the major results of the theory.

Theorem 1 Let P be a program. Suppose $T_P \uparrow \omega$ is a model of P. Then

(1) Every model of P contains $T_P \uparrow \omega$, i.e., $T_P \uparrow \omega$ is the least model of P.
(2) $T_P \uparrow \omega$ is a justified model of P.

Proof. (1) We first prove $T_P \uparrow \omega$ is least. Let N be a model of P. We prove by induction on i that

$$T_P \uparrow i \sqsubseteq N \qquad (1)$$

The basis is clearly true. Suppose the claim holds for $i \geq 0$. By Proposition 10 (3), we have $T_P(T_P \uparrow i) \subseteq T_P(N)$. By Proposition 6 (2), we have $C(T_P(T_P \uparrow i)) \sqsubseteq C(T_P(N))$. By Proposition 7, $T_P \uparrow i+1 = C(T_P(T_P \uparrow i)) \sqsubseteq C(T_P(N)) \sqsubseteq N$. Therefore (1) holds for all i and we obtain $T_P \uparrow \omega \sqsubseteq N$.

(2) Let M denote $T_P \uparrow \omega$. We now prove by induction on i that

$$T_P \uparrow i \sqsubseteq C(T_P(M)) \qquad (2)$$

The basis is clearly true. Assume the claim holds for $i \geq 0$. By Proposition 7, $T_P \uparrow i \sqsubseteq C(T_P(M)) \sqsubseteq M$. By Proposition 10 (3), $T_P(T_P \uparrow i) \subseteq T_P(M)$. By Proposition 6 (2), $T_P \uparrow i+1 = C(T_P(T_P \uparrow i)) \sqsubseteq C(T_P(M))$. Therefore (2) holds for all i and we have $T_P \uparrow \omega \sqsubseteq C(T_P(T_P \uparrow \omega))$. By Proposition 8, $T_P \uparrow \omega$ is a justified model. □

Example 9 Consider again the program of Example 1:

$$\begin{aligned}
& class : class[isa \Rrightarrow class] \\
& person : class \\
& student : class[isa \twoheadrightarrow \{person\}] \\
& mary : student \\
& O : D \Leftarrow O : C, C[isa \twoheadrightarrow \{D\}]
\end{aligned}$$

The powers of T_P are computed as follows:

$$\begin{aligned}
T_P \uparrow 0 &= \emptyset \\
T_P \uparrow 1 &= \{class : class\} \\
T_P \uparrow 2 &= \{class[isa \Rrightarrow class]\} \cup T_P \uparrow 1 \\
T_P \uparrow 3 &= \{person : class, student : class\} \\
& \quad \cup T_P \uparrow 2 \\
T_P \uparrow 4 &= \{student[isa \twoheadrightarrow \{person\}], \\
& \quad mary : student\} \\
& \quad \cup T_P \uparrow 3 \\
T_P \uparrow 5 &= \{mary : person\} \cup T_P \uparrow 4 \\
T_P \uparrow \omega &= T_P \uparrow 5 = I_0
\end{aligned}$$

which is a model. It is therefore the least and justified model by Theorem 1.

Unlike traditional logic program, $T_P \uparrow \omega$ is not always defined due to the consistency constraint. Even if $T_P \uparrow \omega$ is defined, it may not be a model of P due to the well-definedness and well-typedness constraints.

Example 10 Consider the following program.

$class : class$
$person : class[spouse \Rightarrow person]$
$mary : person$
$smith : person$
$mary[spouse \rightarrow smith]$
$mary[spouse \rightarrow mary]$

This program has no models since no interpretation would contain both $mary[spouse \rightarrow smith]$ and $mary[spouse \rightarrow mary]$ by definition. We have

$$\begin{aligned}
&T_P \uparrow 0 = \emptyset \\
&T_P \uparrow 1 = \{class : class\} \\
&T_P \uparrow 2 = \{class : class, person : class\} \\
&T_P \uparrow 3 = \{person[spouse \Rightarrow person]\} \\
&\qquad \cup\, T_P \uparrow 2 \\
&T_P \uparrow 4 = \{mary : person, smith : person\} \\
&\qquad \cup T_P \uparrow 3 \\
&T_P(T_P \uparrow 4) = \{mary[spouse \rightarrow smith], \\
&\qquad mary[spouse \rightarrow mary]\} \\
&\qquad \cup\, T_P \uparrow 4
\end{aligned}$$

Therefore, $T_P \uparrow 5$ and $T_P \uparrow \omega$ are not defined since C is not defined on $T_P(T_P \uparrow 4)$.

The next example shows that it is possible for a program to have the least and justified model but $T_P \uparrow \omega$ is not a model.

Example 11 Consider the following program.

$class : class$
$person : class$
$john : person[likes \twoheadrightarrow \{john\}]$
$person[likes \Rightarrow\!\!\!\!\Rightarrow \{person\}] \Leftarrow$
$\qquad X[likes \twoheadrightarrow \{Y\}]$

The least and justified model of this program is:

$$\begin{aligned}
M = \{&class : class, \\
&person : class, \\
&person[likes \Rightarrow\!\!\!\!\Rightarrow \{person\}], \\
&john : person, \\
&john[likes \twoheadrightarrow \{john\}]\}
\end{aligned}$$

The powers of T_P are computed as follows:

$$\begin{aligned}
&T_P \uparrow 0 = \emptyset \\
&T_P \uparrow 1 = \{class : class\} \\
&T_P \uparrow 2 = \{person : class\} \cup T_P \uparrow 1 \\
&T_P \uparrow 3 = \{john : person\} \cup T_P \uparrow 2 \\
&T_P \uparrow \omega = T_P \uparrow 3
\end{aligned}$$

$T_P \uparrow \omega$ is not a model of P since it cannot satisfy the fact $john[likes \twoheadrightarrow \{cat\}]$.

Definition 26 A program P is *well-written* if $T_P \uparrow \omega$ is a model.

The program in Example 9 is well-defined, but the programs in Examples 10 and 11 are not well-written according to the definition.

Proposition 11 Let P be a well-written program. Then $T_P \uparrow \omega$ is the unique least and justified model.

Proof. Direct from Theorem 1. □

Thus, if a program is well-written, then it has the least and justified model which can be obtained by a bottom-up least fixpoint computation.

Theorem 2 It is decidable whether or not a program is well-written.

Proof. We compute the powers of the operators T_P according to the definition. As we compute $T_P \uparrow n$, if a rule is satisfied, we mark it satisfied; otherwise, we mark it unsatisfied. Since the number of objects, class and attribute symbols of a given program is finite, the possible ground substitutions for the object, class and attribute variables have to be finite. Therefore there exists an i, such that either $T_P \uparrow i$ is undefined or $T_P \uparrow i$ is a fixpoint such that $T_P \uparrow \omega = T_P \uparrow i$. If $T_P \uparrow \omega$ is defined and no rule is marked unsatisfied, then $T_P \uparrow \omega$ is a model and consequently is the least and justified model of the program by Theorem 1. Otherwise, the program is not well-written. □

Definition 27 Let P be a well-written program. Then its *declarative semantics* is given by $T_P \uparrow \omega$.

Definition 28 Let P be a well-written program and ?– $A_1, ..., A_n$ a query. Then an answer to the query is a ground substitution θ such that $T_P \uparrow \omega \models A_1\theta$, ..., $T_P \uparrow \omega \models A_n\theta$.

Theorem 3 Let P be a program and Q a query. Then it is decidable whether or not there is an answer to Q.

Proof. Direct from Theorem 2. □

This theorem says that even through HLO is a higher-order language, but the limited use of higher-order features in a deductive object-oriented database framework does not pose any problems.

6 Conclusion

In this paper, we have presented a high-order deductive object-oriented database language which is capable of directly and naturally supporting object-oriented features such as object identity, complex object, classes, and multiple inheritance in a uniform framework. We gave the language a simple Herbrand-like semantics which naturally incorporates three kinds of powerful integrity constraints: consistency, well-definedness, and well-typedness constraints. A number of interesting semantics properties of the HLO programs were investigated. We showed that incorporating the consistency, well-definedness and well-typeness constraints negates many important properties held for traditional logic programs. We then discussed the necessary condition for the most important property that should be held by HLO programs and the decidability of checking the necessary condition. Finally, we illustrated that if the necessary condition is satisfied, we can compute a program bottom-up to obtain its intended semantics based on which the queries can be answered.

We are currently considering extensions to the language to include set-valued variables, negation, and arithmetic and comparison operations. Besides, we intend to investigate how to incorporate update constructs into the language while maintaining the least model semantics and how to efficiently implement the language.

Acknowledgments

The work was partially supported by the Natural Sciences and Engineering Research Council of Canada.

References

[1] S. Abiteboul and S. Grumbash. COL: A logic-based language for complex objects. *ACM TODS*, 16(1): 1–30, 1991.

[2] S. Abiteboul and P.C. Kanellakis. Object identity as a query language. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, 159–173, 1989.

[3] S. Abiteboul, G. Lausen, H. Uphoff, and E. Waller. Methods and rules. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, 32–41, 1993.

[4] Serge Abiteboul. Towards a deductive object-oriented database language. *Data and Knowledge Engineering*, 5(2): 263–287, 1990.

[5] C. Beeri, S. Naqvi, O. Shmueli, and S. Tsur. Set construction in a logic database language. *J. Logic Programming*, 10(3,4): 181–232, April/May 1991.

[6] E. Bertino and D. Montesi. Towards a logical object-oriented programming language for databases. In *Proc. Intl. Conf. on Extending Database Technology*, 168–183, Springer-Verlag, March 1992.

[7] F. Cacace, S. Ceri, S. Crepi-Reghizzi, L. Tanca, and R. Zicari. Integrating object-oriented data modelling with a rule-based programming paradigm. In *Proc. Intl. Conf. on Very Large Data Bases*, 251–261, 1990.

[8] S. Ceri, G. Gottlob, and T. Tanca. *Logic Programming and Databases.* Springer-Verlag, 1990.

[9] W. Chen and D.S. Warren. C-Logic for complex objects. In *Proc. ACM Symp. on Principles of Database Systems*, 369–378, 1989.

[10] G. Dobbie and R. Topor. A model for sets and multiple inheritance in deductive object-oriented systems. In S. Ceri, K. Tanaka, and S. Tsur, editors, *Deductive and Object-Oriented Databases*, 473–488, Phoenix, Arizona, USA, December 1993. Springer-Verlag Lecture Notes in Computer Science 760.

[11] D.H. Fishman, B. Beech, H.P. Cate, E.C. Chow, T. Connors, J.W. Davis, N. Derrett, C.G. Hoch, W. Kent, P. Lyngbaek, B. Mahbod, M.A. Neimat, T.A. Ryan, and M.C. Shan. Iris: An object-oriented database management system. *ACM Trans. on Office*

Information Systems, 5(1): 48–69, January 1987.

[12] W.D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13: 225–230, 1981.

[13] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. Technical Report 90/14, Dept of CS, SUNY at Stony Brook, 1990.

[14] M. Kifer and J. Wu. A logic for programming with complex objects. *J. Computer and System Sciences*, 47: 77–120, 1993.

[15] Won Kim. *Introduction to Object-Oriented Databases.* The MIT Press, 1990.

[16] R. Krishnamurthy and S. Naqvi. Towards a real horn clause language. In *Proc. Intl. Conf. on Very Large Data Bases*, 252–263, Los Angles, USA, 1988.

[17] C. Lecluse and P. Richard. The O_2 database programming language. In *Proc. Intl. Conf. on Very Large Data Bases*, 411–422, Amsterdam, The Netherlands, 1989.

[18] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 2 edition, 1987.

[19] Y. Lou and M. Ozsoyoglu. LLO: A deductive language with methods and method inheritance. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, 198–207, 1991.

[20] D. Maier. A logic for objects. Technical Report CS/E-86-012, Oregon Graduate Center, Beaverton, Oregon, 1986.

[21] Francis G. McCabe. *Logic and Objects.* Prentice Hall., 1992.

[22] J. Mylopoulos, P.A. Bernstein, and H.K.T. Wong. A Language Facility for Designing Database-Intensive Applications. *ACM Trans. Database Systems*, 5(2): 185–207, June 1980.

[23] Shamim Naqvi and Shalom Tsur. *A Logical Language for Data and Knowledge Bases.* Computer Science Press, 1989.

[24] R. Ramakrishnan, D. Srivastava, and S. Sudarshan. CORAL: Control, relations and logic. In *Proc. Intl. Conf. on Very Large Data Bases*, 238–250, 1992.

[25] Carlo Zaniolo. Object identity and inheritance in deductive database —an evolutionary approach. In W. Kim, J.M. Nicolas, and S. Nishio, editors, *Deductive and Object-Oriented Databases*, 7–21, Kyoto, Japan, December 1989. North-Holland.

On the Balance of the Informational Exchange, Its Flow, and Fractional Revealing Large Informational Quanta, in the 'Hot' Living Systems ($T < 0_-$)

Jíři Šlechta
Member of New York Academy of Sciences
18 Lidgett Hill
Leeds 8, LS8 1PE, U.K.

Keywords: 'hot' living systems ($T < 0_-$), informational exchange, informational quanta

Edited by: A.P. Železnikar

Received: February 1, 1995 **Revised:** July 14, 1995 **Accepted:** August 4, 1995

It is discussed the case of the informational 'saser' in the 'hot' living systems ($T < 0$), for $T < 0_-$ (the absolute negative zero), and explained why the large quanta of information, like that stored and developed by a genius, or a voluminous new discovery, or a marital love, spread less easily, except for $T \approx 0_-$, than the smaller quanta, like a random love.

A way how to improve this property is to design a sparse set of smaller quanta which span the large quantum (e.g. a kind of teaching the large issue, or policy making about it). It is discussed how to design such a set.

In the balance of the equivalence of the various forms of the spanning, a generalized equation of the conservation 'energy' figures the relative, and not the absolute, precisions.

There is provided a formalized framework of the intercommunication among n persons.

On the basis of it there is also given the first exact explanation, in existence, why the 'two house' sex is superior to any other form of it, why an offspring of higher animals needs two parents, the theory of measurement of the objective reality, the relation between the theory and experiment, the difference between working of the brain of the people with associative and verbatim memories, and modern and underdeveloped societies, etc.

The theory presented here is important for the design of an artificial intelligence [the logic of the real self-organizing systems—(fuzzy) physical logic], and methodology of teaching, the theoretical design of the optimal society, etc.

1 Introduction

In [1, 2] there were formulated theoretical foundations of two kinds of the self-organizing living systems, the brain [1], on one side, and the society [2], on the other side. They have in common they consist of sparsely distributed members of these systems which are, relative to their surroundings, informationally charged by an informational quantum ε_0.

These members are either the memory traces (further MTs), in the brain [1, 3–6] or, the relatively, against their surroundings, informed members of the given society (further SMs) [2, 7–9], which are both excited relatively to their surroundings.

Though the detail properties of these systems are different their theoretical descriptions have a lot in common.

Because of this it is to be formulated, here, a theoretical core common to both of these systems, at first.

The results of this study are then applied to the special cases, parallels, in each of these two systems, in particular.

2 A Unitary Description of the Brain and Society as 'Hot' Living Systems

Within such a general core the members of a self-organizing living system, of which these two kinds of living systems are specific examples, will be called the informed members of the given living system (further IMLSs).

In [6, 7] it was shown that to such a kind of a living system, further called a generalized living system (or GLS), there may be ascribed the absolute temperature T given by the formula ([6, 7])

$$\frac{1}{T} = \frac{1}{2\varepsilon_0} \ln\left(\frac{N_-}{N_+}\right) \qquad (1)$$

where $N_- + N_+ = N$.

Here, N is the total number of the members of the living system (further MLSs) capable to be excited, N_- is the number of the unexcited MLS, and N_+ the number of the excited MLS, or IMLSs, T is the absolute temperature of a saser in a society [2, 7], or of a generalized laser in the brain [1, 3, 6], and ε_0 is the energy of the excited level of MLSs, relative to the surroundings [1, 2, 4].

As it is known from the theory of the lasers [10–12], in the case of the so-called inverse occupation of these discrete excited levels, when $N_t > N_-$, that is, when the number of IMLSs is higher than the number of the unexcited MLSs, for the absolute temperature T it holds that $T < 0$ [1, 2]. This is a property having the physical meaning for the discrete systems only [10–12].

In [1, 2] this concept was applied and generalized for the thought processes in the brain, and the group of organizers and information processors in a society (e.g., the managerial group in a plant, the scientists and politicians in a society, etc.)

When the total number of MLS exceeds 15–25 the cooperative GLS starts to have macroscopic features, so the thermodynamics [1–10] can be applied to it. It holds

$$dS = \frac{dE}{T} \qquad (2)$$

where S is its entropy, E its internal energy and T its absolute temperature.

In [7, 13] it was shown that for a two component GLS consisting of: a) a continuous system I with $T_I > 0$ and, b) a finite discrete system II with $T_{II} < 0$, it holds

$$dS = dE_{I,II}\left(\frac{1}{T_I} + \frac{1}{\mathrm{abs}(T_{II})}\right) \qquad (3)$$

where $E_{I,II}$ denotes the energy of the mutual exchange between I and II and for $dE_{I,II} > 0$ the energy flows from II to I.

Therefore, because $dS > 0$ is always true, the energy in the system is bound to flow from II to I, providing that in I a receiving resonance needed [7] can be found . The equation expresses the fact that a system with $T < 0$ is 'hotter' than any system with $T > 0$ (an ordinary continuous system [13–15]).

3 The Case of $T <> 0$

In detail, in [6, 7], mainly the case close to the negative absolute zero 0_-, when all the MLS capable to do so are excited, was discussed.

One kind of the limiting case, for $T \approx 0_-$, was the case of a 'genius' [2, 7] and other singular situations, like a mother, or a father to their child, etc. [7], to both of which the same formulae can be applied.

This paper deals, in more detail, with cases of $T < 0_-$, where T is considerably different in regard to 0_-.

While in case $T \approx 0_-$, the temperature depends on how close the situation is to the hundred percent inversion, only in case $T <> 0_-$ it depends also upon the size of ε_0 [Eq. (1)].

In the latter case, for the same degree of inversion, that is for the same value of the ratio $\frac{N_-}{N_+} <> 0$ in Eq. (1), the larger ε_0 is, the larger abs(T) will be, that is, for $T < 0$, the lower temperature (on the temperature axis, with the temperature arranged with a growing degree of the inversion, or the internal energy) the system will have.

It means, for the same degree of the inversion, the more voluminous message (e.g. large original works or strong love, especially the marital and parental ones, etc.) has a smaller temperature gradient relative to its surroundings than a shorter message (e.g. short texts, like letters, or occasional short love affairs, etc.). This causes the former ones to spread less easily, more slowly, more cumbersomely.

By this a well known fact is described that the larger works happen to be more 'static'. It is for them more difficult to become known. For this reasons the scientific papers, or magazine articles, as kinds of the published form, are a necessity among letters and books.

In case of the brain, the long term MTs are more voluminous: they possess either a large ε_0 of the knowledge of 'a virgin' type (a kind of a new background knowledge [2]), or messages gained by a relatively small ε_0 projected against the large background knowledge, already present in the brain (a sudden release of this explains the power of jokes, or the poetry, and why it is difficult to be gained by foreigners).

When an event is sampled by a field of random small probes, for example, in the form of a short interview of a bigger personality, it may result in a Brownian type result [14] for the appreciation of his/her uniqueness, that is in a zero appreciation and the degree of understanding, and acceptance of his/her message.

Because of this the voluminous works have lesser 'irradiability' than the short ones. To improve their acceptability, or even to prove their uniqueness, requires more extra efforts, often unwilling to be inserted by other people and thus remaining entirely out the horizon of their capabilities.

The voluminous knowledge is more 'calm' and stable (e.g. already formed marriages, relative to 'testing pairing'). The static feature of the voluminous ε_0 may be enhanced by the unease to find for it a full resonance [7].

This can have a practical consequence for the real top people and even those who are not entirely unique and absolute, can appear often relatively 'static'. So it takes time for them to become recognized in comparison to people of fast short wits of specks of obvious uniqueness (often 'tested' locally only).

However, in a long term run, according to the thermodynamical results in [7], the top people of the voluminous knowledge are eventually bound to be accepted, to win, and then they are impervious to be 'scattered' from their course by 'petty' attacks (here the theorem works on the opposite parameters which control the statements like 'nothing is wrong with his/her [their] work', 'his/her [their] work has been really best [in the world, or a given region], in a direction', etc., which stabilizes the gained results).

The transition between these two states may be dramatic and up to a catastrophic strengths for some individuals involved.

However, to describe this mathematically has been beyond the thermodynamical framework of [7] and the related works [2, 7–9, 15, 16].

Even the works on time scales in economy [17], and thought processes [18], are still not kinetically detailed enough as they lack the nonlinear coupling between the processes at the individual level of IMLSs and the macroparameters (especially the phase transformation in them) of the given living system, within its surroundings.

4 On a Fractional Spread of a Voluminous Piece of Information

As pointed out in [7], a larger ε_0 can be passed by smaller portions ε_{0i}, such that

$$\sum_i \varepsilon_{0i} = \varepsilon_0 \quad \text{(superfuzzily)} \qquad (4)$$

Here, the superfuzzy 'equal' means it is not necessarily so that the sets of the lessened portions ε_{0i} are crisply defined in an unique way. Their choice can be from a fuzzy superset [19] of all such sets, each of them with a fuzzy character, which fulfills these spanning properties.

Each portion $\varepsilon_{0i} <$ (or even $\ll$) ε_0, and therefore, according to Eq. (1), the absolute temperature T_{0i} ascribed to it, is for the same degree of inversion for the given ε_{0i} as for ε_0 higher than T_0 of the original quantum of information ε_0, and closer to 0_-.

Moreover, to generate the inversion for ε_{0i} is, in principle, easier than for ε_0, because it is easier to find SMs capable to absorb them resonantly [6].

It makes the set of the SMs with the index i 'hotter', relative to the temperatures of the set of the other SMs to whom it is to be passed, than the original voluminous quantum ε_0. Therefore, the latter SMs are to be able to receive it more easily and more readily.

Because of this it is likely there will be more members of GLs ready to receive the partial information ε_{i0} than the original complete information ε_0. This makes possible that the number N_i of these can be much higher than N (often zero) of those able to receive ε_0.

This, however, makes the portion ε_{0i} less unique because relatively more many of GLS members, able to receive the fractional ε_{0i} remain unexcited, initially, which lowers the temperature T_{0i} relative to T_0—an opposite tendency to that due to the smaller size of partial ε_{0i}.

This is enhanced further due to the fact that to achieve a partial fraction $\frac{N_{i-}}{N_{i+}} \approx 0$ (an absolute inversion for ε_{0i}) to increase T_{0i} [Eq. (1)] to gain a larger gradient with the relevant receivers, may take a considerable effort, which decreases the gain achieved by suitable 'biting' the original message ε_0.

Nevertheless, as pointed above, it is in principle practically more readily possible and is rather the question of a supply of the energy to the teaching, propaganda, or activities, while it was virtually impossible for ε_0, to enhance the 'mass' receptivity of the total geniality within the same period of time, well nigh impossible.

However, on top of these two competing tendencies, the conservation sum (4), inclusive its particular fractionating, makes it again very unique, with the summary effect of shifting the whole sum towards 0_- (the situation 'genius') of the measure of its uniqueness equal to 1, thus achieving the goal: the total result of the fractionalization is both not less unique than the original message ε_0 and it has the improved capability to be perceived (received) by the society (per partes).

Even when each ε_{i0} is also 'genially' unique, the number of MLSs resonating with these shortened quanta is likely to be much higher than 1, which guarantees a success of the social spread of the knowledge of ε_0.

To produce such a fractionalized communication is one of the skills of politicians.

Such a situation happens also in teaching of any subject in schools in smaller units of individual lessons (a kind of a sparse control of knowledge [7, 15, 16]).

The role of the publisher is to provide the resonance [6] to the authors and provide a way of dissemination of it, and making for them the coefficient of efficiency $\eta > 1$ [8]. Otherwise, this is a daunting task for any individual. Historically, it has been evidence the lack of a plenty of publishers leads to a stagnation of the given intellectual area.

To provide for a publishing freedom has been, therefore, a nobel duty of any government.

5 On the Relative Precision

Let us present a brief generalization of the theory of the evaluation of the precision of the measurements like it is done in physics [17].

When dA is the precision of the measurement, or contribution *of* or *to* a quantity A, the ratio

$$\mathrm{rp}A = \frac{dA}{A} \tag{5}$$

is the relative precision or contribution.

For example, an absolutely small, but nonzero, contribution dB to an area B about which there has been no knowledge, that is $B = 0$, has its rp$B$ infinitely large (e.g., the case of a 'genius' [2]), while rp$C$ of even a large volume dC added to a well developed field C may be rather small (here, it depends upon the structure of the contribution of dC related to C; the more abstract and general dC is the larger the related rp$C$ can be).

Let us apply this to the precision of the satisfying Eq. (4).

Let res(ε_0) mean the imprecision of the balance of Eq. (4); then the ratio

$$\mathrm{rres}(\varepsilon_0) = \mathrm{abs}\left(\frac{\mathrm{res}(\varepsilon_0)}{\varepsilon_0}\right) \tag{6}$$

may be called the 'relative residuum'.

The balance equation (4) for two original quanta is satisfied comparatively to the same degree of precision when the two relative residua related to them are of approximately the same value.

To use, for such a comparison, the absolute precision of the balance of the original quanta of very different sizes is practically incorrect for the larger one, because the situation, related to it, is likewise more 'coarse'.

When comparing the fulfillments of the given two different balance equations (4) it is more correct to compare the related relative residua because to compare the absolute precisions of such fulfillments is unfair to the equation with the larger ε_{0i}. Such an equation may be fulfilled, in absolute terms, rather coarsely, that is less finely. This may be scoffed upon, yet, in relative terms it is equally sufficient.

For example, it is sufficient to balance the plans and estimates of budgets in millions pounds, to thousands pounds, while, budgets in hundreds pounds need to be made to the last penny. Especially, to forecast the large budgets more precisely, in the absolute terms, is, by and large, practically impossible and has no economical sense. Rather the opposite is true because the impossibility of the forward predictability may waste vast sums of money, and it is mostly futile, anyway (this is true, however, for the forward planning only, while in the case of the real accountancy 'backward', a recursive usage of Eq. (4) makes possible, and desirable, to balance it to the last penny).

6 Informational Balance During a Fractional Compound Pass of an Informational Quantum

Let I_{Si} be the informational background of the SM_i and IP_{Si} the information passed by the receipt of ε_{0i}; then ([2])

$$IP_{Si} = I_{Si} - \varepsilon_{0i} \quad \text{(superfuzzily)} \qquad (7)$$

To achieve the fractional pass of the original informational quantum ε_0, in Eq. (4), by spanning it by IP_{Si}, it should be that $\sum_i IP_{Si}$ contains ε_0 superfuzzily, that is,

$$\sum_i IP_{Si} \supset \varepsilon_0 \quad \text{(superfuzzily)} \qquad (8)$$

If $I_{Si} = 0$ then ε_{0i} must be considered for a 'virgin' piece of information, and not to be judged [2].

This happens, for example, in the case of a formal schooling, when the students do not have, at the beginning of their course, any background in its direction.

Also two experts should consider this to happen when communicating with each other. This was studied in details within the concept of the 'spinning top' [21].

The process of fractional spanning can be recursive. This is practiced during the formal education from the primary to the higher education.

In scientific papers the references are given which span the necessary background against which, in the mind of the author, the given paper was written, and against which it should be read, too.

This property is recursive until the reader reaches the level where he already reads everything with understanding. From such a level he can return 'up' to the initial paper of his interest.

Often, within the life situations, except the scientific papers, the relation expressed by Eq. (7) is not taken into an account because of the lack of the awareness of its existence, and of the underlying informational processes and effects, which make the compound spread of a voluminous piece of information unreliable.

For example, it is commonly forgotten or abandoned the check whether the integral fractional message, represented by the left-hand side of Eq. (4), represents sufficiently adequate the original message at the right-hand side of the equation, and what kind of misimpression, and how large, is created by this.

A special case of such a situation is the lack of a check how precise are the collected personal data about a person in the cadre material files. This makes such a kind of the formation of the personal information unreliable, even socially dangerous (e.g. the known erroneous cadre politics in the former communist block of the East Europe).

In many cases the smaller quanta ε_{0i} related to the message ε_0, in Eq. (4), are meant to advertise ε_0 to be studied more comprehensively, rather than to replace it.

This is an important technique of sketching new voluminous, medium and long term projects, both by leading politicians (from Form 3 and 2, to Form 2 and 1 [2]) and scientists (from Form 3 to 3, and 2).

7 On Intercommunication between n Persons

Let us sketch briefly a linear theory of the intercommunication between n people.

Let A_i be the total knowledge of the i-person. Then, in a linear approximation, it may be written in the form

$$A_i = \sum_j A_{ij} \quad \text{(superfuzzily)} \tag{9}$$

where A_{ij} is the partial well defined j-area or j-field of the knowledge of the i-person, noninteracting with other areas of the same person (the property of the linear approximation). It means that these are well defined, distinct areas or fields of a person's knowledge.

However, the superfuzzy sense of the relation (9) caters for a situation when the components A_{ij} and A_i are determined by a practical observable kind of procedures, like psychological or sociological experiments, consisting of a series of separate sessions or events (the fuzzy part), within a series of various situations or circumstances (the 'super'-fuzzy part over it), selected at random patterns.

The informational contents of the area A_i, the quantity also needed in Eq.s (4) to (8), may be defined by the Mod(A_i).

Then the set product

$$A_i \cap A_k = \sum_{j\ell} A_{ij} \cap A_{k\ell} \quad \text{(superfuzzily)} \tag{10}$$

defines the common part of the total knowledge shared by the i- and j-persons (mutually determined, for example, by two lovers within repetitions [the fuzzy part] of various [the 'super'-fuzzy part] shared events).

The members of the set of the shared partial pieces of knowledge

$$A_{ij,k\ell} = A_{ij} \cap A_{k\ell} \quad \text{(superfuzzily)} \tag{11}$$

are, if they exist, rather smaller, narrower, than the members of the set of the two compound sets A_{ij} and $A_{k\ell}$.

The spectrum of the perception of the new piece of information A by the i-person is given by the set product

$$A \cap A_i = \sum_j A \cap A_{ij} \quad \text{(superfuzzily)} \tag{12}$$

where the set

$$B_{ij} = A \cap A_{ij} \quad \text{(superfuzzily)} \tag{13}$$

is an analogue of the spectral 'lines', or 'colors' of A perceived by the i-person. Nothing else of A is recognized by the person.

The shared perception of such a new area A by the i- and j-person is given by the set product

$$A \cap A_i \cap A_j = \sum_{j\ell} A \cap A_{ij} \cap A_{k\ell} \quad \text{(superfuzzily)} \tag{14}$$

The shared set

$$B_{ijk\ell} = A \cap A_{ij} \cap A_{k\ell} \quad \text{(superfuzzily)} \tag{15}$$

consists, again, of mostly much narrower members than the compound set $A \cap A_{ij}$. It is often even all empty (those two people have nothing in common to share about the area A).

The structure, relative to the total knowledge, insets of the persons involved, given by the subtractions

$$C_{ij} = A_i - A_{ij} \quad \text{(superfuzzily)} \tag{16}$$

determines the quality of the perception of the area A_{ij} by the i-person, for example, a sense of its beauty.

The set

$$D_i = \sum_j B_{ij} \quad \text{(superfuzzily)} \tag{17}$$

defines the integral sense of beauty the i-person feels about his total perception of the world.

The set product

$$D_{ij} = D_i \cap D_j \quad \text{(superfuzzily)} \tag{18}$$

provides then the total mutually shared beauty by the i-person and j-person, for example, their mutual love at the first sight [4], etc.

Sometimes, some of the elements $A_{ij,k\ell}$, in Eq. (11), form a crystalline grid, for example, a party sharing the same well-defined and rather short, or/and relatively (in contrast to the whole 'teaching') simple doctrine. The narrower they are, the more rarely this happens, however, the stiffer such a grid is. It may become then, due to the le Chatelieu principle, a source of the possible oppressive and morbid effects [2] (e.g., the oppression by the Nazi, or Communist parties, or simple nationalism).

8 Further Consequences

1) On the relation of the experimental results R_{0i} to the 'objective' (total observable) reality R

Eq. (4) written here in the form

$$\sum_i R_{0i} = R \quad \text{(superfuzzily)} \qquad (19)$$

where the summation is to be taken for a Stiltjes integration, expresses sufficiently how the objective reality can be (is) extrapolated by the set of the measurements (observations) R_{0i}.

Eq. (7) can be written in the form

$$IP_{R_{0i}} = C_{R_{0i}} - R_{0i} \quad \text{(superfuzzily)} \qquad (20)$$

where $C_{R_{0i}}$ is the context within which the i-experiment is carried. $IP_{R_{0i}}$ expresses the meaning, the informational yield of (the message received by) and R_{0i} represents an experiment within such a context.

Within the first laboratory measurement of the given kind ever, $C_{R_{0i}} = 0$, and the experiment R_{0i} is the message per itself. Only any further repetition of it yields a new extra differential message.

Then the inequality

$$IP_{R_{0i}} \supset R \quad \text{(superfuzzily)} \qquad (21)$$

conveys that the total message in all the measurements contains also R.

2) On the relation between the theory and experiment

The theory R_{mod} of R represents a theoretical description, an image of R, inclusive the way of how it is accessible by the set of experiments $R_{0i\text{mod}}$. It gives the prescription how to evaluate $R_{0i\text{mod}}$.

When $R_{0i\text{mod}} = R_{0i}$, the theory of R is said to be a proper one.

If no other equivalent or better theory is designed, then R_{mod} is considered to be the (only) proper one.

Within the progress of the development, the exploration of R by the extrapolation of R, defined by Eq. (19), is enlarged.

Historically, such an extrapolation process had started by naive observatory results, a kind of naive experimental data which, then only after, were theorized upon.

Here it can be said that the art of philosophy includes a nonmathematical way, the first stage of theorizing.

Professionally, the theorizing preceded professional experimenting because this includes a theoretical design of the experiment itself, based upon the preceding theory, at the beginning a naive one.

From this stage what is 'at first and the next' has reverted the order.

The professional theorizing opens a new way of evaluation the experimental data inclusive the theoretical design of new experiments to be theorized upon.

In this way, in the present, the theory precedes the experiment.

In details, a new field of study starts with the first phenomenological data, an analogue of the naive way of the past, and then it progresses as described above.

3) More on the spanning a large information quantum by a set of smaller, partial ones—on properties of their grids

The partial messages ε_{0i}, each of them, form more easily, than the original quantum ε_0, a crystalline grid, which can both: a) form a grid of reference for an acceptance of ε_0, or b) cause a refusal of ε_0.

For example, if the set of ε_{0i}, about the habits, or knowledge, of a person is chosen, even deliberately, viciously, simplified, stressing the negative points, then the person's special features are taken for a nuisance and not accepted.

Although such a person may be refused on the basis of his/her more comprehensive integral personal properties ε_0, he/she may be found a good person and as such accepted [e.g., any personal interview of a senior published person(ality) may easily go berserk].

The negative 'grid' may happen to be formed by either a spontaneous overlook or vicious change, by the phenomena described by Eq. (7).

This must also be taken into account in case of using phone (cellular), because the issues propagated by phone networks may be far from the truth about the bulk of the reality.

Moreover, the crystal-like grid of the erroneous small messages, created like this, may repel any real managing of the reality. It is because it can be formed actually of very sharp, impressively sharp points, because the set product of the lack of the understanding expressed deliberately in the opposite 'light' (expressed like 'this, we all agree, needs to be understood') produces them as such. However, when an expert is found for such a problem, the problem needs not to be really soluble or well defined on the set of all known things.

This may lead to the futile engagements.

For this it is best to declare the problems on the most general level possible and leave it to be considered by the whole set of the most top experts, each of them trusting individually, that is not to be bossed, or supervised by anybody [2].

4) On the superiority of 'two-house sex'

Eq. (4) of fractional revealing explains why the bisexual 'two-house' (when male and female sex is located in two bodily different partners) reproduction is superior to any other form of the sex 'software'. This is because the parents form a nucleus of a 'crystalline' kind of a 'virtual grid' in the direction of rearing [creation—(re)production], and guarding, their offsprings. Its memory traces are deposited during the time flow in the space by the (wild)life around the parents in focus. These memorize their guarding and feeding actions.

The crystallinity character of this nucleus expels most of the (substantial) danger to their offsprings, though it may attract the predators. This case of 'guarding parents couple' includes also the social rules of the ethics of the formation of marriages. The grid of these rules provides the coherency conditions for the self-regulating genetic selection of the required properties of the next generation (self-developing the population to the future needs of the society) according to Eq. (10).

In the overcrowded conditions the grid of the elements $A_{ij,k\ell}$ [Eq. (11)] prevents the breeding because its elements collide with the availability of the 'society' slots of prospective people's aspiration [22].

One-house sex or child care does not have this property. That is why the children living with one parent only, are so much less protected relative to the children from a complete, two-parent family.

However, in the overcrowded or difficult conditions the one-house sex may provide freer rules for breeding, because the conditions of a free copulation (a kind of a modern matriarchate) are less tough than any formal rules of the parentship within a (e.g. Christian) marriage (a possible source of the population explosion [22], with a week genetic coherency, making the population time static, nonadjustable to the technological and other needs and changed in underdeveloped nations).

They are however, less coherent, that those of two-house sex and this decreases the speed of the genetic progress of the development of the given community (spice).

This kind of improving of a breed may be improved by the type of relations spanned by Eq.s (10–15), which is a hallmark of a pedigree breeding (e.g. of dogs, while in the case of cats the man-controlled 'logic' is much weaker). This includes a record of it all (e.g. family trees)—a necessity of any kind of coherency.

On the other hand, to have more than two guardians does not improve the offspring security, because in the case when the offspring is, at any given moment, with one of its parents (guardian) only, which is often so in the real situation out of their home (nest, den, family house), while the second partner is around to be available in case of further needs, inclusive the provision of the food for the family, the danger to the offsprings decreases exponentially only, with any additional guardian (in a way, a secured family dwelling is a kind of a virtual [third] guardian [22]).

Due to this the increase of the safety of the offsprings by the third guardian is already relatively small, which makes the provision of it not economical enough justified.

The filter of a recognition of a danger is also likely to work on the principle of Eq. (10), when instead of n persons there is considered a set product of n events, the returns of the parents to their nest, by which there is selected (amplified) sharply the smell, and other traces, of a possible intruder or a danger.

5) On the safety aspects of a friendship

By a similar pattern, any kind of a strong friendship works, especially in dangerous and risky situations, or love, and in the sense of affinity.

6) Further on problems of fractional revealing and feedbacks

According to the 'spinning top' [21]), a detailed revelation of the working of Eq. (7) (and its simpler form in [2]), development of the fractionalizing (which includes any partial) revealing by phone, is correct either for the 'virgin initiation' of the matter, by short calls, or on top of a large volume of a body of information only.

In both cases the informational yield is high.

Between these two cases, however, there must come written forms, or real personal inspections.

If this is not upheld this may lead to a creation of paradoxes, spreads of social phobias, prejudices, etc.

These are mostly caused by nonuse of the correct feedbacks to test the minimal necessary relative precision (and also what this means) of the satisfaction of Eq. (4).

This includes testing of the convergence of the choice of a set of the experiments R_{0i} to satisfy the 'equal mark' in Eq. (19).

Often the one-step decision technique is used when the saser of IMLSs in focus is considered as a small system with a fully defined informational chart [23], rather than a part of a big system (i.e., the core saser being embedded into the rest of the society [2, 7]) which requires using a negative feedback technique [23].

Best to realize this is by the three-way technique: 1) to announce the decision, 2) to check how precisely it is corresponding to the intention, or being fulfilled, and to define the way of correcting it, if this is not minimally relatively precise, and 3) to check the workings of the corrective measure. If the minimal precision is not fulfilled in the sense to progress in a recursive manner, repeating the point 1) again and defining new corrective measures, till the minimal relative precision is reached, or being concluded the further attempting, it is not realistic.

In the way of discriminating it should be used the minimal possible number of negative criteria, of an elimination, as an example of these are the Ten Commandments of the Holy Bible, rather than a long list of those cases which are considered [8, 21].

In case of the published matter it should be passed everything possible [for example in the case of a 'virgin' text on a new subject the relative precision—see Eq. (6)—is high nearly in any case having a sufficient form], and the referee should be forced to define as precisely as possible the reasons (like further references to be considered), because the set of the mutually shared background experience $A_{ij,k\ell}$ of the referee and the author is bound to be narrow (this may cause a controversial and not too much, relatively precise referee's judgement).

Only really provable erroneous papers should be rejected. Otherwise, even a controversial paper is better to be published (because the relative precision of the saser of its referees need not to be representative within the rest of the world science) and let it to be evaluated by the whole mankind—its scientific part.

Any other way of dealing with it causes the creation of an extra entropy, and costs, due to it ([8]).

7) On a comparison of the 'lexicographic' and associative kinds of memorizing

The brain of the people who memorize easily a controlled amount of information beyond the common short-term memory working pattern [1, 3] form easily the long-term MTs of a large ε_0.

These, however, do not recoil easily in cooperation with any of their surroundings, and this explains why such people, for example linguistically talented people, are likely to lack the associative skills, and are not apt to mathematics.

On the contrary, the associative brains seem to form easily associations of a short lifetime, which create the long-term memories along a complex dynamic path.

This is, however, suitable, for abstract thinking (of a mathematical kind), which is carried upon the eigenvalues of the arrays of short-term data, which needs a fast diagonalization [24].

8) On the role of informational Forms for the properties of the given society

In a parallel analogy, in the direction of the application of the general core to the society, the nations which prefer the published works (Form 3—[2]—the mode of science, and arts) only, because of underdeveloped phone, and other electronic ways of communication and information processing, tend to be heavy, and less short-term creative, though with a permanent memory. This makes them only long-term creative, though cumbersomely, while many detailed correlations are lost (a property of underdeveloped nations, especially the large ones, of old cultures).

While the nations with the preference of the

Form 1 are very creative in short-term scale (folklore), while having no permanent records left (they are long-term Brownian)—the property of the nomadic nations ([25]).

The best is cultivating all three Forms, with minimal number of taboos and compulsory "do's" and "not-to-do's", functionally unjustified (the condition for optimal, non-noisy cellular automaton [7]).

The Form 2 provides the intermediate coupling in the society [9, 13] (modes of management and government).

This is performed by the really talented nations (often of a smaller size). They both create long-term memories, and are optimally socially associative, at any required speed, with necessary acceleration.

Within this point it is good to realize the role of the private car in the society. It made possible the society 'shake' off the relatively large, and cumbersome, steps of the coherency of the public transport, suitable for medium and large distances, only, which causes a lot of mistiming (entropy) within the everyday life, and replace it with the possibility of any size of the transportation step (within the scale of the personal private everyday life). This has made the modern society the 'talented' one.

9) On the role of the fractional revealing for the understanding among nations

The hatred between nations or religions, is created by the projections of any local events and issues, upon very different backgrounds of different integral cultures, in Eq. (7), within the people of different nations or creeds. They project the messages on their particular encounters.

It can be applied for this situation the case of interaction between n persons, with $n = 2$, etc.

Namely, this implies without the awareness of the workings of Eq. (7) it can happen to be build a grid of mutual hatred in the form of a 'crystal' of simple mutual prejudices, see Eq. (11), which then expel traces of any in their mutual tolerance. This may result in large scale national hysteria, as known in the history.

Such a 'crystal' of prejudices is the sharper the more mutually distant the cultures of those two given nations involved, are.

The role of the cultural and political workers is then to make this crystal less sharp, by making the members of the set, defined by Eq. (11), broader, even each its member different, thus the grid amorphous.

10) On the role of publishing

The published matter provides the energy to the people of the sasers near to $T \approx 0_-$. This makes possible to understand the kinetic reason for the efficiency coefficient $\eta > 1$ [15]. Any way of publishing is a form of a fractional revealing of the integrity of the author of the published matter.

A given set of published matter has to compete with the pool of the similar matter, and other forms of the dissemination of information in a free style. Any way of making it to be known compulsory to the whole public, and declaring it to be the winner by the definition of the state like, for example, books and public speeches of political leaders of the communist regimes in East Europe, is bound to create dictators with known consequences (due to the closeness of such a system, ending with maximal entropy as possible within the set of constrains of the system).

On the political scene, the competition of various political styles in a Free World style must be done only through multiparty system [12].

11) On the fractional revealing and leadership

The leaders create their disciples each of whom grasps and propagates, as his career, a smaller fractional message of his leader teaching.

The set of the disciples, by their professional skill, as enrolled is bound to be seen more 'genial', with T closer to 0_-, than their leader, who is more static, even when individually totally more absolutely more unique. He runs his disciples, as 'kites' [16], by which he provides the necessary support (informational energy), as if 'from below' (making it to self-organize, improve and develop his teaching further), to the saser consisting of those of his disciples who have developed his teaching in the particular direction more than he himself, that is those who are talented and independently creative ones on the basis of his teaching (an ad-hoc-gracy arrangement [2, 7, 26]).

A leader of the resistance has a high probability to be undiscovered.

12) On the fractional revealing, its physical implementation, and the artificial intelligence

The theory presented here is significant for the understanding of the working of the brain and society, and it is a contribution towards the design of the artificial intelligence. It provides for the first time, the foundation of the 'physical logic' for it, a kind of a special representation and realization of the principle of the duality of energy and information [27].

The social design of the system of the institution of a society is also a kind of an artificial intelligence (see the similarity between the structure of the theory of the brain [3–6, 10], on one side, and the scientific sociology [2, 7, 15] and economy [9, 17], on the other side).

References

[1] SLECHTA, J., *On Quantum Statistical Theory of the Brain—Brain as Generalized Laser.* In the Proceedings of the 7th International Congress of Cybernetics and Systems: The Way Ahead, ed. J.Rose, (1987) 919–924.

[2] SLECHTA, J., *Ethical Aspects of Expert Systems.* In the Proceedings, of the 12th International Congress on Cybernetics, Namur, Belgium, (1989) 58–68.

[3] SLECHTA, J., *On Thermodynamics of Thought Processes within a Living Body which exchanges both Energy and Matter with Its Environment.* In the Proceedings, of the 12th International Congress on Cybernetics, Namur, Belgium, (1989) 886–898.

[4] SLECHTA, J., *On Molecular Foundations of Thermodynamics of the Thought Processes in the Brain.* In the Proceedings, of the 12th International Congress on Cybernetics, Namur, Belgium, (1989) 870–877.

[5] SLECHTA, J., *On Thermodynamics of Thought Processes within a Living Body in a Changing Environment.* In the Proceedings, of the 12th International Congress on Cybernetics, Namur, Belgium, (1989) 878–885.

[6] SLECHTA, J., *Brain as a 'Hot' Cellular Automaton.* In the Proceedings, of the 12th International Congress on Cybernetics, Namur, Belgium, (1989) 862–869.

[7] SLECHTA, J., *Society as a 'Hot' Cellular Automaton.* In the Proceedings of the 13th International Congress on Cybernetics, Namur, Belgium (1992) 405–409.

[8] SLECHTA, J., *On Cybernetic Theory of Democracy.* In the Proceedings of the 13th International Congress on Cybernetics, Namur, Belgium (1992) 904–908.

[9] SLECHTA, J., *On Thermodynamics of 'Hot' ($T < 0$) Self-Organizing Processes in an Environment in a Changing Environment.* In the Proceedings of the 13th International Congress on Cybernetics, Namur, Belgium (1992) 935–939.

[10] KUBO, R., *Statistical Mechanics.* North-Holland Publ. Co., Amsterdam, 1965.

[11] KITTLE, C., *Elementary Statistical Physics.* John Wiley & Sons, London, 1960.

[12] FAIN, I.M. and J.I. CANIN, *Quantum Radiophysics.* Soviet Radio, Moscow, 1965.

[13] SLECHTA, J., *On Thermodynamics of the Brain.* In the Proceedings of the 7th International Congress of Cybernetics and Systems: The Way Ahead, ed. J.Rose, (1987) 914–919.

[14] BALESCU R., *Equilibrium and Non-equilibrium Statistical Mechanics.* John Wiley & Sons, New York, 1975.

[15] SLECHTA, J., *On Cybernetic Theory of the Democracy.* Presented at the 13th Int. Congress on Cybernetics, Namur, Belgium (1992), to be published.

[16] SLECHTA, J., *On Cybernetic Theory of the Global Knowledge System.* Presented at the 13th Int. Congress on Cybernetics, Namur, Belgium (1992), to be published.

[17] SLECHTA, J., *On the Theory of Time Scales in Economic Activities.* Presented at the 13th Int. Congress on Cybernetics, Namur, Belgium (1992), to be published.

[18] SLECHTA, J., *On the Molecular Theory of the Biological Time (Circadian Rhythms, Psychological Time).* Presented at the 13th Int. Congress on Cybernetics, Namur, Belgium (1992), to be published.

[19] GUPTA, M.M., REGADE, R.K., YAGER R.R., (Eds.), *Advances in Fuzzy Set Theory and Applications.* North-Holland Publishing Company, Amsterdam, 1979.

[20] MADELUNG, E., *Die mathematische Hilfsmittel der Physik.* Springer-Verlag, Berlin, 1957.

[21] SLECHTA, J., *On Informational Approach to the Problem of 'Seeded' Experts and Its Ethical Aspects.* Presented at the 13th Int. Congress on Cybernetics, Namur, Belgium (1992), to be published.

[22] SLECHTA, J., *On the Importance of the Acceleration Term for the Population Dynamics.* Presented at the 14th Int. Congress on Cybernetics, Namur, Belgium (1995), to be published.

[23] ASHBY, R., *Introduction to Cybernetics.* Chapman, London, 1956.

[24] SLECHTA, J., *On Spectral Properties of Neural Networks: Their Relation to Psychological Properties of an Animal.* Presented at the 1990 Principle James Williams Congress, Amsterdam (1990), to be published.

[25] BRONOWSKI J., *The Ascent of Man.* BBC, 1973.

[26] TOFLER, A., *Future Shock.* Pan, 1971.

[27] SLECHTA, J., *On Quantum-statistical Theory of Pair Interaction between Memory Traces in the Brain.* Informatica **17** (1993) 109–115.

ELEMENTS OF METAMATHEMATICAL AND INFORMATIONAL CALCULUS

Anton P. Železnikar
An Active Member of the New York Academy of Sciences
Volaričeva ulica 8, SI 61111 Ljubljana, Slovenia

Keywords: axioms of parallelism, serialism, circularity and spontaneity; catalogue of informational rules; decomposition axioms, decomposition theorem, deduction theorems, implicative axioms, inference rules; informational calculus, predicate calculus, propositional calculus; substitution rule; various axioms and theorems (implicative, conjunctive, disjunctive, equivalent, negatory, etc.)

Edited by: L. Birnbaum
Received: November 4, 1994 **Revised:** January 11, 1995 **Accepted:** February 3, 1995

This article deals with problems pertaining to the elements of axiomatics in traditional (mathematical, symbolic, philosophical) logic and to the problems of newly emerging axiomatics in informational logic. Informational axioms can be derived from propositional and predicate axioms and then, particularized and universalized within the informational domain. Traditional axiomatic formulas of the propositional and predicate calculus can become a rebounding cause for the construction of essentially different axioms in general informational theory. It is shown how propositional and predicate axioms and rules can be informationally extended for the needs of the general informational theory.

1 Introduction

Which are the basic and inferential informational axioms[1] which govern the derivation of formulas (theorems, consequences), that is, their proving procedures (proofs) in informational theories? This question is significant for the consciousness of that what scientific theories do outside of theories themselves, in their—from theories themselves separated—metatheories. On the other hand, informational theories are always unions of object theories and metatheories, where the last have the role of being the producers of theories in the sense of informational arising, that is, spontaneity and circularity.

2 Fundamental Figures of Syllogistic Inference

Syllogistic inference can be interpreted in different ways, for instance, in the scholastic (philosophical), informational and traditional-mathematical manner.

[1]This paper is a private author's work and no part of it may be used, reproduced or translated in any manner whatsoever without written permission except in the case of brief quotations embodied in critical articles.

2.1 An Informational Interpretation of Syllogism

Syllogism (συλλογή, in Greek, means gathering, collecting, assembly, concourse and συλλογίζομαι means to reckon, consider, think, reflect; to infer, conclude) (in German, das Zusammenrechnen, der Schluß, die Folgerung) is a valid (e.g., true) inference in a syllogistic form.

Syllogistics (in Greek, συλλογιστικῆ τέχνη the art of inferring, concluding) was founded by Aristotle and developed in scholasticism as teachings of (correct) inference in syllogistic form. This technique became the keystone of the traditional logic ([4], pp. 407–409).

The inferring in a syllogistic form proceeds from two premises to one conclusion. Thus, the valid inference, the so-called syllogism, with true premises, delivers a true conclusion. In parallel to the traditional logic, as premises of syllogisms, only the following forms of 'equivalent' informational formulas are allowed:

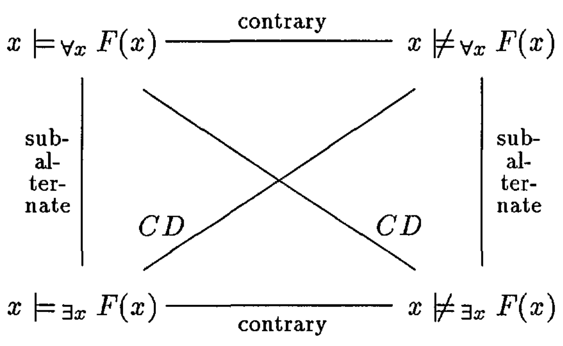
1. "All x are $F(x)$." Formula
$$x \models_{\forall x} F(x)$$
is a universal affirmative judgement and reads, informationally, x informs for all x the property (entity) $F(x)$.

2. "No x is $F(x)$." Formula
$$x \not\models_{\forall x} F(x)$$
is a universal negatory judgement and reads, informationally, x does not inform for all x the property (entity) $F(x)$.

3. "Some x are $F(x)$." Formula
$$x \models_{\exists x} F(x)$$
is a partial affirmative judgement and reads, informationally, x informs for some x the property $F(x)$.

4. "Some x are not $F(x)$." Formula
$$x \not\models_{\exists x} F(x)$$
is a partial negatory judgement and reads, informationally, x does not inform for some x the property (entity) $F(x)$.

The connections among these four formulas can be presented by the logical quadrate in Fig. 1.



$x \models_{\forall x} F(x)$
contrary
$x \not\models_{\forall x} F(x)$
sub-al-ter-nate
CD
CD
sub-al-ter-nate
$x \models_{\exists x} F(x)$
contrary
$x \not\models_{\exists x} F(x)$

Figure 1: *The logical quadrate, drawn according to the informational formulas of syllogistics.* CD *means contradictory.*

Four fundamental figures of the syllogistic inference can be distinguished, where $\models_{\mathfrak{a}}$, $\models_{\mathfrak{b}}$ and $\models_{\mathfrak{c}}$ are determined by

$$\models_{\mathfrak{a}}, \models_{\mathfrak{b}}, \models_{\mathfrak{c}} \in \{\models_{\forall x}, \not\models_{\forall x}, \models_{\exists x}, \not\models_{\exists x}\}$$

The position of the so-called middle entity μ is decisive and it must appear in both premises in the following manner:

$$\text{(FF1)}\ \frac{\begin{pmatrix} \mu \models_{\mathfrak{a}} \pi; \\ \sigma \models_{\mathfrak{b}} \mu \end{pmatrix}}{\sigma \models_{\mathfrak{c}} \pi}; \qquad \text{(FF2)}\ \frac{\begin{pmatrix} \pi \models_{\mathfrak{a}} \mu; \\ \sigma \models_{\mathfrak{b}} \mu \end{pmatrix}}{\sigma \models_{\mathfrak{c}} \pi};$$
$$\text{(FF3)}\ \frac{\begin{pmatrix} \mu \models_{\mathfrak{a}} \pi; \\ \mu \models_{\mathfrak{b}} \sigma \end{pmatrix}}{\sigma \models_{\mathfrak{c}} \pi}; \qquad \text{(FF4)}\ \frac{\begin{pmatrix} \pi \models_{\mathfrak{a}} \mu; \\ \mu \models_{\mathfrak{b}} \sigma \end{pmatrix}}{\sigma \models_{\mathfrak{c}} \pi}$$

If operators $\models_{\mathfrak{a}}$, $\models_{\mathfrak{b}}$ and $\models_{\mathfrak{c}}$ are replaced through concrete (particularized) operators $\models_{\forall x}$, $\not\models_{\forall x}$, $\models_{\exists x}$ and $\not\models_{\exists x}$, for every case an inference modus is obtained. In all, there are $4^3 = 64$ modi for each fundamental inference figure, that is, 256 figures. Only 24 of them are valid, that is, syllogisms (6 for each fundamental form, and for some additional suppositions have to be introduced). A modus is uniquely determined if the fundamental figure and the operators $\models_{\mathfrak{a}}$, $\models_{\mathfrak{b}}$ and $\models_{\mathfrak{c}}$ are known.

2.2 A Mathematical Interpretation of Syllogism

In modern logic, syllogism is treated in the framework of the first order predicate logic. The so-called universal and existential qauntifier, $\forall$ and $\exists$, can be represented with conjunctive ($\wedge$) and disjunctive ($\vee$) logical connectives within a certain domain (set) D of elements. E.g., the scholastic modi "B*a*rb*a*r*a*" and "F*e*l*a*pt*o*n", where the emphasized *a* stands "for all", *e* for "no", and *o* for "some ... are not", become

$$\frac{\begin{pmatrix} \bigwedge_{x \in D} M(x) \rightarrow P(x) \\ \bigwedge_{x \in D} S(x) \rightarrow M(x) \end{pmatrix}}{\bigwedge_{x \in D} S(x) \rightarrow P(x)} \quad \text{and}$$

$$\frac{\begin{pmatrix} \bigwedge_{x \in D} M(x) \rightarrow \overline{P(x)} \\ \bigwedge_{x \in D} M(x) \rightarrow S(x) \end{pmatrix}}{\bigvee_{x \in D} S(x) \wedge \overline{P(x)}}$$

respectively. The premise $\bigwedge_{x\in D} M(x)$ in Felapton must be completed. Predicate $P(x)$ corresponds to the middle entity μ in fundamental formulas (FF1–4) and, adequately, predicates $P(x)$ and $S(x)$ correspond to entities π and σ in (FF1–4).

The reader can find the logic quadrate for formulas, together with negatory cases, using universal and existential quantifiers and predicate $F(x)$ in Fig. 2.

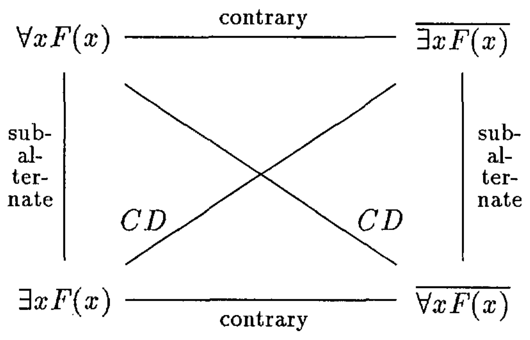


Figure 2: *The logical quadrate, drawn according to the predicate formulas of syllogistics using quantifiers* ∀ *and* ∃. CD *means contradictory.*

In Fig. 2 the contrary, contradictory and subalternate cases are shown in a clear, that is, negatory manner. There is, certainly,

$$\overline{\forall x F(x)} \equiv \exists x \overline{F(x)} \quad \text{and} \quad \overline{\exists x F(x)} \equiv \forall x \overline{F(x)}$$

and this equivalences (see Subsection 3.7) can be used as interpretations in Fig. 3.

3 A Short Overview of Some Mathematical Axioms

In der Aussage: „Der Hammer ist zu schwer“ ist das für die Sicht Entdeckte kein „Sinn“, sondern ein Seiendes in der Weise seiner Zuhandenheit. ...Aussage besagt soviel wie *Prädikation.* Von einem „Subjekt“ wird ein „Prädikat“ „ausgesagt“, jenes wird durch dieses *bestimmt.* Das Ausgesagte in dieser Bedeutung von Aussage ist nicht etwa das Prädikat, sondern „der Hammer selbst“.

Martin Heidegger [2] 154

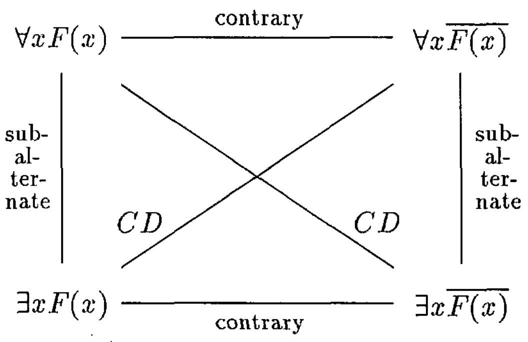


Figure 3: *The logical quadrate, drawn according to formulas in the previous picture with the resolved negations of the quantified formulas.* CD *means contradictory.*

3.1 Introduction

What is the nature of mathematical axioms and which kind of axioms are significant for our discussion? As we shall see, propositional axioms can serve as an outlook to the axioms used in the predicate calculus and also in the informational approach. The beginning of the general informational theory (GIT) has to be founded in logical axioms by which the informational phenomenalism (externalism, internalism, metaphysicalism) becomes a consequence of the very first assumption, that is, of the informational entity.

3.2 Axioms of the Propositional Calculus

Aussage bedeütet primär *Aufzeigung.* ...Aussage bedeutet *Mitteilung,* Heraussage. Als diese hat sie direkten Bezug zur Aussage in der ersten und zweiten Beduetung. Sie ist Mitsehenlassen des in der Weise des Bestimmens Aufgezeigten.

Martin Heidegger [2] 154–155

Axioms of propositional calculus are fundamental for (the entire) mathematics (metamathematics) so that they can be reasonably extended, for example, to different predicate calculuses, arithmetic and mathematical logic in general. There are several "systems of axioms" which differ, through time, from one author to another. In our approach, the axiomatic systems of Hilbert [3] and Novikov [5] were chosen.

In propositional calculus, axioms can be grouped in a traditional way (e.g., following Hilbert and Bernays [3] and Novikov [5]). According to [3], p.66, the initial axiomatic formulas can be grouped and, roughly, written in the form of axiomatic rules:

I. Axioms of implication
   1) $A \rightarrow (B \rightarrow A)$,
   2) $(A \rightarrow (A \rightarrow B)) \rightarrow (A \rightarrow B)$,
   3) $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$

II. Axioms of conjunction
   1) $A \wedge B \rightarrow A$,
   2) $A \wedge B \rightarrow B$,
   3) $(A \rightarrow B) \rightarrow$
      $((A \rightarrow C) \rightarrow (A \rightarrow B \wedge C))$

III. Axioms of disjunction
   1) $A \rightarrow A \vee B$,
   2) $B \rightarrow A \vee B$,
   3) $(A \rightarrow C) \rightarrow$
      $((B \rightarrow C) \rightarrow (A \vee B \rightarrow C))$

IV. Axioms of equivalence
   1) $(A \equiv B) \rightarrow (A \rightarrow B)$,
   2) $(A \equiv B) \rightarrow (B \rightarrow A)$,
   3) $(A \rightarrow B) \rightarrow ((B \rightarrow A) \rightarrow (A \equiv B))$

V. Axioms of negation
   1) $(A \rightarrow B) \rightarrow (\overline{B} \rightarrow \overline{A})$,
   2) $A \rightarrow \overline{\overline{A}}$,
   3) $\overline{\overline{A}} \rightarrow A$

The presented system of propositional axioms is not the only possible. For instance, Novikov [5], p. 75, chooses the group I of implication axioms in the form

1) $A \rightarrow (B \rightarrow A)$,
2) $(A \rightarrow (A \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

Various axioms can be useful for the interpretation of basic informational cases of phenomenalism, as we shall see in one of following subsections.

3.3 Inference Rules of the Propositional Calculus

Inference rules of propositional calculus are constructed on the basis of propositional axioms in the preceding subsection. It is hard to say which "rules" are the primary, the axiomatic or the inferential. However, it is clear that inferential rules must strictly consider the primitive axioms of propositional calculus which are, for example, implicative, conjuctive, disjunctive, equivalent and negatory.

Which is the general philosophy of making (constructing) a rule, precisely, the inference rule? Irrespective of the theory, for which rules are constructed, these are always implicative, although they express something more than a pure implication, because they concern the so-called derivation procedure. The role of a rule in the derivation procedure is the following: taking a rule in which several premises are logically connected in one or another way, some of them can be detached in the form of the so-called conclusion and, thus, between the respective premises of the rule and its conclusion a special operator, marked by $\vdash$ is used in propositional and predicate calculus while in the informational calculus we use operator $\twoheadrightarrow$ for marking derivation and an operator $\models$ for marking the circular form of informing.

Let us settle the general form of an inferential rule for deriving propositional formulas from axioms or from already derived formulas.

The first rule is substitution. Let us mark propositional formulas which depend on various propositional variables by the capital Fraktur letters, for example, as $\mathfrak{A}$ or, in more detail, $\mathfrak{A}(A_1, A_2, \cdots, A_n)$. Let $\mathbb{S}$ mark the operator of substitution (in informational terms, the function of substitution). Let us introduce

$$\mathbb{S}_{A_1, A_2, \cdots, A_n}^{\mathfrak{B}_1, \mathfrak{B}_2, \cdots, \mathfrak{B}_n} \mathfrak{A}(A_1, A_2, \cdots, A_n) = \mathbb{S}_{A_n}^{\mathfrak{B}_n} \left(\cdots \mathbb{S}_{A_2}^{\mathfrak{B}_2} \left(\mathbb{S}_{A_1}^{\mathfrak{B}_1} \mathfrak{A}(A_1, A_2, \cdots, A_n) \right) \cdots \right)$$

where $\mathfrak{B}_1, \mathfrak{B}_2, \cdots, \mathfrak{B}_n$ are propositional (identically true) formulas. Thus, the substitution rule has the form

$$\frac{\mathfrak{A}(A_1, A_2, \cdots, A_n)}{\mathbb{S}_{A_1, A_2, \cdots, A_n}^{\mathfrak{B}_1, \mathfrak{B}_2, \cdots, \mathfrak{B}_n} \mathfrak{A}(A_1, A_2, \cdots, A_n)}$$

The second production rule is applied to a formula which is structured as a parenthesized sequence of implications, that is,

$$\mathfrak{A}_1 \rightarrow (\mathfrak{A}_2 \rightarrow (\cdots (\mathfrak{A}_{n-1} \rightarrow \mathfrak{A}_n) \cdots))$$

and is expressed in the following manner: *if formulas*

$$\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_{n-1} \text{ and}$$
$$\mathfrak{A}_1 \rightarrow (\mathfrak{A}_2 \rightarrow (\cdots(\mathfrak{A}_{n-1} \rightarrow \mathfrak{A}_n)\cdots))$$

are true then formula $\mathfrak{A}_n$ is true in the propositional calculus. The complex rule of inference (modus ponens) is

$$\frac{\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_{n-1}, \quad \mathfrak{A}_1 \rightarrow (\mathfrak{A}_2 \rightarrow (\cdots(\mathfrak{A}_{n-1} \rightarrow \mathfrak{A}_n)\cdots))}{\mathfrak{A}_n}$$

In general, we have the following scheme of inference, where $\mathfrak{P}_i$ are true premises and $\mathfrak{C}_j$ are true conclusions too:

$$\frac{\mathfrak{P}_1, \mathfrak{P}_2, \cdots, \mathfrak{P}_m}{\mathfrak{C}_1, \mathfrak{C}_2, \cdots, \mathfrak{C}_n}$$

In case $m = n$, the senseful inference scheme becomes

$$\frac{\mathfrak{P}_1(\mathfrak{Q}_1, \mathfrak{C}_1), \mathfrak{P}_2(\mathfrak{Q}_2, \mathfrak{C}_2), \cdots, \mathfrak{P}_n(\mathfrak{Q}_n, \mathfrak{C}_n)}{\mathfrak{C}_1, \mathfrak{C}_2, \cdots, \mathfrak{C}_n}$$

where conclusions

$$\mathfrak{C}_1, \mathfrak{C}_2, \cdots, \mathfrak{C}_n$$

are detached from premises

$$\mathfrak{P}_1(\mathfrak{Q}_1, \mathfrak{C}_1), \mathfrak{P}_2(\mathfrak{Q}_2, \mathfrak{C}_2), \cdots, \mathfrak{P}_n(\mathfrak{Q}_n, \mathfrak{C}_n)$$

and, according to the convention of derivation formulas, there is

$$\mathfrak{Q}_1 \vdash \mathfrak{C}_1, \mathfrak{Q}_2 \vdash \mathfrak{C}_2, \cdots, \mathfrak{Q}_n \vdash \mathfrak{C}_n$$

or, even more generally,

$$\mathfrak{Q}_1, \mathfrak{Q}_2, \cdots, \mathfrak{Q}_n \vdash \mathfrak{C}_1, \mathfrak{C}_2, \cdots, \mathfrak{C}_n$$

3.4 A Theorem of Deduction within the Propositional Calculus

Formula $\mathfrak{B}$ is derivable from formulas $\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_n$, that is,

$$\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_n \vdash \mathfrak{B}$$

if it is possible to derive formula $\mathfrak{B}$ merely by means of inference rules, using the initial formulas $\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_n$ and any true formulas within the propositional calculus.

Deduction Theorem. *If formula $\mathfrak{B}$ is derivable from formulas $\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_n$, then*

$$\mathfrak{A}_1 \rightarrow (\mathfrak{A}_2 \rightarrow (\cdots(\mathfrak{A}_n \rightarrow \mathfrak{B})\cdots))$$

is a true formula. □

By induction it is possible to prove the following: if

$$\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_{n-1}, \mathfrak{A}_n \vdash \mathfrak{B}$$

then

$$\mathfrak{A}_1, \mathfrak{A}_2, \cdots, \mathfrak{A}_{n-1} \vdash \mathfrak{A}_n \rightarrow \mathfrak{B}$$

where symbol $\vdash$ is used for marking a derived formula within the calculus.

3.5 Axioms of the Predicate Calculus

If $\mathfrak{M}$ is a set of objects and a, b, c, d are elements of $\mathfrak{M}$, then $P(a)$, $Q(b)$, $R(c, d)$, etc. are propositions concerning objects a, b, c, d. These propositions can be true as well as false.

Expressions $F(x)$, $G(x, y)$, $H(x_1, \cdots, x_n)$, $I(x, x)$, etc. are predicates, that is, functions of arguments belonging to a domain (field) $\mathfrak{M}$. Further, formula $\forall x F(x)$ is a proposition being true if $F(x)$ is true for each element of field $\mathfrak{M}$ and being false in the opposite case. Formula $\exists x F(x)$ is a proposition being true if there exists an element of field $\mathfrak{M}$ for which $F(x)$ is true and being false in the opposite case. $F(x)$ can also be a propositional formula of predicates.

A system of axioms for the predicate calculus is obtained, if to the axiom system I, II, ..., V of subsection 3.2 the following group of axioms is added:

VI. Predicate axioms
   1) $\forall x F(x) \rightarrow F(y)$,
   2) $F(y) \rightarrow \exists x F(x)$

3.6 Inferential Axioms of the Predicate Calculus

The question is how the inferential axioms of the predicate calculus can impact the construction of axioms in the informational calculus. In which respect could the informational entities "for all" and "exists", which belong to quantifiers $\forall$ and $\exists$, respectively, be important at all within the informational calculus? So, let us make merely a short overview to the matter which is treated in detail in [5].

In principle, the inference philosophy of the predicate calculus does not differ essentially from that of the propositional calculus although the situation with quantifiers brings differences which must be considered with special care. As within the propositional calculus, the rules of substitution and inference keep their central roles in the predicate calculus too. To make the difference clear, let us introduce the operator $\rightsquigarrow$ (for 're-place') instead of $\mathbb{S}$ (for 'substitute').

In a predicate formula $\mathfrak{A}$ (formula with predicates), a proposition A or a predicate $F(\cdots)$ can be replaced (substituted) by formula $\mathfrak{B}$. In this case, the substitution is marked by

$$\rightsquigarrow_{A}^{\mathfrak{B}}(\mathfrak{A}) \quad \text{or} \quad \rightsquigarrow_{F(\cdots)}^{\mathfrak{B}(t_1,\cdots,t_n)}(\mathfrak{A})$$

respectively, where A is a variable proposition, F a variable predicate of n variables; formula $\mathfrak{B}(t_1, \cdots, t_n)$ includes among their free variables specially marked variables $t_1, \cdots, t_n$, the number of which is equal to the number of variables of predicate F, that is, n.

Rules connected with quantifiers are the following:

1. *If $\mathfrak{B} \rightarrow \mathfrak{A}(x)$ is a true formula and $\mathfrak{B}$ does not include variable x, then $\mathfrak{B} \rightarrow \forall x\, \mathfrak{A}(x)$ is a true formula too.*
2. *If $\mathfrak{A}(x) \rightarrow \mathfrak{B}$ is a true formula and $\mathfrak{B}$ does not include variable x, then $\exists x\, \mathfrak{A}(x) \rightarrow \mathfrak{B}$ is a true formula too.*

The basic inference scheme (rule of modus ponens) of the predicate calculus remains

$$\frac{\begin{array}{c}\mathfrak{A}\\ \mathfrak{A} \rightarrow \mathfrak{B}\end{array}}{\mathfrak{B}}$$

According to [3], various rules for the predicate calculus can be derived by means of the substitution and inference rule considering the basic axioms. Such a scheme is, for instance,

$$\frac{\begin{array}{c}\mathfrak{A} \rightarrow \mathfrak{B}\\ \mathfrak{A} \rightarrow \mathfrak{C}\end{array}}{\mathfrak{A} \rightarrow \mathfrak{B} \wedge \mathfrak{C}}$$

or, in a general form, at given variables $\mathfrak{a}, \mathfrak{b}, \cdots, \mathfrak{k}$

$$\frac{\begin{array}{c}\mathfrak{A} \rightarrow \mathfrak{B}(\mathfrak{a})\\ \mathfrak{A} \rightarrow \mathfrak{B}(\mathfrak{b})\\ \vdots\\ \mathfrak{A} \rightarrow \mathfrak{B}(\mathfrak{k})\end{array}}{\mathfrak{A} \rightarrow \mathfrak{B}(\mathfrak{a}) \wedge \mathfrak{B}(\mathfrak{b}) \wedge \cdots \wedge \mathfrak{B}(\mathfrak{k})}$$

Another inference scheme, with the universal quantifier, being important for a later consideration could be

$$\frac{\mathfrak{A} \rightarrow \mathfrak{B}(a)}{\mathfrak{A} \rightarrow \forall x\, \mathfrak{B}(x)}$$

where x must not appear in $\mathfrak{B}(a)$. The next inference scheme which comes out is, for instance,

$$\frac{\begin{array}{c}\mathfrak{A} \rightarrow (\mathfrak{B} \rightarrow \mathfrak{U})\\ \mathfrak{A} \rightarrow (\mathfrak{B} \rightarrow \mathfrak{V})\end{array}}{\mathfrak{A} \rightarrow (\mathfrak{B} \rightarrow \mathfrak{U} \wedge \mathfrak{V})}$$

This scheme can be extended by a transition from a two-part conjunction to the value domain of a variable a, that is,

$$\frac{\mathfrak{A} \rightarrow (\mathfrak{B} \rightarrow \mathfrak{C}(a))}{\mathfrak{A} \rightarrow (\mathfrak{B} \rightarrow \forall x\, \mathfrak{C}(x))}$$

where in the premise x must not appear.

Analogously, for the existential quantifier, the scheme

$$\frac{\mathfrak{B}(a) \rightarrow \mathfrak{A}}{\exists x\, \mathfrak{B}(x) \rightarrow \mathfrak{A}}$$

can be derived. This scheme corresponds to the disjunction scheme

$$\frac{\begin{array}{c}\mathfrak{A} \rightarrow \mathfrak{C}\\ \mathfrak{B} \rightarrow \mathfrak{C}\end{array}}{\mathfrak{A} \vee \mathfrak{B} \rightarrow \mathfrak{C}}$$

etc. The listed inference schemes will be used for comparison with schemes of informational calculus.

3.7 Theorems of Deduction within the Predicate Calculus

The basic theorem of deduction in the predicate calculus is similar to the theorem in the propositional calculus. But there are several theorems concerning formulas with quantifiers.

Deduction Theorems for Predicates. *If formula $\mathfrak{B}$ is derivable from formula $\mathfrak{A}$, then formula*

$$\mathfrak{A} \rightarrow \mathfrak{B}$$

is derivable in the predicate calculus, that is,

$$\vdash \mathfrak{A} \rightarrow \mathfrak{B}$$

Further, for formulas including universal and existential quantifiers, there is

$$\begin{array}{l}
\vdash \forall x F(x) \rightarrow \exists x F(x), \\
\forall x \forall y F(x,y) \equiv \forall y \forall x F(x,y), \\
\exists x \forall y F(x,y) \rightarrow \forall y \exists x F(x,y), \\
\vdash \forall x (F(x) \rightarrow G(x)) \rightarrow (\forall x F(x) \rightarrow \forall x G(x)), \\
\vdash \forall x (F(x) \rightarrow G(x)) \rightarrow (\exists x F(x) \rightarrow \exists x G(x)), \\
\vdash \forall x (F(x) \equiv G(x)) \rightarrow (\forall x F(x) \equiv \forall x G(x)), \\
\exists x F(x) \equiv \overline{\forall x \overline{F(x)}},\ \exists x \overline{F(x)} \equiv \overline{\forall x F(x)}, \\
\overline{\exists x F(x)} \equiv \forall x \overline{F(x)},\ \overline{\exists x \overline{F(x)}} \equiv \forall x F(x)
\end{array}$$

where $\mathfrak{A} \equiv \mathfrak{B}$ represents the formula

$$(\mathfrak{A} \rightarrow \mathfrak{B}) \wedge (\mathfrak{B} \rightarrow \mathfrak{A})$$

□

Proofs for the listed theorems can be found in [5].

4 Axioms within an Informational Theory

4.1 Introduction

The question is how could the axioms of the propositional and predicate calculus be turned over to informational calculus. The main difference seems to be between the realm of truth in the traditional logic and realm of information in the informational logic. The informational realm is substantially broader and within it the truth appears only as a very particular case. Instead to say that formulas in the traditional logic are true or false, in the informational logic formulas can inform in one or another way, truly and falsely, particularly and universally, in parallel and serially, straightforwardly and circularly, algorithmically and spontaneously, programmingly and intelligently, routinely and creatively, logically (consistently) and controversially, etc. Informational logic becomes an active part of any informational system, the theoretical and the practical one.

4.2 Axioms of the Informational Calculus

Axioms of informational calculus can find a logical support in axioms of propositional and predicate calculus (Subsections 3.2 and 3.5, respectively). At the first glance, the axiom designer can behave in a withholding way, considering the traditional axioms as much as possible. But, thereupon, when getting the appropriate experience, the designer of an informational theory can go his/her own way, considering the entirety of the possible informational realm. Within this general scope, the true as a particular situation can be replaced by the informational as an extreme attitude of the informationally possible. The discussion in this section will follow the traditional way of axiom construction as much as possible. In the next section the most general and informationally open way will replace the traditional thinking.

First, let us classify the informational axioms according to the tradition in the propositional and predicate calculus. This way of interpretation will give us the necessary feeling of difference and informational generality in respect to the usual understanding of "logical" axioms. It will be possible to recognize the essential difference which governs the informational realm in its universality in comparison to the classical logical (propositional, predicate) realm. It certainly does not mean that the particular axiom situation in logical calculuses does not fit the informational principles—it fits them in a particular manner.

4.2.1 "Implication" Axioms of the Informational Calculus

In case of implicational axioms, we are concerned with two basic possibilities. At the beginning of the axiomatizing process we are concerned with the so-called informational phenomenalism for which we have to design particular axioms giving us the certainty of our initial steps into a general theory of the informational. This means, we have to explain in a formally consistent way the arising of initial axioms themselves. For instance, we must induce the primitive axioms of externalism, internalism and metaphysicalism, which constitute the very general axiom of informational phenomenalism.

In Subsection 3.2 we listed the group of axioms of implication. In informational calculus, we have a broader definition of the so-called informational implication, discussed in [10]. The question is where to begin the process of axiomatization in informational calculus. We are confronted with the principal difference existing between the truth in logical calculuses and the informing in informational calculus.

In the traditional logic, propositions and predicates inform in a true or false manner. In the informational calculus, informational entities—precisely informational operands—inform and are informed. Thus, we can choose the operand—informational entity—as the point from which one can begin the process of axiomatization. We shall see how different *initial* axioms will become possible and how they will be circularly interweaved. This axiomatic analysis will deepen the understanding of the informational phenomenalism, that is, the phenomenalism of the informational entity.

According to Subsection 3.2, axioms I. 1–3, in informational case, the following is obtained:

I. Axioms of informational implication

$$\begin{array}{ll} 1) & \alpha \Longrightarrow (\beta \Longrightarrow \alpha), \\ 2) & (\alpha \Longrightarrow (\alpha \Longrightarrow \beta)) \Longrightarrow (\alpha \Longrightarrow \beta), \\ 3) & (\alpha \Longrightarrow \beta) \Longrightarrow ((\beta \Longrightarrow \gamma) \Longrightarrow (\alpha \Longrightarrow \gamma)) \end{array}$$

where α and β are informational operands (entities) and $\Longrightarrow$ represents the operator of informational implication (in the most general and complex form [10]).

Prior to the informational systemization of implication axioms, let us look at examples which bring to the surface the logical sense of the listed implication axioms within the informational realm.

An axiom of the form (I.1)

$$\alpha \Longrightarrow ((\alpha \models) \Longrightarrow \alpha) \tag{1}$$

seems to be regular. It means that an informational entity α simply implies that entity α is implied by informing of the entity, that is, by $\alpha \models$. Theoretically, it seems to be impossible to oppose such an initial principle because the informational nature of an entity is circular in respect to itself and its informing. If introducing α's informing in the form $\mathfrak{I}_\alpha$, the last axiom can be expressed in the form

$$\alpha \Longrightarrow (\mathfrak{I}_\alpha \Longrightarrow \alpha) \tag{2}$$

This form of the axiom is more general then the preceding one since it presupposes also the phenomenon of informational internalism, that is,

$$\alpha \Longrightarrow ((\models \alpha) \Longrightarrow \alpha) \tag{3}$$

However, there is not an equivalence between the first (1) and the third (3) axiom on one side and the second (2) axiom on the other side.

A resulting system of axioms concerning informational phenomenalism in the sense of the propositional axiom (I.1) in Subsection 3.2 is

$$\begin{array}{l} \alpha \Longrightarrow (\beta \Longrightarrow \alpha); \\ \beta \in \{\alpha \models, \models \alpha, \alpha \models \alpha, (\alpha \models; \models \alpha)\} \end{array}$$

or informationally explicitly

$$\alpha \Longrightarrow \left(\left(\beta \in \left\{ \begin{array}{l} \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \begin{pmatrix} \alpha \models; \\ \models \alpha \end{pmatrix} \end{array} \right\} \right) \Longrightarrow \alpha \right)$$

or simply and evidently

$$\alpha \Longrightarrow \left(\left(\begin{array}{l} \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \begin{pmatrix} \alpha \models; \\ \models \alpha \end{pmatrix} \end{array} \right) \Longrightarrow \alpha \right)$$

In this formula, the essential difference between "informational operators" *comma* and *semicolon* must be distinguished (the alternative and the parallel operator, respectively).

An extension of the discussed axiom system is evidently the following:

$$\left(\begin{array}{l} \alpha; \\ \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \begin{pmatrix} \alpha \models; \\ \models \alpha \end{pmatrix} \end{array} \right) \Longrightarrow \left(\left(\begin{array}{l} \alpha; \\ \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \begin{pmatrix} \alpha \models; \\ \models \alpha \end{pmatrix} \end{array} \right) \Longrightarrow \left(\begin{array}{l} \alpha; \\ \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \begin{pmatrix} \alpha \models; \\ \models \alpha \end{pmatrix} \end{array} \right) \right)$$

An example of this system is not only the axiom

$$(\alpha \models) \Longrightarrow ((\models \alpha) \Longrightarrow (\alpha \models))$$

which fits the general axiomatic scheme (I.1), but also

$$(\alpha \models) \Longrightarrow ((\models \alpha) \Longrightarrow (\alpha \models \alpha))$$

if arbitrary items of alternative array (α, $\alpha \models$, $\models \alpha$, $\alpha \models \alpha$, $(\alpha \models; \models \alpha)$) are chosen. It is to stress that cases of non-informing, that is, $\alpha \not\models$, $\not\models \alpha$, $\alpha \not\models \alpha$, etc. are understood as particular cases of informing.

Informational axiom (I.2) offers other significant formulas (basic informational implications) which are used, for instance, in informational modus ponens and modus tollens.

One of the most important informational axioms, following the scheme (I.2), seems to be

$$(\alpha \Longrightarrow (\alpha \Longrightarrow (\alpha \models))) \Longrightarrow (\alpha \Longrightarrow (\alpha \models))$$

which delivers the necessary conclusion $\alpha \Longrightarrow (\alpha \models)$ needed in various rules of inference.

A resulting system of axioms concerning informational phenomenalism in the sense of the propositional axiom (I.2) in Subsection 3.2 is

$$(\alpha \Longrightarrow (\alpha \Longrightarrow \beta)) \Longrightarrow (\alpha \Longrightarrow \beta);$$
$$\beta \in \{\alpha \models, \models \alpha, \alpha \models \alpha, (\alpha \models; \models \alpha)\}$$

or informationally explicitly

$$\left(\alpha \Longrightarrow \left(\alpha \Longrightarrow \left(\beta \in \left\{\begin{array}{l} \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right\}\right)\right)\right) \Longrightarrow$$
$$\left(\alpha \Longrightarrow \left(\beta \in \left\{\begin{array}{l} \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right\}\right)\right)$$

One can imagine what happens if the first β and the second β in the last formula are chosen as different entities which, in the last scheme, is possible in several ways. Appearances of β within the alternative scheme (set) are legal independently of the randomly chosen element in each concrete case. It is to mention that some alternative informational schemes concerning the axiomatic attitude will also be discussed in Subsection 4.2.2.

This particular axiomatic situation can now be transformed simply and evidently into a more general axiomatic scheme of the form

$$\left(\alpha \Longrightarrow \left(\alpha \Longrightarrow \left(\begin{array}{l} \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right)\right)\right) \Longrightarrow$$
$$\left(\alpha \Longrightarrow \left(\begin{array}{l} \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right)\right)$$

The most general extension of the discussed axiom system could evidently be the following:

$$\left(\left(\begin{array}{l} \alpha; \\ \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right) \Longrightarrow \left(\left(\begin{array}{l} \alpha; \\ \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right) \Longrightarrow \left(\begin{array}{l} \alpha; \\ \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right)\right)\right)$$
$$\Longrightarrow \left(\left(\begin{array}{l} \alpha; \\ \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right) \Longrightarrow \left(\begin{array}{l} \alpha, \\ \alpha \models, \\ \models \alpha, \\ \alpha \models \alpha, \\ \left(\begin{array}{l} \alpha \models; \\ \models \alpha \end{array}\right) \end{array}\right)\right)$$

A characteristic and progressively diverse form of the last axiom system would be, for example,

$$(\alpha \Longrightarrow ((\alpha \models) \Longrightarrow (\models \alpha))) \Longrightarrow$$
$$((\alpha \models \alpha) \Longrightarrow (\alpha \models; \models \alpha))$$

which shows a phenomenalistic sequence extending in a straight implicative manner from informational entity α over its externalism $\alpha \models$ and internalism $\models \alpha$ to its metaphysicalism $\alpha \models \alpha$ and phenomenalism $\alpha \models; \models \alpha$.

Axiomatic scheme (I.3) delivers a very fundamental property of informing of two entities concerning the third entity. Such axiom is, for instance,

$$(\alpha \Longrightarrow (\alpha \models)) \Longrightarrow$$
$$(((\alpha \models) \Longrightarrow (\models \alpha)) \Longrightarrow (\alpha \Longrightarrow (\models \alpha)))$$

or also the pair of axioms

$$((\alpha \models) \Longrightarrow (\models \alpha)) \Longrightarrow$$
$$(((\models \alpha) \Longrightarrow (\alpha \models \alpha)) \Longrightarrow ((\alpha \models) \Longrightarrow (\alpha \models \alpha)));$$

$$((\alpha \models) \Longrightarrow (\models \alpha)) \Longrightarrow$$
$$(((\models \alpha) \Longrightarrow (\alpha \models; \models \alpha)) \Longrightarrow ((\alpha \models) \Longrightarrow$$
$$(\alpha \models; \models \alpha)))$$

The general scheme for the implicative rule (I.3) becomes

$$(\alpha \Longrightarrow \beta) \Longrightarrow ((\beta \Longrightarrow \gamma) \Longrightarrow (\alpha \Longrightarrow \gamma));$$
$$\beta,\gamma \in \{\alpha \models, \models \alpha, \alpha \models \alpha, (\alpha \models; \models \alpha)\}$$

etc., in the sense of the previous examples (I.1) and (I.2). Thus, the informational version of the last system becomes

$$\left(\alpha \Longrightarrow \left(\beta \in \left\{\begin{matrix}\alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{matrix}\right\}\right)\right) \Longrightarrow$$
$$\left(\left(\left(\beta \in \left\{\begin{matrix}\alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{matrix}\right\}\right) \Longrightarrow \left(\gamma \in \left\{\begin{matrix}\alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{matrix}\right\}\right)\right)\right.$$
$$\left.\Longrightarrow \left(\alpha \Longrightarrow \left(\gamma \in \left\{\begin{matrix}\alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{matrix}\right\}\right)\right)\right)$$

The most general axiomatic version of the last system would be

$$\left(\begin{pmatrix}\alpha,\\ \alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{pmatrix} \Longrightarrow \begin{pmatrix}\alpha,\\ \alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{pmatrix}\right) \Longrightarrow$$
$$\left(\left(\begin{pmatrix}\alpha,\\ \alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{pmatrix} \Longrightarrow \begin{pmatrix}\alpha,\\ \alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{pmatrix}\right) \Longrightarrow\right.$$
$$\left.\left(\begin{pmatrix}\alpha,\\ \alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{pmatrix} \Longrightarrow \begin{pmatrix}\alpha,\\ \alpha \models,\\ \models \alpha,\\ \alpha \models \alpha,\\ \begin{pmatrix}\alpha \models;\\ \models \alpha\end{pmatrix}\end{pmatrix}\right)\right)$$

Evidently, from this axiomatic scheme, a "complete" implicative circular formula proceeds:

$$(\alpha \Longrightarrow (\alpha \models)) \Longrightarrow$$
$$(((\models \alpha) \Longrightarrow (\alpha \models \alpha)) \Longrightarrow ((\alpha \models; \models \alpha) \Longrightarrow \alpha))$$

This formula is in no way an impossible speculation since all operands are only different phenomenal forms of one and the same informational entity α (with exception of the first and the last operand which are equal).

4.2.2 Another Form of "Implication" Axioms of the Informational Calculus

Informational calculus bases on the informing of entities and not solely on the logical truth. As the reader can observe, the discussed propositional and predicate axioms are always identically true logical formulas. We can show how other informational axioms which do not base on propositional and predicate axioms can be derived intuitively, trivially and formally in the same manner as the preceding informational axioms.

So let us take the syntactically rearranged axiom (I.1) in the form

$$(\alpha \Longrightarrow \beta) \Longrightarrow \alpha$$

instead of $\alpha \Longrightarrow (\beta \Longrightarrow \alpha)$. Propositional formula $(A \rightarrow B) \rightarrow A$ is not an identically true formula and its value is A. Axiom (I.1) is called *the axiom of the consequent determination.* What could the rearranged axiom mean at all? Let us check its meaning by some basic examples.

Let us look the semantic difference between formulas

$$\alpha \Longrightarrow ((\alpha \models) \Longrightarrow \alpha) \quad \text{and} \quad (\alpha \Longrightarrow (\alpha \models)) \Longrightarrow \alpha$$

If the first formula says that any informational entity α implies its externalistic informing $\alpha \models$ as a reason of itself, the second formula stresses that any informational entity α is implied by an implication in which entity α implies its externalistic informing $\alpha \models$. The reader will agree that it is practically impossible to argue against this argumentation. In informational cases we do strictly distinguish between circular and serially noncircular cases. As soon as $\alpha \models$ appears within the cycle $\alpha \Longrightarrow ((\alpha \models) \Longrightarrow \alpha)$, the transition $\alpha \Longrightarrow (\alpha \models)$ represents a part of the "whole history" of the cycle and, therefore, can or must be considered within the form $(\alpha \Longrightarrow (\alpha \models)) \Longrightarrow \alpha$. Thus, another substantial implication comes to the surface:

$$(\alpha \Longrightarrow ((\alpha \models) \Longrightarrow \alpha)) \Longrightarrow ((\alpha \Longrightarrow (\alpha \models)) \Longrightarrow \alpha)$$

Possibly, this fact becomes evident by the formula

$$(\alpha \Longrightarrow \mathfrak{I}_\alpha) \Longrightarrow \alpha$$

where $\mathfrak{I}_\alpha$ expresses the entire phenomenalistic nature of α's informing, that is, its hermeneutics (a kind of regular interpretation of α's history) which considers not only the history of both appearing components α and $\mathfrak{I}_\alpha$, but also transition $\alpha \Longrightarrow \mathfrak{I}_\alpha$ (or $\mathfrak{I}_\alpha \Longrightarrow \alpha$ in the first case).

4.2.3 "Conjunction" Axioms of the Informational Calculus

Let us draw an informational parallel to the propositional conjunction axioms (II.1–3) in Subsection 3.2.

What could be the conjunction in an informational sense? How could it be generalized?

The logical "and", represented by operator $\wedge$, means also "and simultaneously" or "in parallel". Informational operator of parallelism is $\Vert\!\models$ or, commonly, semicolon ';'. Thus, for the first axiom of conjunction (II.1) there is, informationally,

$$(\alpha \Vert\!\models \beta) \Longrightarrow \alpha \quad \text{or} \quad \begin{pmatrix} \alpha; \\ \beta \end{pmatrix} \Longrightarrow \alpha \ .$$

and for the second axiom of conjunction (II.2),

$$(\alpha \Vert\!\models \beta) \Longrightarrow \beta \quad \text{or} \quad \begin{pmatrix} \alpha; \\ \beta \end{pmatrix} \Longrightarrow \beta$$

For the third axiom of conjunction (II.3) there is

$$(\alpha \Longrightarrow \beta) \Longrightarrow ((\alpha \Longrightarrow \gamma) \Longrightarrow (\alpha \Longrightarrow (\beta \Vert\!\models \gamma)))$$

or, in a common informational form,

$$(\alpha \Longrightarrow \beta) \Longrightarrow \left((\alpha \Longrightarrow \gamma) \Longrightarrow \left(\alpha \Longrightarrow \begin{pmatrix} \beta; \\ \gamma \end{pmatrix}\right)\right)$$

The sense of parallelism axioms is significant in cases of the so-called informational decomposition (see Section 7). Two characteristic cases are, for example,

$$\begin{pmatrix} \alpha; \\ \alpha \models \end{pmatrix} \Longrightarrow \alpha; \quad \begin{pmatrix} \alpha; \\ \alpha \models \end{pmatrix} \Longrightarrow (\alpha \models)$$

4.2.4 "Disjunction" Axioms of the Informational Calculus

Disjunction axioms (III.1–3) in Subsection 3.2 introduce the meaning of informational formula $\alpha_1, \alpha_2, \cdots, \alpha_n$ where commas are used instead of semicolons. What does, in an informational formula, a comma mean at all?

Informationally, propositional axiom (III.1) can be interpreted as

$$\alpha \Longrightarrow (\alpha, \beta)$$

Instead of propositional formula $A \vee B$ there is informationally, simply α, β where the last formula just lists two alternatives which are α and β. Alternatives are informational entities which can be chosen by an informational system.

From which point of the philosophy could the discussed alternativeness come from? In an externalistic case, $\alpha \models$, intuitively, a presumption $\alpha \models \beta$ exists, otherwise the externalism of α would not perform meaningfully. One can agree that from the process represented by formula $\alpha \models \beta$ operands α and β can be listed. Thus,

$$(\alpha \models \beta) \Longrightarrow (\alpha, \beta) \quad \text{and} \quad (\alpha \models) \Longrightarrow (\alpha, \beta)$$

Through this demonstration the step towards the "disjunction" axiom in informational calculus becomes evident.

According to propositional axiom (III.2), there is

$$\beta \Longrightarrow (\alpha, \beta)$$

which is another form of the first disjunction informational axiom. It is not meant that the list α, β is ordered.

Finally, according to propositional axiom (III.3), we have

$$(\alpha \Longrightarrow \gamma) \Longrightarrow ((\beta \Longrightarrow \gamma) \Longrightarrow (\alpha, \beta \Longrightarrow \gamma))$$

In the last axiom we have to explain formula $\alpha, \beta \Longrightarrow \gamma$ additionally; the meaning is the following:

$$(\alpha, \beta \Longrightarrow \gamma) \rightleftharpoons \begin{pmatrix} \alpha \Longrightarrow \gamma; \\ \beta \Longrightarrow \gamma \end{pmatrix} \quad \text{and}$$

$$(\alpha, \beta) \rightleftharpoons \{\alpha, \beta\}$$

Thus, a list α, β connected with an operator in a formula results into a parallel system. In general,

$$(\alpha, \beta \models \gamma) \rightleftharpoons \begin{pmatrix} \alpha \models \gamma; \\ \beta \models \gamma \end{pmatrix} \quad \text{and}$$

$$(\alpha, \beta \models) \rightleftharpoons \begin{pmatrix} \alpha \models; \\ \beta \models \end{pmatrix}$$

This short discussion rounds up the possibilities of construction to the propositional disjunctive axioms parallel informational axioms.

4.2.5 "Equivalence" Axioms of the Informational Calculus

The reader might observe that we have never defined a rigorous informational formula by which informational operator of implication, $\Longrightarrow$, would be defined once for all. In [10], only verbal possibilities of the informational contents of implication have been listed. On the other hand, propositional implication, $\rightarrow$, is defined by a kind of logical table once for all. Does it mean that the contents (not a rough definition itself) of informational implication changes (arises) from case to case? The answer might be the following: operator $\Longrightarrow$ is a regular informational operator which, from case to case, can (must) be particularized and universalized, according to the involved operands. So, it must be interpreted only to the sufficient informational extent. The reader can imagine how complex and never ending interpretation of informational implication would proceed from the basis of its verbal (dictionary-like) determinations.

This fact does not restrict the discussion of informational equivalence in which informational implication plays an essential role. Thus, we can put the question of informational equivalence in parallel to the existing notion of propositional calculus.

According to the axiom group (IV.1–3) in Subsection 3.2, the adequate informational axioms concerning informational equivalence are the following:

$$\begin{array}{l} (\alpha \Longleftrightarrow \beta) \Longrightarrow (\alpha \Longrightarrow \beta); \\ (\alpha \Longleftrightarrow \beta) \Longrightarrow (\beta \Longrightarrow \alpha); \\ (\alpha \Longrightarrow \beta) \Longrightarrow ((\beta \Longrightarrow \alpha) \Longrightarrow (\alpha \Longleftrightarrow \beta)) \end{array}$$

Informational operator of equivalence, $\Longleftrightarrow$, can be defined dependently on informational implication, that is, according to deduction theorems of predicate calculus in Subsection 3.7. In this sense,

$$(\alpha \Longleftrightarrow \beta) \rightleftharpoons_{\text{Def}} ((\alpha \Longrightarrow \beta; \beta \Longrightarrow \alpha))$$

Thus, in each particular case, when particularizing operator $\Longleftrightarrow$, this particularization depends on the particularization of implicative operator $\Longrightarrow$.

4.2.6 "Negation" Axioms of the Informational Calculus

The purpose of the negation of a statement in propositional calculus is to obtain the true negated statement, that is, the true otherwise false statement. In informational calculus, an entity informs or can be informed in a certain manner, but not in another one. This last situation can be discussed in the framework of an entity's non-informing.

Thus, an informational equivalent to the propositional situation $\overline{A}$ is, for example, $\alpha \not\models$ and $\not\models \alpha$ which reads α does not inform (in a certain manner) and α is not informed (in a certain manner), respectively. Additionally, operator $\not\models$ represents only a particular case of operator $\models$.

For a double negation $\overline{\overline{A}}$ it is possible to distinguish different cases of double non-informing, for instance,

$$(\alpha \not\models) \not\models; \quad \not\models (\alpha \not\models); \quad (\not\models \alpha) \not\models; \quad \not\models (\not\models \alpha)$$

Considering the axiom group of negation (V.1–3) in propositional calculus (Subsection 3.2), there is, informationally,

$$\begin{array}{l} (\alpha \Longrightarrow \beta) \Longrightarrow ((\beta \not\models; \not\models \beta) \Longrightarrow (\alpha \not\models; \not\models \alpha)); \\ \alpha \Longrightarrow ((\alpha \not\models; \not\models \alpha) \not\models; \not\models (\alpha \not\models; \not\models \alpha)); \\ ((\alpha \not\models; \not\models \alpha) \not\models; \not\models (\alpha \not\models; \not\models \alpha)) \Longrightarrow \alpha \end{array}$$

However, to these axioms, characteristic informational axioms concerning the phenomenon of non-informing can be adopted. For example,

$$\begin{array}{l} (\alpha \Longrightarrow (\alpha \models)) \Longrightarrow (\alpha \Longrightarrow (\alpha \not\models)); \\ (\alpha \Longrightarrow (\models \alpha)) \Longrightarrow (\alpha \Longrightarrow (\not\models \alpha)); \\ (\alpha \Longrightarrow (\alpha \models \alpha)) \Longrightarrow (\alpha \Longrightarrow (\alpha \not\models \alpha)); \\ (\alpha \Longrightarrow (\alpha \models; \models \alpha)) \Longrightarrow (\alpha \Longrightarrow (\alpha \not\models; \not\models \alpha)) \end{array}$$

etc. concern operator particularization in the processes of decomposition.

4.2.7 "Predicate" Axioms and Theorems of the Informational Calculus

The universal and existential quantifier do not play a significant role in the informational calculus. They are rather very uncommon entities within the informational realm. For instance, operator $\models_{\forall_\alpha}$ reads *informs (is informed) for all α's* and formulas

$$\begin{array}{lll}
\alpha \models_{\mathrm{for\_all\_}\alpha} & \text{or} & \alpha \models_{\forall_\alpha}; \\
\models_{\mathrm{for\_all\_}\alpha} \alpha & \text{or} & \models_{\forall_\alpha} \alpha; \\
\alpha \models_{\mathrm{for\_all\_}\alpha} \alpha & \text{or} & \alpha \models_{\forall_\alpha} \alpha; \\
\alpha \models_{\mathrm{for\_all\_}\alpha}; \models_{\mathrm{for\_all\_}\alpha} \alpha & \text{or} & \alpha \models_{\forall_\alpha}; \models_{\forall_\alpha} \alpha
\end{array}$$

are very special cases since it is not clear to which informational realm the 'all' could refer. A similar, problematic situation occurs in the case of the existential quantifier where formulas

$$\begin{array}{lll}
\alpha \models_{\mathrm{exist\_}\alpha} & \text{or} & \alpha \models_{\exists_\alpha}; \\
\models_{\mathrm{exist\_}\alpha} \alpha & \text{or} & \models_{\exists_\alpha} \alpha; \\
\alpha \models_{\mathrm{exist\_}\alpha} \alpha & \text{or} & \alpha \models_{\exists_\alpha} \alpha; \\
\alpha \models_{\mathrm{exist\_}\alpha}; \models_{\mathrm{exist\_}\alpha} \alpha & \text{or} & \alpha \models_{\exists_\alpha}; \models_{\exists_\alpha} \alpha
\end{array}$$

Analogously to the predicate axiom group (VI.1–2) in Subsection 3.5, the adequate informational axioms could take the form

$$\begin{array}{c}
(\alpha \models_{\forall_\alpha} \varphi(\alpha)) \Longrightarrow \varphi(\beta); \\
\varphi(\beta) \Longrightarrow (\alpha \models_{\exists_\alpha} \varphi(\alpha))
\end{array}$$

The reader can imagine how the operator attribute *for all α's* in an informational realm causes that the informational function φ is informationally impacted by β which is within the realm of *all* α, etc. On the other hand, informational function $\varphi(\beta)$ (as determined in [9]) implies the existence of function φ being dependent on β, etc.

In parallel to the deduction theorems for predicates in Subsection 3.7 it is possible to derive "similar" informational theorems for informational entities.

Decomposition Theorems of Informational Entities. *If formula β is informationally derivable from formula α, then formula*

$$\alpha \Longrightarrow \beta$$

is derivable in the informational calculus, that is,

$$\twoheadrightarrow(\alpha \Longrightarrow \beta)$$

The theorems of predicate calculus in Subsection 3.7 become, informationally,

$$\begin{array}{l}
\twoheadrightarrow((\alpha \models_{\forall_\alpha} \varphi(\alpha)) \Longrightarrow (\alpha \models_{\exists_\alpha} \varphi(\alpha))); \\
(\alpha \models_{\forall_\alpha} (\beta \models_{\forall_\beta} \varphi(\alpha,\beta))) \Longleftrightarrow \\
\qquad (\beta \models_{\forall_\beta} (\alpha \models_{\forall_\alpha} \varphi(\alpha,\beta))); \\
(\alpha \models_{\exists_\alpha} (\beta \models_{\forall_\beta} \varphi(\alpha,\beta))) \Longleftrightarrow \\
\qquad (\beta \models_{\forall_\beta} (\alpha \models_{\exists_\alpha} \varphi(\alpha,\beta))); \\
\twoheadrightarrow((\alpha \models_{\forall_\alpha} (\varphi(\alpha) \Longrightarrow \psi(\alpha))) \Longrightarrow \\
\qquad ((\alpha \models_{\forall_\alpha} \varphi(\alpha)) \Longrightarrow (\alpha \models_{\forall_\alpha} \psi(\alpha)))); \\
\twoheadrightarrow((\alpha \models_{\forall_\alpha} (\varphi(\alpha) \Longrightarrow \psi(\alpha))) \Longrightarrow \\
\qquad ((\alpha \models_{\exists_\alpha} \varphi(\alpha)) \Longrightarrow (\alpha \models_{\exists_\alpha} \psi(\alpha)))); \\
\twoheadrightarrow((\alpha \models_{\forall_\alpha} (\varphi(\alpha) \Longleftrightarrow \psi(\alpha))) \Longrightarrow \\
\qquad ((\alpha \models_{\forall_\alpha} \varphi(\alpha)) \Longleftrightarrow (\alpha \models_{\forall_\alpha} \psi(\alpha)))); \\
(\alpha \models_{\exists_\alpha} \varphi(\alpha)) \Longleftrightarrow (\alpha \not\models_{\forall_\alpha} (\varphi(\alpha) \not\models; \not\models \varphi(\alpha))); \\
(\alpha \models_{\exists_\alpha} (\varphi(\alpha) \not\models; \not\models \varphi(\alpha))) \Longleftrightarrow \\
\qquad (\alpha \not\models_{\forall_\alpha} (\varphi(\alpha) \not\models; \not\models \varphi(\alpha))); \\
(\alpha \models_{\exists_\alpha} (\varphi(\alpha) \not\models; \not\models \varphi(\alpha))) \Longleftrightarrow \\
\qquad (\alpha \models_{\forall_\alpha} (\varphi(\alpha) \not\models; \not\models \varphi(\alpha))); \\
(\alpha \models_{\exists_\alpha} (\varphi(\alpha) \not\models; \not\models \varphi(\alpha))) \Longleftrightarrow (\alpha \models_{\forall_\alpha} \varphi(\alpha))
\end{array}$$

□

These theorems can cause a retrograde and logically more critical understanding of the predicate theorems listed in Subsection 3.7.

4.3 The Variety of Informational Axioms of Inference

One of the basic questions concerning the inference schemes in different calculuses is in which manner inference rules are constructed and accepted by scientific communities. Obviously, the way to various inference rules leads through the understanding of the basic axioms treated in the preceding subsections. An inference rule is nothing else than a means for the derivation (deduction) process where it is used for a constructive transformation of theory-legal formulas into new formulas. The next step concerning axioms and inference rules is the introduction of the so-called replacement rules by which formulas can be transformed automatically, e.g. in a machine-like fashion.

Informational rules of inference function like axioms and only by them it is permitted to conclude in a theory-consistent manner. E.g., rules of substitution and modus ponens belong to the most obvious and widely accepted rules for formula transformation within a deduction process. On the other hand, informational inference rules can cover informationally unlimited possibilities and they can come into existence together with specific and particular informational problems.

The so-called modi informationis [6] can be grouped and expressed in several possible ways. We have the following possibilities:

1. *Modus ponens* is the most obvious inference rule in mathematics in the realm of truth. Informational modus ponens follows the general (phenomenalistic) principle of informing where the mathematical *to be true* is replaced by the informational *to inform.* This does not mean that the true is excluded, it only appears as a particular case of the informational. In this sense, informational modus ponens as an inference rule retains the basic logical form which is

$$\frac{\alpha; (\alpha \Longrightarrow \beta)}{\beta}$$

and where the implicational operator $\Longrightarrow$ has a broadened meaning in comparison with the logical implication operator $\rightarrow$, determined, for example, by a logical table. Implicit informational operator of parallelism ';' which simultaneously performs as a formula separator replaces the logical operator $\wedge$ (also, a comma in the traditional inference rule).

Particularizations of modus ponens can appear as special rules called, for example, modus procedendi, modus operandi and others.

2. *Modus tollens* is a rule of a reverse inferring in respect to the implication formula of its premise. There is a truly substantial difference between the mathematical and informational modus tollens. In the domain of truth, the falsity is the only essential counterpart to the truth and, so, the operator of negation of a formula becomes truly essential. Through the introduction of negation it becomes possible to express the falsity of a statement as its truth (e.g., identically true falsity).

In informational modus tollens the negation is replaced by the operator of noninforming (in a certain way), denoted by $\not\models$. The reader can now feel the dramatic difference existing between a static state of negation and the dynamic state of noninforming concerning a logical and an informational entity, respectively. Thus, the informational modus tollens in comparison with the logical one keeps the form

$$\frac{(\alpha \Longrightarrow \beta); (\beta \not\models; \not\models \beta)}{\alpha \not\models; \not\models \alpha}$$

with already mentioned differences concerning the modus ponens.

3. *Modus rectus* is not a mathematical inference rule and its origin can be searched in the analysis of the Latin speech. The aim of this rule is the filtering-out (detaching) of the intention marked by $\iota_{\text{intention}}(\alpha)$, hidden in entity represented by α through the informing of α, e.g. to an entity represented by β. It is understood that this intention is an informational function of α which is reflected in β to which α informs intentionally (or in an intending manner, that is, 'intendingly'). In this sense, a possibility of informational modus rectus becomes

$$\frac{\alpha; ((\alpha \models_{\text{intend}} \beta) \Longrightarrow \iota_{\text{intention}}(\alpha))}{(\iota_{\text{intention}}(\alpha) \models; \models \iota_{\text{intention}}(\alpha)) \subset \beta}$$

One can certainly construct 'intentional' rules where the intention or an intention-like entity (functional operand) performs in a certain manner, so, its detachment becomes possible.

4. *Modus obliquus* belongs to the most contentious inference rules because it usually proceeds from an absurd situation where the inference from a contradictory situation suggest to use the rule for an achievement of the logically consistent result. From a deceitful situation just the absurd inference should help to reach a firm conclusion. In fact, modus obliquus belongs to the so-called discursive informing where through a discourse (as communication, informing, reasoning) by all possible logical tricks, including absurd, contradiction, controversial informing, a valuable conclusion should be detached. In this sense, modus obliquus becomes a discursive filter delivering a useful result. In this sense, one of its possible schemes could be

$$\frac{\begin{array}{l}\alpha_{\text{absurd}}(\alpha) \subset \alpha; \\ (\alpha_{\text{absurd}}(\alpha) \models; \models \alpha_{\text{absurd}}(\alpha)) \Longrightarrow \beta)\end{array}}{(\alpha_{\text{absurd}}(\alpha) \not\models; \not\models \alpha_{\text{absurd}}(\alpha)) \subset \beta}$$

where $\alpha_{\text{absurd}}(\alpha)$ marks an absurd component of entity α hidden in α which phenomenalistically informs β, so that its informational phenomenalism can be identified in β as the observer.

The presented example of modus obliquus shows how other inference schemes would be possible and how, by further decomposition of the rule and its components, more complex and semantically concrete inferential schemes could be developed.

5. *Modus procedendi* is a mood by which a goal informational entity $\mathfrak{g}_{\text{goal}}(\alpha)$ will be detached from the informing entity α. An example of modus procedendi is

$$\frac{(\mathfrak{g}_{\text{goal}} \subset_{\text{goal}} \alpha); (\alpha \Longrightarrow_{\text{goal}} \mathfrak{g}_{\text{goal}})}{\alpha \models_{\text{goal}} (\mathfrak{g}_{\text{goal}} \models_{\text{goal}} \alpha)};$$

where implication operator $\Longrightarrow$ is particularized to $\Longrightarrow_{\text{goal}}$, causing a consequent goal-like informing (operators $\subset_{\text{goal}}$ and $\models_{\text{goal}}$). The conclusion of the inference rule can certainly be extended, bringing other components of the goal structure to the parallel interpretative surface, for instance,

$$\begin{array}{l}
\alpha \models_{\text{goal}} (\mathfrak{g}_{\text{goal}} \models_{\text{goal}} \alpha); \\
(\alpha \models_{\text{goal}} \mathfrak{g}_{\text{goal}}) \models_{\text{goal}} \alpha; \\
\mathfrak{g}_{\text{goal}} \models (\mathfrak{G}_{\text{goal}} \models \mathfrak{g}_{\text{goal}}); \\
(\mathfrak{g}_{\text{goal}} \models \mathfrak{G}_{\text{goal}}) \models \mathfrak{g}_{\text{goal}}; \\
\vdots
\end{array}$$

To this goal system within entity α other interpretative formulas can be added.

6. *Modus operandi* discovers the inside of an entity, its modus operandi. The inside structure of an entity α is its metaphysicalism in the form of informing $\mathfrak{I}_\alpha$, counterinforming $\mathfrak{C}_\alpha$ and informational embedding $\mathfrak{E}_\alpha$, together with counterinformational entity $\mathfrak{c}_\alpha$ and embedding entity $\mathfrak{e}_\alpha$. Thus, the consequent inference rules are nothing else than circularly structured modi ponens, for example,

$$\begin{array}{ll}
\dfrac{\alpha; (\alpha \Longrightarrow \mathfrak{I}_\alpha)}{\mathfrak{I}_\alpha}; & \dfrac{\mathfrak{I}_\alpha; (\mathfrak{I}_\alpha \Longrightarrow \mathfrak{C}_\alpha)}{\mathfrak{C}_\alpha}; \\
\dfrac{\mathfrak{C}_\alpha; (\mathfrak{C}_\alpha \Longrightarrow \mathfrak{c}_\alpha)}{\mathfrak{c}_\alpha}; & \dfrac{\mathfrak{c}_\alpha; (\mathfrak{c}_\alpha \Longrightarrow \mathfrak{E}_\alpha)}{\mathfrak{E}_\alpha}; \\
\dfrac{\mathfrak{E}_\alpha; (\mathfrak{E}_\alpha \Longrightarrow \mathfrak{e}_\alpha)}{\mathfrak{e}_\alpha}; & \dfrac{\mathfrak{e}_\alpha; (\mathfrak{e}_\alpha \Longrightarrow \alpha)}{\alpha}
\end{array}$$

An additional inference rule (modus operandi) could be

$$\frac{\alpha; (\alpha \Longrightarrow ((((\mathfrak{e}_\alpha \subset \mathfrak{E}_\alpha) \subset \mathfrak{c}_\alpha) \subset \mathfrak{C}_\alpha) \subset \mathfrak{I}_\alpha) \subset \alpha)}{\mathfrak{e}_\alpha; \mathfrak{E}_\alpha; \mathfrak{c}_\alpha; \mathfrak{C}_\alpha; \mathfrak{I}_\alpha}$$

7. *Modus vivendi* is certainly an informational case, for instance, how to infer in an (extremely) critical situation. Modus vivendi does not necessarily consider an extremely logical intention, so it can extend from an uncertain situation, e.g., modus possibilitatis, modus obliquus to modus ponens, as the most certain, approved and standard rule of inference.

Informational modus vivendi of an informational realm concerns inference in situations of life and surviving compromises happening under entity's environment, individual, populational and social circumstances [6].

Let the sensory information $\sigma_\alpha(\beta)$ of entity α observing entity β inform the α's metaphysicalistic structure, marked by $\alpha \models_{\text{metaphysicalistically}} \alpha$, where the metaphysicalistic structure of α is something represented by the circular formula

$$\alpha \models (\mathfrak{I}_\alpha \models (\mathfrak{C}_\alpha \models (\mathfrak{c}_\alpha \models (\mathfrak{E}_\alpha \models (\mathfrak{e}_\alpha \models \alpha)))))$$

and other formulas belonging to the α's metaphysicalistic gestalt [9] of length $\ell = 6$. Thus,

$$\beta \models (\sigma_\alpha(\beta) \models (\alpha \models_{\text{metaphysicalistically}} \alpha))$$

The last formula is nothing else than an informational decomposition of transition $\beta \models \alpha$ where the essential operator frames [9] in this formula are

$$\beta \boxed{\models (\sigma_\alpha(\beta) \models} (\alpha \models_{\text{metaphysicalistically}} \alpha) \boxed{)} \text{ and}$$
$$\beta \boxed{\models (\sigma_\alpha(\beta) \models (\alpha \models_{\text{metaphysicalistically}}} \alpha \boxed{))}$$

These cases of transition $\beta \models \alpha$ can be interpreted in the form of the so-called operator composition $\beta \models_\beta \circ \models_\alpha \alpha$, that is,

$$\beta \models_\beta \circ \models_{\sigma_\alpha(\beta)} (\alpha \models_{\text{metaphysicalistically}} \alpha) \quad \text{and}$$
$$\beta \models_\beta \circ \models_{\sigma_\alpha(\beta) \models_{\text{metaphysicalistically}} \alpha} \alpha$$

By observing of β by α, the so-called behavioral information $\beta_{\text{behavior}}(\alpha, \beta)$ produced by the α's metaphysicalistic structure, through its sensory information $\sigma_\alpha(\beta)$ and, dependent on environment β, has to be detached, where $\beta_{\text{behavior}}(\alpha, \beta)$ is on the way to assure the survival of α depending on environmental information β (in the sense of modus vivendi). Within this discourse, the following rule of modus vivendi arises:

$$\frac{\begin{pmatrix}(\sigma_\alpha(\beta) \subset (\beta \models \alpha)); \\ ((\sigma_\alpha(\beta) \models (\alpha \models_{\text{metaphysicalistically}} \alpha)) \Longrightarrow \\ \beta_{\text{behavior}}(\alpha, \beta))\end{pmatrix}}{\beta_{\text{behavior}}(\alpha, \beta)}$$

Senseful other examples of informational modus vivendi can come to the surface in more concrete informational situations and attitudes.

8. *Modus possibilitatis* roots in the philosophy of modal logic [1]. Instead of $\Diamond A$ which means *possibly A* (exactly, A is possibly true), we can introduce $\alpha \models_{\text{possibly}}$ which reads α informs possibly. A shortcut would be $\alpha\Diamond$. Thus, $\alpha\Diamond\beta$ is read *α possibly informs β* or also *β is possibly informed by α*. Sometimes, $\alpha \models_\Diamond \beta$ is an appropriate notation for a case of possible informational transition.

The question is, for instance, which are the possibilities of an informational entity α's arising. How can α possibly arise? The possibility of α's arising depends on the possible informing of α itself and on informing of an exterior entity, say β, which could possibly impact α informationally. Thus, the basic situation is a double transition $\alpha, \beta \models_\Diamond \alpha$ with the meaning

$$(\alpha, \beta \models_\Diamond \alpha) \rightleftharpoons \begin{pmatrix}\alpha \models_\Diamond \alpha; \\ \beta \models_\Diamond \alpha\end{pmatrix}$$

Let $\pi_\Diamond$ mark a structure of informational possibilities, where $\pi_1, \pi_2, \cdots, \pi_n \subset \pi_\Diamond$. These components can be detached transitionally by modus possibilitatis in the form

$$\frac{(\alpha, \beta, \pi_\Diamond); ((\alpha, \beta \models_\Diamond \alpha) \Longrightarrow \pi_\Diamond)}{\pi_1, \pi_2, \cdots, \pi_n \subset \pi_\Diamond}$$

Transitionally means, that the conclusion has to be unfolded in the sense of the operator $\subset$ definition [8]. In general, by informational modus possibilitatis it is possible to infer about possibilities of informing of interrelated entities.

9. *Modus necessitatis*, also, roots in the philosophy of modal logic [1]. Instead of $\Box A$ which means *necessarily A* (exactly, A is necessarily true), we can introduce $\alpha \models_{\text{necessarily}}$ which reads α informs necessarily. A shortcut would be $\alpha\Box$. Thus, $\alpha\Box\beta$ is read *α necessarily informs β* or also *β is necessarily informed by α*. Sometimes, $\alpha \models_\Box \beta$ is an appropriate notation for a case of possible informational transition. In modal logic, the interplay of possibility and necessity becomes the essential point of formula derivation. Thus, in modal logic, a schema embodying the idea of that *what is possible is just what is not-necessarily-not* is given by the formula

$$\Diamond A \leftrightarrow \neg\Box\neg A \quad (\text{or } \overline{\Box\overline{A}})$$

An informational transcription of this modal formula would be

$$(\alpha \models_\Diamond) \Longleftrightarrow (\alpha \not\models) \not\models_\Box$$

with the meaning 'α informs possibly' is informationally equivalent to 'α does not inform, does not inform necessarily'.

The question is again, which are the necessities of an informational entity α's arising. How can α necessarily arise? The necessity of α's arising depends on the necessary informing of α itself and on informing of an exterior entity, say β, which could necessarily impact α informationally. Thus, the basic situation is a double transition $\alpha, \beta \models_\Box \alpha$ with the meaning

$$(\alpha, \beta \models_\Box \alpha) \rightleftharpoons \begin{pmatrix}\alpha \models_\Box \alpha; \\ \beta \models_\Box \alpha\end{pmatrix}$$

Let $\nu_\Box$ mark a structure of informational necessities, where $\nu_1, \nu_2, \cdots, \nu_n \subset \nu_\Box$. These components can be detached transitionally by modus necessitatis in the form

$$\frac{(\alpha, \beta, \nu_\Box); ((\alpha, \beta \models_\Box \alpha) \Longrightarrow \nu_\Box)}{\nu_1, \nu_2, \cdots, \nu_n \subset \nu_\Box}$$

Transitionally means, that the conclusion has to be unfolded in the sense of the operator $\subset$ definition [8]. In general, by informational modus necessitatis it is possible to infer about necessities of informing of interrelated entities.

5 General Schemes of Informational Axioms

Up to now we have discussed the nature of informational axioms rooting to some extent in the tradition of mathematical logic. This tradition has its foundation in the groups I–V of propositional axioms dealt with in Subsection 3.2 and the two predicate axioms in the group VI in Subsection 3.5. The critical question is if these axioms can be generalized in a more radical way as they have been changed in an informational manner through a slight universalization of propositional operators when replacing implication $\rightarrow$, conjunction $\wedge$, disjunction $\vee$, equivalence $\equiv$ and negation $^{-}$ by informational implication $\Longrightarrow$, parallel operator ';' (semicolon), operand separator ',' (comma), informational equivalence $\Longleftrightarrow$ and operator of a (certain) noninforming $\not\models$, respectively. In case of predicates, the predicate expressions $\forall\alpha$ and $\exists\alpha$ representing the application of the universal and existential quantifier, must be replaced by informational operators of the form $\models_{\forall\alpha}$ and $\models_{\exists\alpha}$, respectively, where the concerned entity α had appeared in the operator subscript. At this occasion, the question of the informational nature of logical (predicate) quantifiers had come to the surface as a reflection which, possibly, may demand a deepened (more critical) informational treatise.

5.1 Informing versus Informational Implication

Implication belongs to the most significant traditional logical concepts. In this respect, the role of implication is revealed or hidden practically in any logical axiom and inference rule. Implication is the basis of the traditional logical calculus.

In informational calculus, the role of implication can be generalized. By generalization, the operator of informational implication, $\Longrightarrow$, is replaced by the most general informational operator (joker), $\models$. In this case, the concluding from one true situation to the other can be replaced by informing from one informing situation to the other. In the uttermost situation, this kind of the concluding informing can be comprehended as a general type of inference in which the traditional if-thenism (implicativeness, causality, consequentiality) is replaced by the most general informational impacting in the sense of informing (informational externalism) and observing (informational internalism), that is, in a form of informational transition occurring (happening) between the informing entity (emitter) on one side and the observing entity (receiver) on the other side. The informational transition between the informer and observer can be discussed in the form of an informational operator decomposition where one part of this decomposition belongs to the informer and the other part of decomposition to the observer. This kind of transitional decomposition between the informer and observer can be understood as the operator composition between two operands.

Let us discuss the axiom groups I–V in Subsection 3.2 and the group VI in Subsection 3.5 under the most general informational circumstances. There is:

I. General axioms of informing
  1) $\alpha \models (\beta \models \alpha)$;
  2) $(\alpha \models (\alpha \models \beta)) \models (\alpha \models \beta)$;
  3) $(\alpha \models \beta) \models ((\beta \models \gamma) \models (\alpha \models \gamma))$

II. General axioms of parallelism
  1) $(\alpha; \beta) \models \alpha$;
  2) $(\alpha; \beta) \models \beta$;
  3) $(\alpha \models \beta) \models$
     $((\alpha \models \gamma) \models (\alpha \models (\beta; \gamma))$

III. General axioms of alternativeness
  1) $\alpha \models \alpha, \beta$;
  2) $\beta \models \alpha, \beta$;
  3) $(\alpha \models \beta) \models$
     $((\beta \models \gamma) \models (\alpha, \beta \models \gamma))$

IV. General axioms of equivalence
  1) $(\alpha \models_{\equiv} \beta) \models (\alpha \models \beta)$;
  2) $(\alpha \models_{\equiv} \beta) \models (\beta \models \alpha)$;
  3) $(\alpha \models \beta) \models ((\beta \models \alpha) \models (\alpha \models_{\equiv} \beta))$

V. General axioms of noninforming
  1) $(\alpha \models \beta) \models$
     $((\beta \not\models; \not\models \beta) \models (\alpha \not\models; \not\models \alpha))$;
  2) $\alpha \models ((\alpha \not\models; \not\models \alpha) \not\models; \not\models (\alpha \not\models; \not\models \alpha))$
  3) $((\alpha \not\models; \not\models \alpha) \not\models; \not\models (\alpha \not\models; \not\models \alpha)) \models \alpha$

VI. General functional axioms
  1) $(\alpha \models_{\forall\alpha} \varphi(\alpha)) \models \varphi(\beta)$;
  2) $\varphi(\beta) \models (\alpha \models_{\exists\alpha} \varphi(\alpha))$

These axioms can be commented in several ways. Axiom (I.1) is evidently a satisfactory one. It says that entity α has, in general, its informer β and that this fact is informed by α itself. If one takes data δ, (I.1) becomes

$$\delta \models (\beta \not\models \delta)$$

Data informs as a fact and cannot be informed. Operator $\not\models$ is nothing else than a particular informing, that is, non-informing. Comments to other axioms can become not only very challenging but also provocative, controversial and novel for the usual understanding.

5.2 Generalized Initial Informational Axioms

The very initial informational axioms will always remain within the scope of the informationally possible (the informationally arising). Which could be the most universal (initial, original) informational axioms? To answer this question one must certainly consider the specific informational phenomenalism with its four basic phenomenal forms which are the externalism, internalism, metaphysicalism and phenomenalism. These axioms have not the usual implicative form but the universally informing one.

Some of the most general informational axioms are those concerning an informational entity's basic phenomenalism. For instance, axiom

$$\alpha \models ((\alpha \models) \models \alpha)$$

can be read in the following way: "Operand α informs in such a way that it is informed how does it inform." Axiomatic formula

$$\alpha \models ((\models \alpha) \models \alpha)$$

can be interpreted as: "Operand α informs in such a way that it is informed how it is being informed." Axiomatic formula

$$\alpha \models ((\alpha \models \alpha) \models \alpha)$$

can be read as: "Operand α informs in such a way that it is informed how does it inform and how is it informed within itself." At last, axiomatic formula

$$\alpha \models ((\alpha \models; \models \alpha) \models \alpha)$$

can mean: "Operand α informs in such a way that it is informed how does it inform and how is it informed as such."

The listed four general initial axioms, proceeding from (I.1), are structured in a strictly observer situation when the main operator $\models$ directly follows the leftmost operand α. The other possibility would be a strictly informer situation when the main operators precedes the rightmost operand. For this case, the listed four general axioms become

$$\begin{gathered}(\alpha \models (\alpha \models)) \models \alpha; \\ (\alpha \models (\models \alpha)) \models \alpha; \\ (\alpha \models (\alpha \models \alpha)) \models \alpha; \\ (\alpha \models (\alpha \models; \models \alpha)) \models \alpha\end{gathered}$$

The point of informer philosophy is that, within the informational transition between α as informer and α as observer (an α-circular case), the entire additional information [$\alpha \models$, $\models \alpha$, $\alpha \models \alpha$, and $(\alpha \models; \models \alpha)$], is on the informer side.

5.3 Generalized Inferential Informational Axioms

In a similar way as the basic axioms, it is possible to generalize the well-known inferential axioms, e.g., modus ponens, modus tollens, modus rectus, etc. A list of such generalizations of informational inference axioms would be the following:

GMP. $(\alpha; \alpha \models \beta) \models \beta;$

GMT. $(\alpha \models \beta; (\beta \not\models; \not\models \beta)) \models$
$(\alpha \not\models; \not\models \alpha);$

GMR. $(\alpha; (\alpha \models_{\text{intend}} \beta) \models \iota_{\text{intention}}(\alpha))) \models$
$((\iota_{\text{intention}}(\alpha) \models; \models \iota_{\text{intention}}(\alpha)) \subset \beta);$

GMO. $\begin{pmatrix} \alpha_{\text{absurd}}(\alpha) \subset \alpha; \\ (\alpha_{\text{absurd}}(\alpha) \models; \models \alpha_{\text{absurd}}(\alpha)) \models \beta \end{pmatrix} \models$
$(\alpha_{\text{absurd}}(\alpha) \not\models; \not\models \alpha_{\text{absurd}}(\alpha)) \subset \beta;$

GMPr. $(\mathfrak{g}_{\text{goal}} \subset \alpha; \alpha \models \mathfrak{g}_{\text{goal}}) \models$
$(\alpha \models (\mathfrak{g}_{\text{goal}} \models \alpha));$

GMOp. $\left\{\begin{array}{l} (\alpha; \alpha \models \mathfrak{I}_\alpha) \models \mathfrak{I}_\alpha; \\ (\mathfrak{I}_\alpha; \mathfrak{I}_\alpha \models \mathfrak{C}_\alpha) \models \mathfrak{C}_\alpha; \\ (\mathfrak{C}_\alpha; \mathfrak{C}_\alpha \models \mathfrak{c}_\alpha) \models \mathfrak{c}_\alpha; \\ (\mathfrak{c}_\alpha; \mathfrak{c}_\alpha \models \mathfrak{E}_\alpha) \models \mathfrak{E}_\alpha; \\ (\mathfrak{E}_\alpha; \mathfrak{E}_\alpha \models \mathfrak{e}_\alpha) \models \mathfrak{e}_\alpha; \\ (\mathfrak{e}_\alpha; \mathfrak{e}_\alpha \models \alpha) \models \alpha \end{array}\right.$

GMV. $\begin{pmatrix} (\sigma_\alpha(\beta) \subset (\beta \models \alpha)); \\ ((\sigma_\alpha(\beta) \models (\alpha \models \alpha)) \models \\ \quad \beta_{\text{behavior}}(\alpha, \beta)) \end{pmatrix} \models$
$\beta_{\text{behavior}}(\alpha, \beta);$

GMPo. $((\alpha, \beta, \pi_\Diamond); ((\alpha, \beta \models_\Diamond \alpha) \models \pi_\Diamond)) \models$
$(\pi_1, \pi_2, \cdots, \pi_n \subset \pi_\Diamond);$

GMN. $((\alpha, \beta, \nu_\Box); ((\alpha, \beta \models_\Box \alpha) \models \nu_\Box)) \models$
$(\nu_1, \nu_2, \cdots, \nu_n \subset \nu_\Box)$

This list of inference-axiom formulas includes the modi 1–9, discussed merely in a superficial (conceptually possible) way in Subsection 4.3. The main operators between the premise and consequence of each rule can be particularized by the use of the so-called detachment operator which, from case to case, can be specifically decomposed, according to the transition between the premise system (operand) and the consequence system (operand). Any inference transition from the premise to the conclusion can be understood to be a real informational discourse which underlies the concept of regular informational arising (phenomenalism) in the scope of principles of circularity, spontaneity, serialism, parallelism, etc.

6 Axioms and Theorems of Decomposition within Informational Calculus

The philosophy of informational decomposition of informational entities brings to the surface a substantial matter of deconstruction of entities which is not only semantic but causes semantic changes also through syntactic possibilities of formula-representing entities. Informational decomposition is a term which replaces the traditional concepts of induction, deduction, abduction and the role of inference. Decomposition means that we are usually concerned with general entity concepts which have to be decomposed into greater and greater detail. Decomposition also has the meaning of interpretation when formulas are added to the already informationally determined formula systems. Decomposition follows the informational possibilities of analysis with synthesis of formulas, by modeling, describing and realizing actively a certain informational system.

6.1 Axioms of the Informational Particularism and Universalism

Particularizing and universalizing the existing informational formulas means to decompose (determine, concretize, supplement) them into greater (more adequate, realistic) details. For example, formula $\alpha \models \beta$ can be decomposed in several ways, that is, in respect to operand α, operator $\models$, and operand β. Such decompositions can be simple and also very complex. We have learned about two possibilities of operator decomposition using operator frames in Subsection 4.3, discussing the informational modus vivendi.

Particularization and universalization as a decompositional procedure performs as a local transformation in the sense of analyzing and synthesizing informational operands and operators by bringing to the surface essential and possible informational details and building them into existing formula structures. Both operands and operators can be universalized (generalized) and particularized (individualized), directly or interpretively in parallel, by operand decomposition and operator composition, that is in several possible ways.

6.2 Axioms of Informational Serialism

Informational serialism springs from the question how could an informational entity be decomposed in a serial form, to determine the serial structure (components) concerning the entity. The basic axiom on this way is

$$\frac{\alpha}{\alpha \models \beta}; \ \beta \subset \alpha$$

which is recursive regarding β. Such a transitive recursiveness transits into a system of the form

$$\frac{\alpha}{\alpha \models \alpha_1}; \ \frac{\alpha_1}{\alpha_1 \models \alpha_2}; \ \cdots; \ \frac{\alpha_{n-1}}{\alpha_{n-1} \models \alpha_n};$$
$$\alpha_1, \alpha_2, \cdots, \alpha_{n-1}, \alpha_n \subset \alpha$$

By the strict substitution from the beginning to the end (from the left to the right), one of the resulting axioms of informational serialism becomes

$$\frac{\alpha}{\alpha \models (\alpha_1 \models (\alpha_2 \cdots \models (\alpha_{n-1} \models \alpha_n) \cdots))};$$
$$\alpha_1, \alpha_2, \cdots, \alpha_{n-1}, \alpha_n \subset \alpha$$

In fact, instead of a single decomposition formula, there can exist various decomposition formulas of length $\ell = n$, where operands α_i $(i = 1, 2, \cdots, n)$ and to them belonging operators preserve their places and only the parentheses pairs are displaced in all possible manners.

On the other side, an informing entity α can form a serial composition with exterior components α_i, where $\alpha_i \not\subset \alpha$.

6.3 Axioms of Informational Parallelism

Informational parallelism emerges from the question how could an informational entity be decomposed in a parallel form, to determine the parallel structure (components) concerning the entity. The basic axiom in this direction is

$$\frac{\alpha}{\alpha; \alpha_i}; \frac{\alpha_i}{\alpha_i; \alpha_i \models \alpha_j}; \alpha_i, \alpha_j \subset \alpha$$

which is recursive regarding α_i and α_j. Such a transitive recursiveness transits, among other possibilities, into a parallel inference system of the form

$$\frac{\alpha}{\alpha; \alpha \models \alpha_1}; \frac{\alpha_1}{\alpha_1; \alpha_1 \models \alpha_2}; \cdots; \frac{\alpha_{n-1}}{\alpha_{n-1}, \alpha_{n-1} \models \alpha_n};$$
$$\alpha_1, \alpha_2, \cdots, \alpha_{n-1}, \alpha_n \subset \alpha$$

By such and such application of the listed inferential rules, one of the formulas for the parallel decomposition becomes

$$\frac{\alpha}{\begin{pmatrix} \alpha; \alpha_i; \alpha_i \models \alpha_j; \\ i, j = 1, 2, \cdots, n \end{pmatrix}}$$

In fact, instead of a single decomposition formula, there can exist various formulas for parallel decomposition of different lengths regarding the participating components α_i and transitions $\alpha_i \models \alpha_j$, which can be mixed in all possible manners. If components α_i are interior to α, there is $\alpha_i \subset \alpha$. This condition [8] brings various possibilities for the decomposition-rule construction.

On the other side, an informing entity α can form a parallel composition with exterior components α_i, where $\alpha_i \not\subset \alpha$.

6.4 Axioms of Informational Circularity

The axiomatic view of informational circularity can stay within the framework of informational serialism. The only difference is that the first and the last operand in a serial formula are the same.

The most important case of informational circularity seems to be the metaphysicalistic one in which the participating components belong to the main circular entity. This does not mean that circularity cannot perform in an exterior way when information informed by an information source returns to it, adequately transformed by the exterior informational entities. Thus, evidently,

$$\frac{\alpha}{\alpha \models (\alpha_1 \models (\alpha_2 \cdots \models (\alpha_{n-1} \models (\alpha_n \models \alpha)) \cdots))};$$
$$\alpha_1, \alpha_2, \cdots, \alpha_{n-1}, \alpha_n \subset \alpha$$

could be one of other possible rules for serial circular decomposition. In a parallel case, the canonical circular decomposition

$$\frac{\alpha}{\begin{pmatrix} \alpha \models \alpha_1; \alpha_i \models \alpha_{i+1}; \alpha_n \models \alpha \\ i = 1, 2, \cdots, n \end{pmatrix}}$$

could be one of the parallel characteristic cases.

A particular and the most interesting case arises with the question of the circular causal informing. In this case, several other rules can be constructed which even the participation of the components (entities, operands, formulas) within a causal loop. Causal problems of informing are studied fundamentally and will be treated on some other places[2].

6.5 Decomposition and Composition Theorems of Informational Calculus

According to deduction theorems of propositional and predicate calculus (Subsection 3.4 and Subsection 3.7, respectively), it is possible to list some fundamental axioms and theorems of the informational calculus (see also Subsubsection 4.2.7). But, decomposition theorems of informational calculus can also become inferential, and new modi informationis can be derived according to informational axioms also beyond the discussed inferential rules in Subsection 4.3. In distinction from inference rules in traditional logic, all of which are axiomatically determined, the general informational theory enables a decompositional rise of inference rules, which become an essential source of the possibilities of informational arising of entities (informational operands and operators, the last ones by a procedure of decomposition).

Decomposition is a much broader term than are, for instance, deduction, induction and abduction for it not only unites them but adds new possible principles of the spontaneously arising in the sense of modi informationis.

Decomposition and Composition Theorem. *If formula β is informationally derivable from formulas α_1, α_2, $\cdots$, α_n, then*

[2]Causal informing of serially and in parallel decomposed informational entities is studied by the author in the paper entitled "Causality of the Informational" which is in preparation and will appear soon.

$$\alpha_1 \models (\alpha_2 \models (\cdots(\alpha_n \models \beta)\cdots))$$

is an informationally regular serial decomposition or composition formula and

$$\begin{aligned}&\alpha_1 \models \alpha_2;\\ &\alpha_2 \models \alpha_3;\\ &\cdots\\ &\alpha_{n-1} \models \alpha_n;\\ &\alpha_n \models \beta\end{aligned}$$

is an informationally regular parallel decomposition or composition formula system. □

By induction it is possible to show the following: if

$$\alpha_1, \alpha_2, \cdots, \alpha_{n-1}, \alpha_n \twoheadrightarrow \beta$$

then

$$\alpha_1, \alpha_2, \cdots, \alpha_{n-1} \twoheadrightarrow (\alpha_n \models \beta)$$

where symbol $\twoheadrightarrow$ is used for marking a derived formula within the informational calculus.

A Decomposition Theorem Concerning Informational Gestalt of a Serial Formula. *Let $\alpha \models (\alpha_1 \models (\alpha_2 \models (\cdots(\alpha_{n-1} \models \alpha_n)\cdots)))$ be a serial formula, where $\alpha_1, \alpha_2, \cdots, \alpha_n \subset \alpha$. It is to stress that the serial formula of length $\ell = n$ can also have arbitrarily distributed parentheses pairs and we mark it by $\sigma^{\rightarrow}(\alpha, \alpha_1, \alpha_2, \cdots, \alpha_n)$. Let mark by $\Gamma^n_{\rightarrow}(\alpha)$ the so-called informational gestalt assigned to a serial formula of length[3] n, informing from the left to the right, and let this gestalt be a parallel informational system of all possible formulas of length n. Then,*

$$\frac{\sigma^{\rightarrow}(\alpha, \alpha_1, \alpha_2, \cdots, \alpha_n)}{\Gamma^n_{\rightarrow}(\alpha)}$$

where system $\Gamma^n_{\rightarrow}(\alpha)$ includes

$$\frac{1}{n+1}\binom{2n}{n}$$

formulas of length n. □

This theorem will be proved on some other place[4].

7 A Preliminary Catalogue of Informational Rules of Decomposition

Rules which can be used in a decomposition (composition) process can be listed by a short catalogue. From these rules more complex and particularized rules can be constructed being adapted to a concrete (semantic, interpretative, arising) situation (possible deconstruction). The presented rule catalogue has to be considered as a preliminary one.

7.1 Basic Phenomenalistic Rules

An informational entity α can be decomposed in four basic forms, which are externalistic, internalistic, metaphysicalistic and phenomenalistic, respectively, that is,

$$\frac{\alpha}{\alpha \models}; \quad \frac{\alpha}{\models \alpha}; \quad \frac{\alpha}{\alpha \models \alpha}; \quad \frac{\alpha}{\alpha \models; \models \alpha}$$

This initial system of transformation (decomposition) rules offers the necessary alternatives by which the informational scope (interpretation) of an informational entity is extended.

An essential comment to the basic phenomenalistic rules concerns operand α which represents any informational entity in the following sense: if α appears in a rule several times, then each appearance could be comprehended as another entity, say, β, γ, etc. Such cases can become informationally legal in several decomposition processes of α's decomposition. Thus, also,

$$\frac{\alpha}{\beta \models}; \quad \frac{\alpha}{\models \beta}; \quad \frac{\alpha}{\alpha \models \beta}; \quad \frac{\alpha}{\beta \models \alpha}; \quad \frac{\alpha}{\beta \models \gamma};$$
$$\frac{\alpha}{\alpha \models; \models \beta}; \quad \frac{\alpha}{\beta \models; \models \beta}; \quad \frac{\alpha}{\beta \models; \models \gamma}$$

etc., derived from basic phenomenalistic rules, are legal and senseful. Another problem is the so-called mutual independence of axioms which may not concern the catalogued informational rules at all.

[3]The length ℓ of an informational formula is always determined by the number of the occurring binary informational operators of the type $\models$. In the debated case, the number of occurring operands in the serial formula is $\ell + 1$. Thus, the length of the formula is n, where there are n binary operators and $n + 1$ operands.

[4]Gestalts as informational entities are examined exhaustively by the author within the study entitled "Informational Frames and Gestalts" which is in preparation and will be issued soon.

7.2 A Formal Extension of the Basic Phenomenalistic Rules

The basic transformation rules in the preceding subsection can be systematically permuted regarding to the externalistic, internalistic, metaphysicalistic and phenomenalistic case, respectively. There is

$$\frac{\alpha \models}{\alpha}; \frac{\alpha \models}{\models \alpha}; \frac{\alpha \models}{\alpha \models \alpha}; \frac{\alpha \models}{\alpha \models; \models \alpha};$$

$$\frac{\models \alpha}{\alpha}; \frac{\models \alpha}{\alpha \models}; \frac{\models \alpha}{\alpha \models \alpha}; \frac{\models \alpha}{\alpha \models; \models \alpha};$$

$$\frac{\alpha \models \alpha}{\alpha}; \frac{\alpha \models \alpha}{\alpha \models}; \frac{\alpha \models \alpha}{\models \alpha}; \frac{\alpha \models \alpha}{\alpha \models; \models \alpha};$$

$$\frac{\alpha \models; \models \alpha}{\alpha}; \frac{\alpha \models; \models \alpha}{\alpha \models}; \frac{\alpha \models; \models \alpha}{\models \alpha}; \frac{\alpha \models; \models \alpha}{\alpha \models \alpha}$$

By these rules, premises can be replaced by the corresponding conclusions of the rules.

As one may observe, the rules

$$\frac{\alpha \models}{\alpha}; \frac{\models \alpha}{\alpha}; \frac{\alpha \models \alpha}{\alpha}; \frac{\alpha \models; \models \alpha}{\alpha}$$

act in an reductionistic way (lowering the degree of a formula complexity).

7.3 Rules of Informational Parallelism

Rules of parallel decomposition of an informational entity root in the axioms of conjunction as particular cases of informational parallelism. It is to stress that the listed rules are in no way informationally independent because the aim of the rule catalogue is to show the reader various possibilities of informational decomposition.

Parallel decomposition covers the view of the possibilities of parallel interpretation of an informational case. The rules guarantee an introduction of parallel cases which explain the respective case into more and more details and in additional manners. In this way, the informational system representing an entity grows and becomes more and more complex, that is, informationally interweaved by its parallel structure.

7.3.1 Rules of Interior Parallel Decomposition

Although the basic rule of an interior parallel decomposition of an entity α is only one, that is,

$$\frac{\alpha}{\alpha; \alpha}$$

this rule can be "multiplied" by considering the rules from Subsection 7.2.

7.3.2 Rules of Exterior Parallel Decomposition

Which formulas can be understood as a parallel performing of their components? An evident informational rule resulting from the basic axiomatic philosophy is, for instance,

$$\frac{\alpha \models \beta}{\begin{pmatrix} \alpha; \\ \beta \end{pmatrix}}$$

This mean that the rule of interior parallel decomposition of entity α, which is $\frac{\alpha}{\alpha; \alpha}$, originates from the initial metaphysicalistic nature of the entity, that is,

$$\frac{\alpha \models \alpha}{\begin{pmatrix} \alpha; \\ \alpha \end{pmatrix}}$$

Thus, rule $\frac{\alpha}{\alpha; \alpha}$ is a consequence of implication

$$\left(\frac{\alpha}{\alpha \models \alpha}; \frac{\alpha \models \alpha}{\alpha; \alpha}\right) \Longrightarrow \frac{\alpha}{\alpha; \alpha}$$

7.4 Rules of Serial Decomposition

If decomposition of an entity α means an interior deconstruction (interpretation, analysis, synthesis) of α's informational structure, the appearing entities in a decomposition formula are informational components of entity α. In the rules of the so-called canonical[5] serial decomposition of α,

[5]As mentioned in a footnote before, the canonical and non-canonical decomposition of informational entities (operands) will be discussed on some other place, that is, in *Informational Frames and Gestalts* of the author, where concepts of canonical and non-canonical formulas will be presented in detail.

$$\frac{\alpha}{\alpha \models (\alpha_1 \models (\alpha_2 \models (\cdots(\alpha_{n-1} \models \alpha_n)\cdots)))};$$
$$\frac{\alpha}{(\alpha \models \alpha_1) \models (\alpha_2 \models (\cdots(\alpha_{n-1} \models \alpha_n)\cdots))};$$
$$\vdots$$
$$\frac{\alpha}{(\cdots((\alpha \models \alpha_1) \models \alpha_2)\cdots) \models (\alpha_{n-1} \models \alpha_n)};$$
$$\frac{\alpha}{((\cdots((\alpha \models \alpha_1) \models \alpha_2)\cdots) \models \alpha_{n-1}) \models \alpha_n}$$

where

$$\alpha_1, \alpha_2, \cdots, \alpha_n \subset \alpha$$

This means: as soon as an entity α_i, $i = 1, 2, \cdots, n$ is introduced (comes to the surface, arises, appears) in the serial structure of α's decomposition, it is internalized through the transition formula $\alpha_i \subset \alpha$ [8]. Other forms of serial decomposition are the so-called non-canonical formulas of length n, for instance formula

$$\frac{\alpha}{(\alpha \models (\alpha_1 \models \alpha_2)) \models (\alpha_3 \models (\cdots(\alpha_{n-1} \models \alpha_n)\cdots))}$$

in which the parenthesis pairs are non-canonically distributed.

7.5 Rules of Circular Decomposition

Rules of circular decomposition consider serial, metaphysicalistic, causal, and parallel decomposition.

7.5.1 Rules of Circular Serial Decomposition

A primitive, non-trivial rule of circular serial decomposition of entity α concerns α's informing $\mathfrak{I}_\alpha$, that is,

$$\frac{\alpha}{\alpha \models (\mathfrak{I}_\alpha \models \alpha)} \quad \text{and} \quad \frac{\alpha}{(\alpha \models \mathfrak{I}_\alpha) \models \alpha}$$

Rules for a general circular serial decomposition are characterized by conclusion formulas in which the first and the last entity are marked equally. Thus, the $n+1$ canonically-serial circular formulas are

$$\frac{\alpha}{\alpha \models (\alpha_1 \models (\alpha_2 \models (\cdots(\alpha_n \models \alpha)\cdots)))};$$
$$\frac{\alpha}{(\alpha \models \alpha_1) \models (\alpha_2 \models (\cdots(\alpha_n \models \alpha)\cdots))};$$
$$\vdots$$
$$\frac{\alpha}{(\cdots((\alpha \models \alpha_1) \models \alpha_2)\cdots) \models (\alpha_n \models \alpha)};$$
$$\frac{\alpha}{((\cdots((\alpha \models \alpha_1) \models \alpha_2)\cdots) \models \alpha_n) \models \alpha}$$

where

$$\alpha_1, \alpha_2, \cdots, \alpha_n \subset \alpha$$

On the other side, there are still

$$\frac{1}{n+2}\binom{2n+2}{n+1} - (n+1)$$

non-canonical serial circular formulas of length $n+1$.

7.5.2 Rules of Serial Metaphysicalistic Decomposition

Metaphysicalistic decomposition considers standard interior components of an entity α which are the following: informing $\mathfrak{I}_\alpha$, counterinforming $\mathfrak{C}_\alpha$, counterinformational entity $\mathfrak{c}_\alpha$, informational embedding $\mathfrak{E}_\alpha$, and informational embedding entity $\mathfrak{e}_\alpha$. Six canonical forward-cycle (right-loop) metaphysicalistic rules are:

$$\frac{\alpha}{\alpha \models (\mathfrak{I}_\alpha \models (\mathfrak{C}_\alpha \models (\mathfrak{c}_\alpha \models (\mathfrak{E}_\alpha \models (\mathfrak{e}_\alpha \models \alpha)))))};$$
$$\frac{\alpha}{(\alpha \models \mathfrak{I}_\alpha) \models (\mathfrak{C}_\alpha \models (\mathfrak{c}_\alpha \models (\mathfrak{E}_\alpha \models (\mathfrak{e}_\alpha \models \alpha))))};$$
$$\frac{\alpha}{((\alpha \models \mathfrak{I}_\alpha) \models \mathfrak{C}_\alpha) \models (\mathfrak{c}_\alpha \models (\mathfrak{E}_\alpha \models (\mathfrak{e}_\alpha \models \alpha)))};$$
$$\frac{\alpha}{(((\alpha \models \mathfrak{I}_\alpha) \models \mathfrak{C}_\alpha) \models \mathfrak{c}_\alpha) \models (\mathfrak{E}_\alpha \models (\mathfrak{e}_\alpha \models \alpha))};$$
$$\frac{\alpha}{((((\alpha \models \mathfrak{I}_\alpha) \models \mathfrak{C}_\alpha) \models \mathfrak{c}_\alpha) \models \mathfrak{E}_\alpha) \models (\mathfrak{e}_\alpha \models \alpha)};$$
$$\frac{\alpha}{(((((\alpha \models \mathfrak{I}_\alpha) \models \mathfrak{C}_\alpha) \models \mathfrak{c}_\alpha) \models \mathfrak{E}_\alpha) \models \mathfrak{e}_\alpha) \models \alpha}$$

To these 6 rules 126 non-canonical forward-cycle metaphysicalistic rules can be added.

Six canonical backward-cycle (left-loop) metaphysicalistic rules are:

$$\frac{\alpha}{\alpha \models (\mathfrak{e}_\alpha \models (\mathfrak{E}_\alpha \models (\mathfrak{c}_\alpha \models (\mathfrak{C}_\alpha \models (\mathfrak{I}_\alpha \models \alpha)))))};$$

$$\frac{\alpha}{(\alpha \models \mathfrak{e}_\alpha) \models (\mathfrak{E}_\alpha \models (\mathfrak{c}_\alpha \models (\mathfrak{C}_\alpha \models (\mathfrak{I}_\alpha \models \alpha))))};$$

$$\frac{\alpha}{((\alpha \models \mathfrak{e}_\alpha) \models \mathfrak{E}_\alpha) \models (\mathfrak{c}_\alpha \models (\mathfrak{C}_\alpha \models (\mathfrak{I}_\alpha \models \alpha)))};$$

$$\frac{\alpha}{(((\alpha \models \mathfrak{e}_\alpha) \models \mathfrak{E}_\alpha) \models \mathfrak{c}_\alpha) \models (\mathfrak{C}_\alpha \models (\mathfrak{I}_\alpha \models \alpha))};$$

$$\frac{\alpha}{((((\alpha \models \mathfrak{e}_\alpha) \models \mathfrak{E}_\alpha) \models \mathfrak{c}_\alpha) \models \mathfrak{C}_\alpha) \models (\mathfrak{I}_\alpha \models \alpha)};$$

$$\frac{\alpha}{(((((\alpha \models \mathfrak{e}_\alpha) \models \mathfrak{E}_\alpha) \models \mathfrak{c}_\alpha) \models \mathfrak{C}_\alpha) \models \mathfrak{I}_\alpha) \models \alpha}$$

To these 6 rules 126 non-canonical backward-cycle metaphysicalistic rules can be added.

7.5.3 Rules of Circular Serial Causal Decomposition

Causal decomposition of an entity (operand) can be serial and serially circular. In the first case, the difference between the ordinary serial and serially causal is only in the nature of informational operators. In the causal case, they must perform in a causal manner, possessing the property of causality.

In a circular serial case of causal decomposition of an entity, the circularly involved entities (components of the decomposed entity) become equally entitled in respect to the cycle. This can cause the arising of components parallel loops and their gestalts. The consequence can be an enormous rise of the number of parallel cycles[6].

7.5.4 Rules of Circular Parallel Decomposition

An operand α can be circularly decomposed in a parallel way by the following rule:

$$\frac{\alpha}{\begin{pmatrix} \alpha \models \alpha_1; \\ \alpha_1 \models \alpha_2; \\ \vdots \\ \alpha_{n-1} \models \alpha_n; \\ \alpha_n \models \alpha \end{pmatrix}}$$

[6]The case of causal circularity is studied exhaustively in *Causality of the Informational*, prepared by the author.

Such a rule describes a circular graph as shown in Fig. 4.

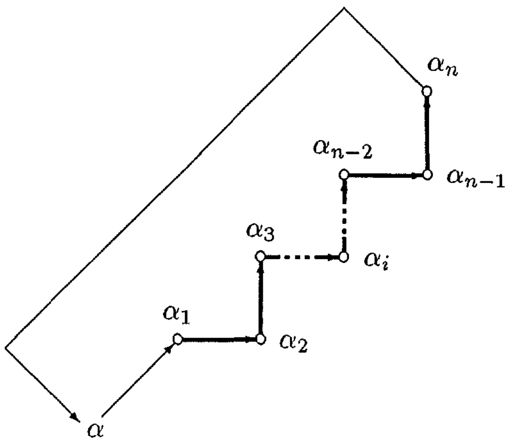


Figure 4: *A circular graph determined by the parallel system of transitions* ($\alpha \models \alpha_1; \alpha_1 \models \alpha_2; \cdots; \alpha_{n-1} \models \alpha_n; \alpha_n \models \alpha$). *The zig-zag course of connections (informational operators) between the informing components α_i of α points to a spontaneously discursive kind of informing.*

As we see, the graph in Fig. 4 can be the source of different circular interpretations, stretching from parallel circularity to serial causal circularity. It can simply mean that a parallel circularity implies all sorts of circular serial and circular causal informing with all possible circular gestalts. Within this comprehension, parallel circularity is the source of all possible rules of serial circularity within the informational calculus. For instance,

$$\frac{\begin{pmatrix} \alpha \models \alpha_1; \\ \alpha_1 \models \alpha_2; \\ \vdots \\ \alpha_{n-1} \models \alpha_n; \\ \alpha_n \models \alpha \end{pmatrix}}{\begin{pmatrix} \alpha \models (\alpha_1 \models (\alpha_2 \models (\cdots (\alpha_n \models \alpha) \cdots))); \\ (\alpha \models \alpha_1) \models (\alpha_2 \models (\cdots (\alpha_n \models \alpha) \cdots)); \\ \vdots \\ (\cdots ((\alpha \models \alpha_1) \models \alpha_2) \cdots) \models (\alpha_n \models \alpha); \\ ((\cdots ((\alpha \models \alpha_1) \models \alpha_2) \cdots) \models \alpha_n) \models \alpha \end{pmatrix}}$$

is an example of the rule by which from a circular parallel system of $n+1$ simple transitional formulas an adequate canonical circular serial system (gestalt) of $n+1$ formulas is obtained. Certainly, such a parallel system can cause also the arising of a non-canonical gestalt of circular serial formulas.

The most general rule for the replacement of a circular system of parallel transitions, marked by $\pi^{\parallel}(\alpha, \alpha_1, \cdots, \alpha_n)$ into a parallel system of circular formulas, that is, gestalts, becomes, for instance,

$$\frac{\pi^{\parallel}(\alpha, \alpha_1, \cdots, \alpha_n)}{\left(\begin{array}{c} \Gamma_{\circlearrowright}^{i+1}(\alpha); \Gamma_{\circlearrowleft}^{i+1}(\alpha); \\ \Gamma_{\circlearrowright}^{i+1}(\alpha_j); \Gamma_{\circlearrowleft}^{i+1}(\alpha_j); \\ i,j = 1,2,\cdots,n \end{array}\right)}$$

where $\Gamma_{\circlearrowright}^{i+1}(\alpha)$ is the gestalt of all forward-cycle (subscript $\circlearrowright$) formulas of length $i+1$ ($i = 1,2,\cdots,n$). Similarly, $\Gamma_{\circlearrowleft}^{i+1}(\alpha)$ is the gestalt of all backward-cycle (subscript $\circlearrowleft$) formulas of length $i+1$ ($i = 1,2,\cdots,n$). In the case of the so-called causal circularity, components $\alpha_1, \alpha_2, \cdots, \alpha_n \subset \alpha$ have the possibility to become informers and observers in informational forward and backward loops of lengths $i+1$, etc. So, $\Gamma_{\circlearrowright}^{i+1}(\alpha_j)$ and $\Gamma_{\circlearrowleft}^{i+1}(\alpha_j)$ are the corresponding gestalts.

7.6 Rules of Alternative Decomposition

Rules of alternative decomposition embrace all discussed rules in this section if the general informational operator (joker) $\models$ is replaced by the general alternative operator (alternative joker) $\mathrel{=\!\!|}$. By this replacement, formulas' informing(ness) operator becomes the informedness operator, that is, the operation of the Inform(s) is replaced by the operation of the Is/Are_Informed. The deduction of alternative informing originates from two different sources. The first one comes from regular informing, where the basic rule of informational transition could be

$$\frac{\alpha \models \beta}{\beta \mathrel{=\!\!|} \alpha}$$

This rule determines nothing more than that formula $\alpha \models \beta$ can be read alternatively as β is informed by α, that is, $\beta \mathrel{=\!\!|} \alpha$.

On the other hand, the alternative informing originates in the informational phenomenalism (Subsection 7.1), where

$$\frac{\alpha}{\mathrel{=\!\!|}\alpha}; \frac{\alpha}{\alpha\mathrel{=\!\!|}}; \frac{\alpha}{\alpha\mathrel{=\!\!|}\alpha}; \frac{\alpha}{\mathrel{=\!\!|}\alpha; \alpha\mathrel{=\!\!|}}$$

are the principled alternative rules.

The basic alternative transformation rules can be systematically inverted regarding to the externalistic, internalistic, metaphysicalistic and phenomenalistic case, respectively. There is

$$\frac{\mathrel{=\!\!|}\alpha}{\alpha}; \frac{\mathrel{=\!\!|}\alpha}{\alpha\mathrel{=\!\!|}}; \frac{\mathrel{=\!\!|}\alpha}{\alpha\mathrel{=\!\!|}\alpha}; \frac{\mathrel{=\!\!|}\alpha}{\mathrel{=\!\!|}\alpha; \alpha\mathrel{=\!\!|}};$$

$$\frac{\alpha\mathrel{=\!\!|}}{\alpha}; \frac{\alpha\mathrel{=\!\!|}}{\mathrel{=\!\!|}\alpha}; \frac{\alpha\mathrel{=\!\!|}}{\alpha\mathrel{=\!\!|}\alpha}; \frac{\alpha\mathrel{=\!\!|}}{\mathrel{=\!\!|}\alpha; \alpha\mathrel{=\!\!|}};$$

$$\frac{\alpha\mathrel{=\!\!|}\alpha}{\alpha}; \frac{\alpha\mathrel{=\!\!|}\alpha}{\mathrel{=\!\!|}\alpha}; \frac{\alpha\mathrel{=\!\!|}\alpha}{\alpha\mathrel{=\!\!|}}; \frac{\alpha\mathrel{=\!\!|}\alpha}{\mathrel{=\!\!|}\alpha; \alpha\mathrel{=\!\!|}};$$

$$\frac{\mathrel{=\!\!|}\alpha; \alpha\mathrel{=\!\!|}}{\alpha}; \frac{\mathrel{=\!\!|}\alpha; \alpha\mathrel{=\!\!|}}{\mathrel{=\!\!|}\alpha}; \frac{\mathrel{=\!\!|}\alpha; \alpha\mathrel{=\!\!|}}{\alpha\mathrel{=\!\!|}}; \frac{\mathrel{=\!\!|}\alpha; \alpha\mathrel{=\!\!|}}{\alpha\mathrel{=\!\!|}\alpha}$$

By these alternative rules, premises can be replaced by the corresponding conclusions of the rules.

As an example, the informational alternativeness enables the following implication:

$$(\alpha \models \beta) \Longrightarrow \left(\begin{array}{l} \beta \mathrel{=\!\!|} \alpha; \\ \alpha \mathrel{=\!\!|}_{\text{observe}} \beta; \\ \beta \models_{\text{observe}} \alpha; \\ \alpha \models_{\text{observingly}} \beta; \\ \beta \mathrel{=\!\!|}_{\text{impact}} \alpha \\ \vdots \end{array}\right)$$

The meanings of parallel formulas on the right side of $\Longrightarrow$ are not only informationally significant (logically instructive) but also various in a semantic manner and are the following:

- β is informed by α;
- α is observed by β;
- β observes α;
- α informs observingly β;
- β is informationally impacted by α;
- etc.

8 Conclusion

One of the aims of this paper is to show that the construction of an informational calculus is possible in a sufficiently (technically) rigorous way by a similar axiomatic approach known in the propositional and predicate calculus. The difference is certainly obvious: informational calculus covers much wider realm of informational possibilities which may lie beyond the propositional and predicate philosophy[7]. What could such assumption mean at all?

The author is aware that the presented construction of the informational calculus is preliminary and that still many theoretical concepts must be developed prior to a satisfactory final result. But it becomes also evident that a sufficiently developed theoretical approach builds up the fundament for something called the informational machine [11]. The main problem as seen by the author on this way is the so-called problem of informational arising which is the basic property of any informing entity. This basic property must be supported by the machine hardware and its operating system, that is automatically, when an entity begins to inform as an informational entity of the machine. And, as we have experienced during the reading of the paper, a strong support to the informational arising of any kind lies in the decomposition phenomenality of entities informing serially, in parallel, circularly, metaphysicalistically, and causally.

A great deal of the problems pointed out already in this paper will be unfolded in the context of the ongoing studies concerning:

1. informational frames and gestalts,
2. informational transition of the form $\alpha \models \beta$,
3. causality of the informational,
4. understanding and interpretation, and
5. informational memory.

By these studies, various cases of informational philosophy and formal theory will be presented and thematically rounded up with the aim to enable the concept and implementation of the informational machine.

[7]A good example of axiomatic informational possibilities is a derived axiom which considers the axioms of type I.3 in the mixed informing-implicative form, that is,

$$(\alpha \models \beta) \Longrightarrow \left((\beta \models \gamma) \Longrightarrow \begin{pmatrix} \alpha \models (\beta \models \gamma); \\ (\alpha \models \beta) \models \gamma) \end{pmatrix}\right)$$

This axiom is very useful and causes the so-called gestalt and causal development of formulas.

References

[1] B.F. Chellas: *Modal Logic.* An Introduction. Cambridge University Press, Cambridge, 1988.

[2] M. Heidegger: *Sein und Zeit.* Sechzehnte Auflage. Max Niemeyer Verlag, Tübingen, 1986.

[3] D. Hilbert und P. Bernays: *Grundlagen der Mathematik.* Erster Band. Die Grundlagen der mathematischen Wissenschaften in Einzeldarstellungen, Band XL. Verlag von Julius Springer, Berlin, 1934.

[4] G. Kwiatkowski (Ed.): *Schüler-Duden: Die Philosophie.* Dudenverlag, Mannheim, 1985.

[5] П.С. Новиков: *Элементы математической логики.* Государственное издательство физико-математической литературы (Физматгиз), Москва, 1959.

[6] A.P. Železnikar: *Informational Logic IV.* Informatica **13** (1989) No. 2, 6–23.

[7] A.P. Železnikar: *Verbal and Formal Deduction of Informational Axioms.* Cybernetica **37** (1994) No. 1, 5–32.

[8] A.P. Železnikar: *Informational Being-in.* Informatica **18** (1994) No. 2, 149–171.

[9] A.P. Železnikar: *Informational Being-of.* Informatica **18** (1994) No. 3, 277–298.

[10] A.P. Železnikar: *Principles of a Formal Axiomatic Structure of the Informational.* Informatica **18** (1995) No. 1, 133–158.

[11] A.P. Železnikar: *A Concept of Informational Machine.* Cybernetica **38** (1995) No. 1, 7–36.

Nonlinear Adaptive Prediction Algorithm and Its Parallel Implementation

Ryszard A. Wasniowski
New Mexico Highlands University, Las Vegas, NM
Phone: (505) 271-1288
E-mail: rwasniowski@acm.org

Keywords: parallel computing, gmdh algorithm

Edited by: Jerzy R. Nawrocki
Received: April 10, 1994 **Revised:** August 10, 1995 **Accepted:** September 7, 1995

This paper discusses and shows how computation intensive engineering problems can be modeled on parallel-like simulators and computed efficiently on massively parallel computers. Designed with tens of thousands of processing elements, these machines now offer substantially improved computation times, improved cost performance, and allow rapidly reach higher performance levels. Networks of workstations and software packages such as PVM is another attempt to provide a unified framework within which large parallel programs can be developed on a collection of heterogeneous machines, and make easy transition from sequential processing into parallel processing. The ease of developing parallel algorithms for systems identification using gmdh algorithm will be discussed.

1 Introduction

In recent years, parallel and distributed processing have been considered to be the most promising solution to the computing requirements of the future. Significant advances in parallel algorithms and architectures have shown the potential for applying parallel computation techniques for real-time systems. Relatively less attention has been given to software development environments or program construction techniques that are required to translate algorithms into operational programs. This aspect is becoming more important as parallel processing progresses from the research laboratories to carrying out parallel processing at various levels of practical applications. This study approaches nonlinear system identification by using parallel algorithms to form layers of a polynomial network in order to model an unknown nonlinear system.

System identification problems have been examined for decades. A substantial amount of work has been done in the identification of systems when outputs are known to vary as linear function of inputs. However, the identification of nonlinear systems remains a difficult problem. Many practical problems in signal processing, control, and machine learning can be cast as system identification problems, where one must gather data from a system whose structure is initially unknown, and build a model of the system's structure. The resulting model can be used for predictions and control of the initially unknown system. This problem compounds when no initial information is available about the system's structure. Once we assume the structure of the system, the problem reduces to a comparatively trivial job of identifying parameters. Identifying and modeling the structure of potentially nonlinear, black-box systems is a problem of particular significance to control engineering and machine learning research.

Our approach is motivated by group method of data handling (gmdh). The gmdh is a procedure that attempts to model an unknown system by iteratively connecting layers of nodes that compute polynomial functions. Unfortunately, this method suffers from huge computational overhead. This paper suggests improvement of the gmdh by using parallel computing. This technique was tested in the identification of continuous and discrete time forecasting systems.

2 The GMDH Algorithm

The design of a successful control system depends on the ability to predict its response during the given operating conditions. The deterministic approach to complex plant modeling and control often fails because the dynamics of the subsystems and their interconnections are not easily understood. The information available is not enough to construct differential equations for the system, so a different approach based on predictive polynomials is tried. The group method of data handling (gmdh) based on the principle of heuristic self-learning belongs to this group. This approach is intended for the solution of diverse interpolation problems in engineering, such as identification of the static and dynamic characteristics of plants, prediction of random processes and events, optimal control, pattern recognition, etc.

The evolution of the gmdh development is easy to follow. It is sufficient to look through the volumes of the journal Avtomatika beginning from 1968. Almost every issue contains papers related to gmdh problems. In a publication (Ivankhnienko 1969), an appendix is devoted to the new method. In the early stages 1968–1971, the gmdh algorithm was applied to the solution of pattern recognition, identification, and short-range prediction problems. The challenge of noisy data and incomplete information was met in the period 1972–1975. Two ways to handle noisy data in the gmdh algorithm were proposed: multicriteria choice of the model with a special form of the selection criteria, and the use of additional a priori information (combined method). In the period 1976–1979 it was shown that many multilayered gmdh algorithms have a multilayerness error, which is small but sometimes essential. This error is similar to the statistical error in control systems. It was shown also that multilayered algorithms, like statistical control systems, converge to an equilibrium point. Types of convergence were studied more thoroughly later. Many interesting results were found during 1980–82 period. It became clear that physical models (with noise) cannot be used for long range predictions. The stronger the noise, the simpler the models chosen by the gmdh algorithms. So physical models are obtained only for use in identification and short-range prediction. In the period 1982–1983 the various estimators were obtained to deal with multicollinearity of gmdh. Literature surveys show continuous interest in using gmdh for various scientific and engineering applications.

To make the ideas about gmdh algorithm clear, suppose that the input consists of N observable $x_1, x_2, ..., x_N$. It is assumed that the output y may be considered as the estimate of some property of the inputs. In general, y will be some nonlinear function as follows:

$$y = f(x_1, x_2, ..., x_N)$$

The problem is to find out the unknown structure y=f(.). In solving this problem, a general relationship between output and input variables is built in the form of a mathematical description, which is also called a reference function. Usually the description is considered as a discrete form of the Volterra functional series or Kolmogorov-Gabor polynomial (Ivankhnienko 1971):

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + ...,$$

where a is the vector of coefficients or weigths. Components of the input vector $x = (x_1, x_2, ..., x_N)$ could be independent variables or functional terms or finite difference terms. However, the determination of the correct terms and coefficients for this expansion is a difficult task. For a practical use the prediction algorithm should have simplicity, need a small amount of computation time, be well suited for real-time operation and applicable to a small amount of available data. Let us assume that $f(.)$ is represented by a polynomial of a certain order with respect to x_i. The application of Kolmogorov-Gabor polynomial will require a large amount of data with computation of high-dimensional matrices. Ivakhnienko suggested the gmdh procedure for finding out the appropriate polynomial for a given set of input/output data by constructing simple computational networks. In the first step gmdh creates the network first layers by assigning an input node to each independent variable x_i . Subsequent layers are created iteratively as follows:

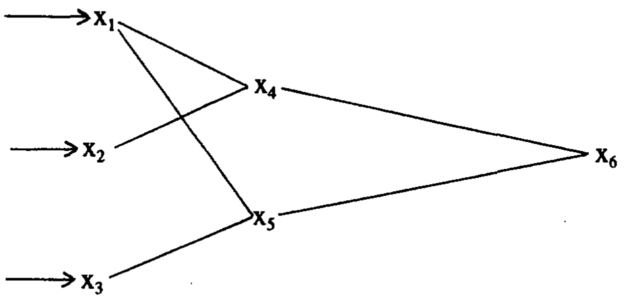


Figure 1: A typical gmdh network.

construct n^2 - n nodes for the new layer, where n is the number of nodes in the previous layer; connect the outputs of pairs of nodes from the previous layer to nodes in the new layer, each new node receives output from one of the $n^2 - n$ pairings of node outputs from the previous layer; find out the output of each node in the new layer by fitting an elementary function of the node's inputs to the y data; remove nodes with high-error from the new layer. (The original gmdh uses the mean-square error as a basis for node elimination.) One possible elementary function mentioned above is a quadratic of the nodes input signals:

$$y = A + Bx_i + Cx_j + Dx_i^2 + Ex_j^2 + Fx_ix_j$$

To find these type of polynomial (the coefficients A, B, C, D, E, and F) it is sufficient to have only six data points at our disposal. Repeated solution of the quadratic polynomials above enables us to construct the complete polynomial of any complexity using only these six points. This process continues until performance stops to improve with the addition of new layers. Figure 1 shows a typical gmdh network. Nodes are labeled with their output variable names. Each node computes its output as an elementary function of its input variables. (Note: In this simple example ($x_1 \& x_3$) produces no output.)

The gmdh can be realized by many algorithms that differ with respect to the construction and the complete description. The gmdh algorithm gives us in fact an alternative to deal with matrices of large dimension, and makes it possible to solve complex problems when the data sequence is relatively small. Versions of the gmdh have been applied with some success to engineering, economic and environmental problems. However, the technique suffers from two major drawbacks: the huge computational overhead associated with enumerating all the possible nodes in each layer, and the typical ills of local search techniques. The limitations implied by the selection of elementary functions are also a target of criticism. Tenorio and Lee (Tenorio & Lee 1990) have used multiple elementary functions and other modifications to improve the performance of the gmdh. However, the use of multiple elementary functions only increases the computational difficulties associated with gmdh. One can also criticize the comparison of nodes in intermediate layers with the optimal final output. Discarding nodes at an intermediate layer based on a final-output criteria could mislead the system to a suboptimal network. A more suitable approach could be to maintain several nodes in each layer based on an intermediate measure of their relative merit in that layer. The following section introduces a parallel polynomial network construction algorithm, and attempts to correct the computational overhead deficiency of the gmdh procedure.

3 GMDH Algorithm and Parallel Computing

Different systems identification methods based on gmdh with specific ranges of applicability have been invented and well documented. A common feature of most of such methods is the fact that they are realized as sequential computer programs that significantly increases the computer running time. In constructing the multilayer gmdh, all independent variables in one layer are combined, two at a time, to predict the local polynomial. This offers wide possibilities of parallelism for the information processing operations required in the construction of its models. These combinations can be considered as separate independent blocks performing the same task on different inputs. The gmdh algorithm could be made more efficient with an array of processing elements (PE), where each block is calculated by a different PE. Such an approach should significantly reduce the computer running time. In order to code the parallel gmdh algorithm the main algorithm should be divided into a number of small processes. Each

$$\begin{array}{lll}
 & x_1 & \\
\text{LAYER 1} & & \Longrightarrow y_1 \\
 & x_2 & \\
 & x_2 & \\
\text{LAYER 2} & & \Longrightarrow y_2 \\
 & x_3 & \\
 & x_1 & \\
\text{LAYER 3} & & \Longrightarrow y_3 \\
 & x_3 &
\end{array}$$

Figure 2: All combinations of variables are independent and can be executed in parallel.

process is independent and communicates with other processes by channels. For example, if we have three independent variables in the first layer, three combinations result in estimating three local polynomials as shown in Figure 2.

In the second layer, another set of local polynomials should be figured out in a similar manner. In fact, in order to make the algorithm more efficient, a coordinator (host) and an array of processing elements (PE) are needed. The process to calculate the polynomial from the combination of two independent variables is transmitted from the host to the three processing elements and each is provided with a different input. The three processing elements perform their work simultaneously and calculate the required polynomials. These calculated values are transmitted back to the host and further calculations are performed, before moving to the next layer of the gmdh algorithm. The basic steps in constructing a gmdh algorithm are as follows:

- the original set of data is divided into training and checking sequences, by separation rule being a heuristic one. Usually the training and the checking sequences are taken alternately.

- quadratic polynomials are formed for all possible combinations of x_i variable taking two at a time.

- for each polynomial a system of normal gaussian equations is constructed using all the data points in the training set. By solving these equations, the values of the intermediate variables y_i are determined.

- the models are used to predict the system response in the training set data region. The predictions are passed through some form of selection criteria, the most widely used being the mean square error.

- the models y_i, are ordered with respect to the smallest mean square error. The models with mean square error less then specified thresholds are allowed to pass to the next level of gmdh. The number of functions selected at this level is arbitrary.

- at the next level the independent variables for the new training and checking sets are found by mapping the original training and checking sets through the single layer that has been formed.

- new polynomials are formed and the above steps are repeated for each layer. As each new layer is formed, the smallest mean square error is stored and plotted as a function of layer number.

- the procedure is terminated when the smallest overall mean square error is reached at any level.

The global minimum is the point of optimum complexity for this choice of network heuristics.

4 Algorithm Implementation

Traditionally, groups of workstations are used as a resource for a series of sequential jobs, but with the development of message-passing software systems a cluster could also work together on a single problem. While this use of clusters is formally less efficient than running the jobs in a serial fashion, increased throughput can be obtained for time-critical applications. In addition, workstation clusters are an attractive and cost effective parallel software development platform. An application can readily be developed on a group of workstations and ported to moderately parallel machines by only making changes to the software that are inherent to the different FORTRAN and/or C compilers. The two programming tools used for parallel implementations of gmdh were Parallel Virtual Machine (PVM), an Network Linda (LINDA). All the work was done on a Cluster of DEC workstations networked with an Ethernet. These two tools were selected because they were based upon UNIX rather than a proprietary operating system; they are available on a wide range of machines, and they are sufficiently high-level programming tools.

LINDA grew out of research work done at Yale University. Scientific Computing Associates is the company formed to bring the product to market. LINDA consists of six operations that access an object called "tuple" space. Tuple space is basically shared memory that is "global" to all the workstations in the network.

The PVM (Geist et al., 1993) system is a programming environment for the development and execution of large parallel applications that consists of many interacting but relatively independent components. It is intended to operate on a collection of heterogeneous computing systems interconnected by one or more networks. The participating processors may be scalar machines, multiprocessors, or special- purpose computers. PVM provides a straight forward and general interface that permits the description of various types of algorithms while the underlying infrastructure permits the execution of applications on a virtual computing environment that supports multiple parallel computation models. PVM is a software package being developed by Oak Ridge National Laboratory. It enables a heterogeneous collection of Unix computers linked by a network to operate as a single large parallel computer. Thus, large computational problems can be solved by the aggregate power and memory of many computers. PVM supplies the functions to start tasks and lets the computers communicate and synchronize with each other. It survives the failure of one or more connected computers and supplies functions for users to make their applications fault tolerant. Users can write applications in Fortran or C and parallel them by calling then PVM message-passing routines. By sending and receiving messages, application subtasks can cooperate to solve a problem in parallel. PVM applications can be run transparently across a variety of architectures. The PVM source code and user's guide are available from netlib@ornl.gov. The source has been tested on various types of workstations and parallel computers. In addition, several vendors such as DEC (ALPHA) supply PVM software optimized for their systems. While PVM provides a solid programming base, it does not give many options for debugging PVM programs. To help the user in the development of PVM programs, Xab, a run time monitor (also available from netlib) can be used to learn about the PVM programs performance. In developing software, the programmer often designs the initial specifications graphically. Designers can visualize the problem's overall structure far more easily from this graphical representation than from textual specifications. With the Hence graphics interface (also available from netlib) implemented on a workstation, a user can develop a parallel program as a computational graph. The nodes in the graph represent the computations to be performed and the arcs represent the dependencies between the computations. From graphical representation, Hence can generate a lower level portable program, which when executed will perform the computations specified by the graph in an order consistent with the dependencies specified. Such a representation enhances the quality of the resulting software. Real-time debugging, visualizing and monitoring are particularly important in a heterogeneous multiprogramming environment where differences in computation and communication speeds result from both heterogeneity and network loads.

We have tested parallel gmdh algorithm using various programming tools and architectures, including simulators such as Parallaxis (Braunl 1992), networks of workstations and PVM, also real massive parallel processors. As testing set of data we have selected data from our previous research (Wasniowski & Wilk 1988). The detailed results of testing the parallel gmdh algorithm can be found in (Wasniowski 1993). In Figure 3 we present an example of output produced by gmdh algorithm.

5 Conclusion

Several methods of self-organizing algorithms relating to the system identification and design have been developed, and a common feature of most of them is that they are computationally intensive. To provide an acceptable speed of response for such algorithms the exploitation of inherent parallelism via parallel computing is possible. For the gmdh algorithm the heavy burden of arithmetic

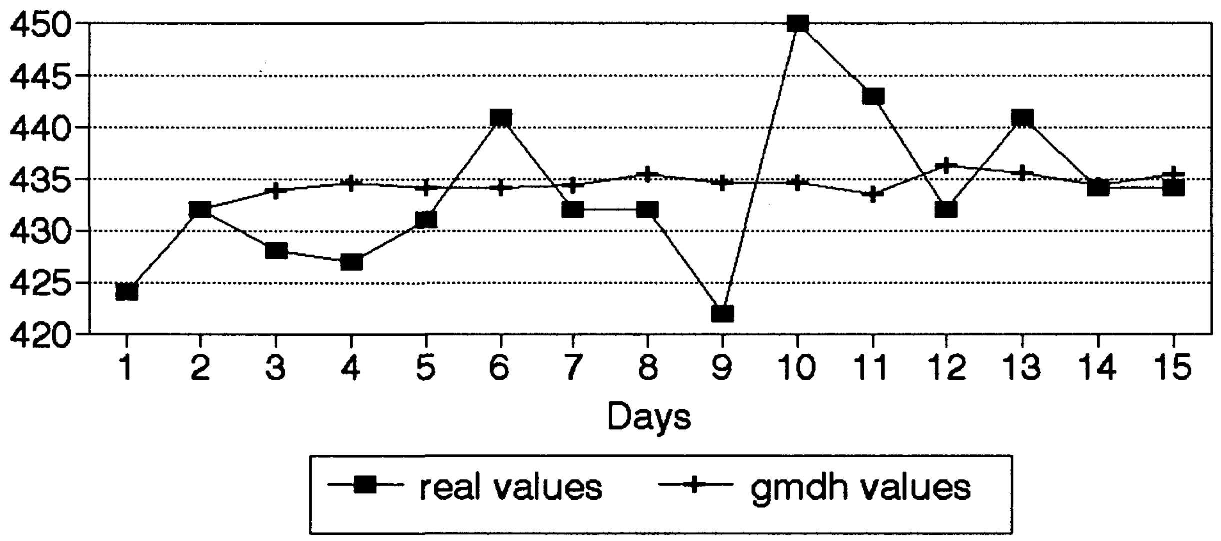


Figure 3: Example of output produced by gmdh algorithm.

processing can easily be implemented on parallel computers. There is a wide variety of distributed computing tools available both in the commercial marketplace as well as the public domain. We have chosen, implemented, and evaluated two of the more common tools (PVM, and LINDA). These tools provide the user with great flexibility in the manner in which parallelism can be exploited. The tools differ in their implementation as well as their utilization. It should be pointed out that probably the most heavily used and implemented of all the tools is PVM. We have shown that cluster of workstations can be effective supercomputers for implementing gmdh algorithm when combined with appropriate software tools and implemented in a parallel fashion.

References

[1] Barto A.G., Sutton R.S. , Anderson C.W. (1983) Neuron Like Adaptive Elements that Can Solve Difficult Learning Control Problems. *IEEE Trans. on Systems, Man and Cybernetics*, Vol. SMC-13, No 5.

[2] Benner R.E., Harris J.M. (1991) Applications, Algorithms, and Software for Massively Parallel Computing. *AT&T Technical Journal*, November/December, p. 59-72.

[3] Braunl Th. (1992) Simulation System for Massive Parallelism. *International Journal in Computer Simulation* , 2, p. 231-250.

[4] Duffy J.J. , Franklin M.A. (1975) A Learning Identification Algorithm and its Application to an Environmental System. *IEEE Transactions on System, Man and Cybernetics*, 5(2), p. 226-240.

[5] Geist A., et al., (1993) PVM 3.0 User's Guide and Reference Manual, Oak Ridge National Laboratory, Tennessee, February.

[6] Heath M., Etheridge J. (1991) Visualizing the Performance of Parallel Programs. *IEEE Software*, Vol. 8, No.5, p.29-39.

[7] Ikeda M.O., Sawaragi Y. (1976) Sequential gmdh Algorithm and its Application to River Flow Prediction. *IEEE Transactions on Systems, Man and Cybernetics*, 6(7), p. 473-479.

[8] Ivakhnienko A.G. (1969) Self-Teaching Systems of Recognition and Automatic Control, Tekhnika.

[9] Ivakhnienko A.G. (1971) Polynomial Theory of Complex Systems. *IEEE Transactions on Systems, Man and Cybernetics*, 1(4), p. 364-378.

[10] Madala H.R.(1991) Comparison of Inductive Versus Deductive Learning Networks. *Complex Systems* , 5(2), p. 239-258.

[11] Muller J.A.(1990) Macroeconomic Modeling by Means of Self-organizing Methods. *11th IFAC* , Tallinn, Vol. 12, p. 50-55.

[12] Nishikawa T., Shimizu S. (1985) An Analysis of the Generalized Ridge Estimator in the Adaptive gmdh. *J. Soc. Sys. Eng.* , 9, p. 53-66.

[13] Tenorio M., Lee W. T. (1990) Self-organizing network for optimum supervised learning. *IEEE Transactions on Neural Networks* , 1(1), p. 100-109.

[14] Wasniowski R, Wilk T. (1988) An Algorithm of Forecasting in Active Systems. *Scientific Papers of the Forecasting Research Center*, No.10, p. 31-44.

[15] Wasniowski R. (1993) The gmdh algorithm and its parallel implementation, Research Report NMHU-RAW 2-93.

Termination Conditions for Parallel Shape Recognition

Zbigniew M. Wojcik and Barbara E. Wojcik
Smart Machines, 13703 Morningbluff Dr.,
San Antonio, TX 78216, USA.

Keywords: parallel shape recognition

Edited by: Jerzy R. Nawrocki
Received: February 28, 1995 **Revised:** July 30, 1995 **Accepted:** September 6, 1995

Elementary features are detected by calculating the number of objects inside partly overlapping windows fixed in an image plane. Each window's contents is processed by a separate processing element (PE) on a SIMD grid or pyramid architecture. Two neighboring feature chunks are merged by adjacent PEs, and the merged feature chunks are joined in each next lth parallel step by every 2^lth PE possessing complementary and adjacent feature. The shape recognition terminates if a complete set of features (such as curve bounds, endings, corners, segments, regions forks and junctions) making a whole object meet together on one PE. This complete set is proved to be sufficient to terminate the shape recognition. Hierarchy of elementary features for a merging process is established to recognize object bounds and more primitive features first to have a complete feature set adequate for the consideration by the termination condition at a given level of parallel image recognition (i.e., primal sketch, $2\frac{1}{2}D$ sketch *and* 3D world model*). The approach has the property of mapping of image segments directly into phrases and English sentences.*

1 Introduction

Many present day computer vision systems analyze shapes sequentially. Can a computer see the entire picture at once? If so, how one processor analyzing one small image fragment will know about the processing performed by many other processors attached to all other image fragments? The first pyramid computers process in parallel [8, 16, 19] pieces of contours detected by template matching. We approach parallel shape analysis using a SIMD grid architecture by merging elementary features detected during the calculation of the number of objects within a window.

Propagation component labeling [2] determines, in parallel, the connectedness of pixels of the same gray level (assuming that each pixel has its own processor). Given an $N \times N$ image, the row-major label $Ni+j$ with $i, j = 0, 1, ..., N-1$ is assigned to each image pixel. Two neighboring pixels (or subregions) of the same (or similar) gray level receive the same (the smallest) label. Initially, only the pixel at $(0,0)$ has its own label if it is of a grey level looked for. All adjacent pixels with the same gray level will assume that label in each new iteration. The final label is the smallest label of any of the connected members. Execution is complete when there are no more label changes. Before propagation labeling is completed, it is not known which label is the minimum (i.e., permanent). Labeling of a spiral (the worst case) will take N^2 steps.

Propagation component labeling does not provide any information about shape. It is also sensitive to accidental discontinuities. The reassignment of the labels in each new step is time consuming. One improvement would be to record addresses of all pixels tentatively labeled (i.e., in the previous step), and then to propagate the new minimum label to the known addresses of PEs. However, very long strings of addresses would have to be broadcast to PEs together with the labels.

It will be shown in this paper, that shape codes can be handled more efficiently than by recording and broadcasting the addresses of pixels. Shape can be recognized directly from window gray levels. Our algorithm for shape recognition is faster than the propagation component labeling: it does not waste time re-labeling pixels and broadcasting addresses of individual pixels. Instead, shape codes (representations) are transmitted carrying all the necessary information about the shape components. Information about individual pixels is involved implicit in shape code.

Sollin's component algorithm for graphs [6] uses a *divide-and-conquer* strategy. This algorithm is based on a quadtree decomposition of shapes on the image plane. The image is divided into quadrants. Each quadrant (greater than 1 pixel) is labeled recursively (i.e., by dividing each subsequent quadrant into quadrants up to 1 pixel quadrant). Minimum label is passed through the adjacent quadrant boundaries in each recursive step. The *divide-and-conquer* strategy is of a logarithmic complexity, but each re-labeling step is very long, as in the propagation component labeling. Re-labeling takes place at fixed quadrant boundaries. Some PEs can specialize in broadcasting labels: single pixel quadrants communicate with single pixels, 4-pixel quadrants with 4-pixel quadrants, 4^n-pixel quadrants with 4^n-pixel quadrants. Broadcast of addresses of pixels having uniform labels can be efficiently organized, e.g., incorporating a pyramid architecture. But addresses of all individual pixels carrying the same label must still be handled.

Methods for finding extreme pixels in each quadrant may ignore a discontinuity inside a quadrant and as a result may merge two or more disjoint objects into one entity, a drawback of the *all-components extreme point approach* [20, 21] Improving this approach, extreme points of separate objects in a quadrant should be found as a result of segmentation by the divide-and-conquer strategy or by propagation component labeling. Extreme points of individual objects can be found sequentially by an effective algorithm in a contour tracking process [25].

Neither the divide-and-conquer strategy nor the propagation component labeling are capable of recognizing shapes. Even the improved version (suggested above, by propagation of addresses for re-labeling) can handle only addresses of individual pixels, because shapes of re-labeled blobs are not coded. The approaches count adjacent pixels of a similar gray level only and extract them from an image: in addition, they must be followed by a shape approximation process if a shape representation is required.

A pyramid technique for parallel detection of smooth curves [8] depends on a bottom-up analysis of the adjacency of elementary curve elements in neighboring blocks (windows) by a higher-level pyramid cell. Each cell of the pyramid records the position and slope of a curved strip and positions of endpoints of the strip. Higher-level cells record more distant endpoints (since a larger block of input image is covered by a higher-level cell). Some higher-level cell will be a root recording a consistent smooth curve. The pyramid architecture designates certain cells in advance to identify many possible occurrences of features. The coordinates of endpoints go directly up the pyramid making a computational burden for some higher level pyramid cell if several curves are detected in the same large field belonging to the cell. The time complexity for such computations is $O(number\_of\_curves)$. For instance, two nearly parallel curves must be handled by the same cell at some level. This higher-level cell will be a bottleneck for more lines found in its field. A search is needed to match endpoints of strips found in adjacent subfields at a lower level. Doubling back of the same curve will also require the same cell at some level for identification.

In a process of collecting features such as forks, junctions, regions, corners and curve bounds connected through edges, a node is capable of making a decision whether to look around for some missing features in the image or to complete the image recognition. Classes of objects are composed of some specific numbers of these features. For example, a polygon has only one region. A corner view of a cube has three regions, four forks, nine edges and three corners. The termination condition for a recognition process may be based on the graph theory concepts using some relations between object components. *Euler's Theorem* [5, 15] defines relation between the number e of edges, the number v of vertices in a planar graph, and the number r of regions (including one unbounded region external to the graph), using the equation: $r = e - v + 2$. The *Handshaking Theorem* $2e = \sum_{v \in V} deg(v)$ [5, 15] expresses the relation between the number of edges e and the degree *deg* of all vertices V making a graph. The above two Theorems can be used in computer vision as a termination condition after disregarding an unbounded region and assigning a meaning 'corner' to a vertex of degree 2, 'fork' to a vertex of degree 3, etc., and 'object edge' to graph edge, and 'image region' to the graph region.

Euler's Theorem does not classify vertices (i.e., does not distinguish between corner, ending, fork, junction, etc.), and the *Handshaking Theorem* does not consider regions, which would be disadvantageous for a termination condition of recognition procedures in computer vision. Combination of them both treats all vertices as equally significant. However, in computer vision the line endings are the least meaningful and should be easily disregarded if a contour is blurred in the image. In general, recognition levels called *primal sketch*, $2\frac{1}{2}D$ *sketch* and *3D world model* are distinguished in literature [3, 14, 18, 22, 23] and should be considered by termination conditions in shape processing.

We describe relations between the numbers of lines (i.e., straight segments or curves) and corners, forks, junctions and regions for computer vision purposes in a way different than used in the literature [3, 10, 14, 18, 22, 23]. Tips are not considered explicit in our termination conditions, but any line is constrained as delimited by either endings, or corners, or curve bounds, or forks, or junctions or by combinations on them. In this way endings are implicit in the number of lines in our termination conditions.

As in [8], the merging of adjacent features is performed by the nodes one level higher. The image plane is divided into quadrants. We propose a SIMD architecture similar to [8], with the following differences: a) We use special termination conditions for every node to know whether the parallel shape recognition is completed when having some set of features. For instance, the resulting approximation is sent one level up if a complete set of features has not been found yet. b) A few termination conditions are set up, each appropriate for making the *primal sketch*, $2\frac{1}{2}D$ *sketch* and 3D *world model* in parallel. c) The termination condition for the *primal sketch* has the highest priority. The $2\frac{1}{2}D$ *sketch* has a lower priority, and the lowest priority is for the 3D (3-dimensional) *world model*. d) The detection of the elementary features is done by calculation of the window objects instead using the template matching. e) The approach has the property of mapping image segments directly into words and phrases.

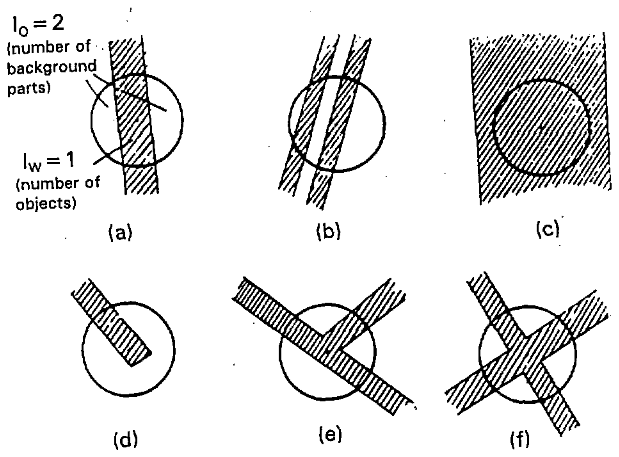


Figure 1: Window approximation by mapping of window gray levels into feature name using the number I_w of window objects and the number I_o of window background parts. The symbolic window approximations are: a) segment, b) two segments; c) interior; d) ending; e) fork; f) junction

2 Feature Detection and Feature Merging

Elementary features are detected in a window using the numbers of window objects [24-26]. Different combinations of the numbers are characteristic to specific image features (as segment, fork, ending, junction, discontinuity). For instance, the name "segment" is achieved if the number I_w of objects equals 1 inside the window and the number I_o of background parts is 2 (Fig. 1). For a junction, I_w=1 and I_o=4. For a fork, I_w=1 and I_o=3. The same numbers are obtained for all possible rotations of the same feature inside the window and for variable thicknesses of the feature. Only one single analysis of window contents (gray levels) is necessary in our method.

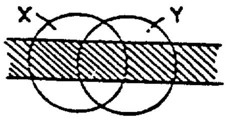
Elementary features, especially segments, detected in parallel by PEs of a grid or a pyramid net are merged into chunks by higher level nodes. When merging, every two consecutive nodes are

$$r\underline{A}(X+Y) = \underline{A}X = \underline{A}Y = \underline{A}(X+Y) = \text{segment}, \quad \text{if } s_X \approx s_Y \text{ and } \underline{A}X = \underline{A}Y = \text{segment},$$

where:

$$s_X \approx s_Y \quad <=> \quad |s_X - s_Y| \leq \Delta s ;$$

r denotes the reduction; Δs is an admissible tolerance for s (slope).



$$rA(X+Y) = \underline{A}X + \text{attributes} \bullet \text{parameters} =$$

$$= \underline{A}X + (\text{mass,position,length,slope}) \bullet (m,(x,y),l,s),$$

$$\text{if } s_X \approx s_Y \text{ and } \underline{A}X = \underline{A}Y = \text{segment}.$$

where:

$$m = m_X + m_Y$$
$$x = (x_X + x_Y)/2$$
$$y = (y_X + y_Y)/2$$
$$l = l_X + l_Y$$
$$s = (s_X + s_Y)/2$$

<u>No reduction:</u>

$$rA(X+Y) = AX + AY, \quad \text{if } \underline{A}X \neq \text{segment or } \underline{A}Y \neq \text{segment}.$$

Figure 2: Symbolic approximation $\underline{A}(X \cup Y)$ of two partly overlapping windows containing segment and approximation $A(X \cup Y)$ of the reduced two-window segment involving both the symbolic and a required quantitative window approximation

replaced by one node when the corresponding segment slopes are similar. Reduction of elementary segments to a straight line or to a curve is performed in this way (Fig. 2). The merged chunks are united in turn into greater pieces, and so on, until the entire shape has been coded in parallel [9]. Maximum of $2^{n/l}$ merging operations can be done in each subsequent lth parallel step for a linear feature occupying $2^n = N$ contiguous PEs. In order to know where to look for some missing chunks, slopes of the possessed feature pieces are used. The missing feature chunks are looked for in the expected directions. Knowledge on the maximum size of the pieces looked for is also available at a given stage of parallel processing: feature size does not exceed the quadrant size belonging to the node at a current level of processing.

In our method, names of features and their parameters (including the count of pixels of the same property) are handled instead of propagating a label having *nothing in common* with the shape name. In particular, a rough set composed of elements consisting of feature names with attached attributes and parameters [24] is broadcast to fixed-address nodes (PEs). The receivers identify more complex features, based on feature adjacency, expectations and constraints. A rough approximation of larger pieces of shapes is done to reduce to a minimum the amount of information representing an object. The rough approximations deal both with symbols (names of features and objects) and with quantitative representations (e.g., with positions, slopes and dimensions of features). Instead of a long process of re-labeling of many pixels, merging of features is performed. A *rough grammar* [29, 30] can be applied to support the divide-and-conquer strategy. A data-driven execution can guide the parallel shape recognition: every leaf PE will wait until the window gray levels will become available for the detection of elementary features.

3 A Parallel Recognition of Edges and Regions

A rooted tree structure is used. At the bottom level of parallel processing, the leaves are the contents of the windows fixed in the image plane, and then the nodes are the ancestors of the lower level sub-trees. Data received from the 4-children is used for merging of segments, endings, curve bounds, forks and junctions, if the object approximation is not complete. A complete shape approximation is transferable to a host.

At the bottom (first) level, all the nodes (PEs) detect their elementary features in parallel by calculating the numbers of window objects [24-26]. Each window (Fig. 3) is under the control of a separate PE. Feature name and its particular quantitative parameters (e.g., segment coordinates, slope, length, mass and thickness) are detected by each PE. The same pixels will belong to a number of overlapping windows. To prevent a single object fragment from multiple processing by several PEs, each overlapping window sends its feature approximation to its neighbors in the directions different from the slope of the possessed segment. Among the selected neighbors sharing the same object fragment, one window is chosen

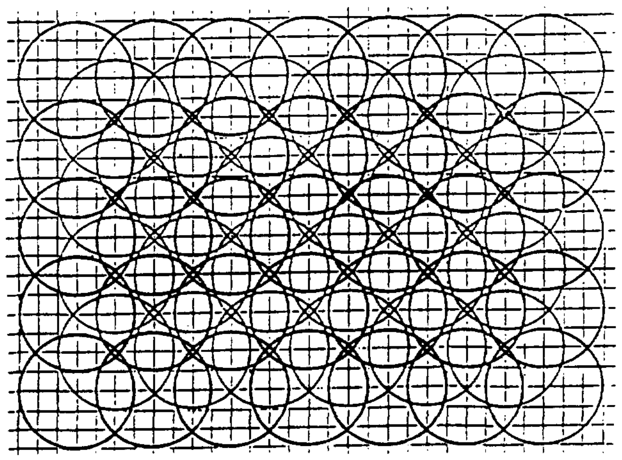


Figure 3: Overlapping windows for parallel shape recognition

having the highest I_w (the number of objects). From among a few overlapping windows with the same (highest) I_w, a window is chosen with a feature of the maximum mass.

Any PE staying active sends its feature approximation to its parent. This feature approximation is merged by the receiver to feature chunks received from other children. The result of merging is sent to the ancestor examining its four children, provided recognition of the entire object was not completed yet. Getting a full set of the features, a parent completes the object recognition.

Example 1. PEs#(4,11) and #(5,10) send their elementary features to their parent PE#(5,11) at the first level of parallel processing of OBJECT 5 in Fig. 4. The PE#(5,11) does not have two endings of the possessed chunk, so it does not announce recognition of a whole object, but sends its approximation to its ancestor, PE#(7,11).

Example 2. PE#(2,8) in Fig. 4 sends its ending approximation to the parent PE#(3,9). At the 3rd level of parallel processing, the PE#(3,7) merges the ending detected by the #(1,6) (and received through 2nd level #1,7)) to the segment detected by PE#(2,7). The merging of the three element curve (OBJECT 1) involves 5 levels of parallel processing on a pyramidal architecture, because OBJECT 1 is located on the border of different quadrants. The 4th level PE#(7,7) still will not receive the ending detected by #(2,8), but the 5th level PE#(15,15) will announce recognition of OBJECT 1, because one segment continuous with two endings constitutes a full set of features for the simplest object.

In general, the results of merging are sent to a nearest doubly odd PE # $(2^l i-1, 2^l j-1), (2^l i-1, 2^l j-1) \in \{\{1,2,...,N-1\},\{1,2,...,N-1\}\}$, $N=2^n$, on a grid configuration, where merging is done on any couple of adjacent segments (Fig. 2) by execution of $a(f_{own} \cup f_{neighbor}, s\}$, where s is a merging operation, f is a feature approximation [24], a is a requested approximation of merged features (qualitative $\underline{A}$ or quantitative A, see Fig. 2), and l is the level of parallel processing. Merging is skipped if two feature chunks do not match each another symbolically.

Example 3. At the third parallel processing level (Fig. 4), the ancestor PE#(7,7) merges three feature chunks received from its children PEs#(3,3), #(3,7) and #(7,7). The chunks involve two endings and make a full set of features, so PE #(7,7) announces identification of the entire curve (OBJECT 3).

Example 4. On the second level of parallel processing, the nodes #(13,1), #(13,3) and #(15,3) send approximations of their chunks to their parent PE#(15,5). The 3rd level PE#(15,5) and PE#(15,7) send their features to 4th level PE#(15,7) which performs merging and having two endings and a one line announces identification of the curve (OBJECT 2).

A formal specification of the termination condition for parallel shape recognition is helpful for appropriate control of execution of programs by nodes (e.g., to know, when segmentation of a single object is completed when having several distinct objects).

THEOREM 1. Parallel recognition of shapes involving only endings, straights, curves and corners (without regions, forks or junctions) is completed, if:

$$L = C + 1, \qquad (1)$$

where C is the number of corners or curvature bounds (such as upper, lower, left and right bounds), L is the number of curves or segments with two endings (e.g., connecting: corner - corner, bound - bound, corner - ending, bound - ending, corner - bound, ending - ending).

PROOF (inductive). Consider Figs. 4 and 5.a as examples of simple drawings. Having no cor-

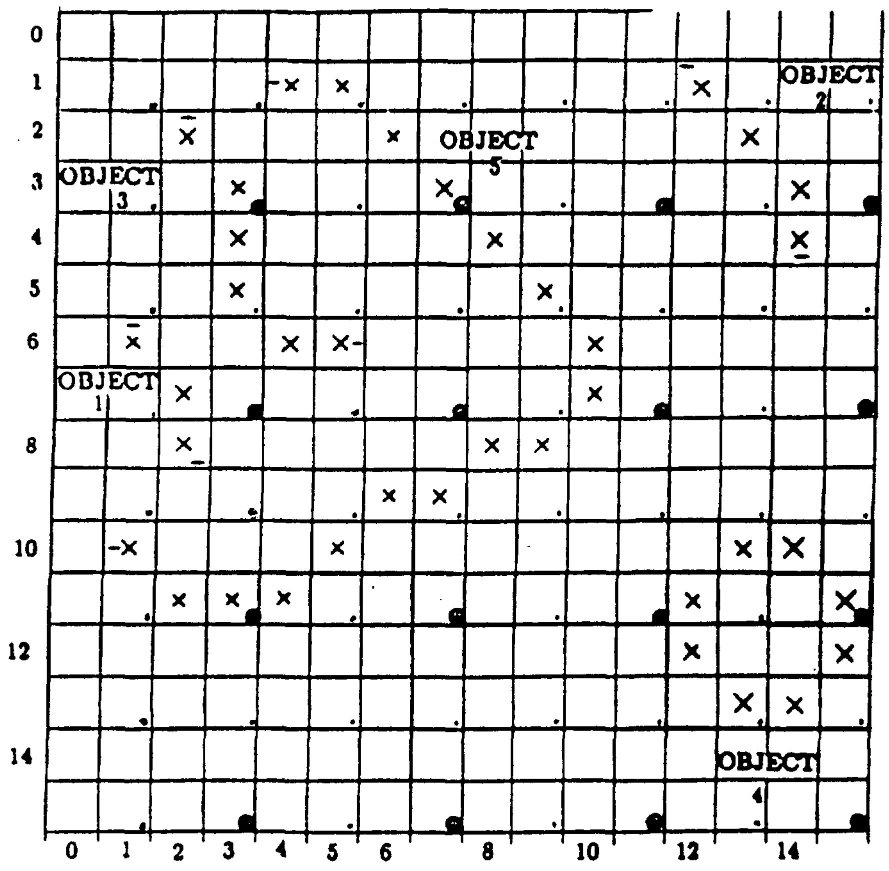


Figure 4: Examples of simple drawings on a grid configuration. Each square represents a PE. To involve the functions of a pyramid, some selected grid nodes are assigned also the functions of nodes of higher level pyramid nodes. Squares with one dot represent both the leaves and the second level nodes of a pyramid, squares with one dot and circle represent both the leaves, the second level nodes and the third level nodes

ners nor curvature bounds we have $L = 1$ (i.e., in the basis step there is only one ending - ending connection). When incrementing L by one, the first and the last segment (or curve) must have a ending (because in this THEOREM a drawing is assumed to be without forks, junctions and regions). In the inductive step let us notice, that any new single corner or curvature bound (denoted by C) increments L by one, and then the true equation $(L + 1) = (C + 1) + 1$ reducing to Eq.(1) is achieved when assuming the truth of the inductive hypothesis (Eq. (1)). •

Any element which does not have the entire object (i.e., with a complete set of endings, curvature bounds and lines (THEOREM 1)) sends the possessed object approximation one level up to its ancestor.

Example 5. OBJECT 5 in Fig. 4 has one corner (or right bound represented by PEs #(10,6) and #(10,7)) two endings and two lines, so $L = 1+1 =$

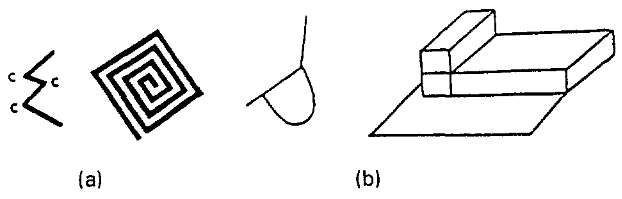


Figure 5: A few simple drawings satisfying: a) Eq.(1); b) Eqs.(2) and (4) (the last 3D object with shading having: L=24 lines, C=5 corners, R= 7 regions, F=9 forks, $J = |features_4| = 1$ junctions and $|features_5| = 1$ and yielding: $L = 24 = |features_5| * 4 + J * 3 + F * 2 + C * 5 + 1 - 7 = 1 * 4 + 1 * 3 + 9 * 2 + 5 + 1 - 7$

2. The 5th level ancestor (root), PE#(15,15), receives two endings from 4th level children, PEs #(7,7) and #(7,15), and detects one right bound based on approximations received from 4th level PEs #(15,7) and #(15,15). The 4th level

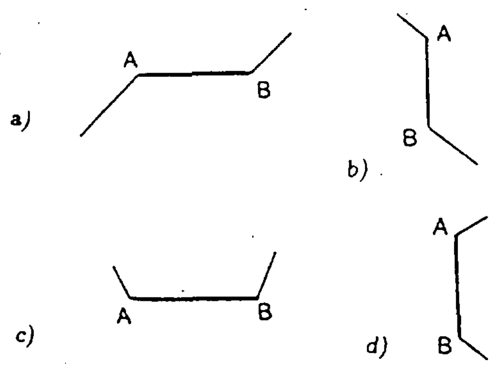


Figure 6: Different type endings A, B of a flat segment: a), b) non-homogeneous signify no curve bound; c), d) homogeneous endings signify a curve bound

PE#(7,15) does not announce identification of an entire object, because it has only one ending. The 5th level PE#(15,15) has a complete set of features, hence it announces recognition of the entire OBJECT 5.

Many horizontal or vertical segments do not form object bounds and then they do not affect the counters C and L in Eq.(1). Any horizontal segment ended by a non-homogeneous ending (e.g., either by a left-lower and right-upper bounds (Fig. 6.a), or by upper-left and lower-right bounds (Fig. 6.b)) does not form curve bounds and as such does not increment L nor C.

LEMMA 1. Any flat segment (horizontal or vertical) delimited by a nonhomogeneous ending (see Fig. 6) can be taken apart from a curve bound.

PROOF. Let A and B be nonhomogeneous endings of a flat segment. Approximations of the two nonhomogeneous endings meet at some level of parallel processing contradicting a bound of a horizontal segment (Fig. 6.a) or a bound of a vertical edge (Fig. 6.b). Only homogeneous endings (i.e., either lower-left and lower-right, or upper-left and upper-right (Fig. 6.c), or upper-right and lower-right (Fig. 6.d), or upper-left and lower-left) meeting at some node of the processing tree and delimiting the same segment form a curve bound. ●

LEMMA 2. Complexity of the detection of a flat nonhomogeneous segment or a curvature bound is log_2 (*flat_segment_length*) or log_2 (*curve_bound_length*) steps.

PROOF. Endings of the same flat segment (hypothesized bound) occur to be contradictory after log_2 (*flat_segment_length*) steps, and information about this contradiction is utilized or transferred up instantly. Information about non-contradictory segments is processed in the same way. ●

Any horizontal or vertical segment may be coded in parallel according to the method presented in [28] (see Fig. 7). The described above parallel processing associated with the *primal sketch* goes through levels until the entire continuous linear object has been collected (i.e., until Eq.(1) is satisfied), accessing the children distant by 2^l or by $\sqrt{2^{2l}+2^{2l}}$ nodes in each lth level of processing. Any fetched feature *chunk*1 is merged with the matching possessed feature *chunk*2 by executing a(*chunk*1 ∪ *chunk*2, s), where a denotes the symbolic or quantitative approximation of the operands listed in the parentheses (compare Fig. 2), and s is the merging operation.

THEOREM 2. Parallel recognition of shapes involving endings, straights, curves, corners and regions and without forks or junctions is completed, if:

$$L = C + 1 - R, \qquad (2)$$

where R is the number elementary regions, L and C have the same meanings as in Eq.(1).

PROOF (inductive). For an object without regions, forks and junction the Eq.(2) reduces to Eq.(1). For an object consisting of one region only (i.e., for a circle) we have L=4, C=4 and R=1, so the basic step is proved. For the inductive step observe, that any new single region decrements L by 1 changing the inductive hypothesis (2) to $L - 1 = C + 1 - (R + 1)$ and satisfying directly Eq.(2). ●

Example 6. There are no endings in the OBJECT 4 in Fig. 4. In the 2nd level of processing, the PEs #(15,11) and #(13,11) send their approximations to 3rd level PE #(15,11), and PEs #(15,13) and #(13,13) to 3rd level PE #(15,15). These 3rd level PEs detect the upper and lower object bounds, but they do not have the full sets of features. The 4th level PE#(15,15) detects the region and two bounds more: left and

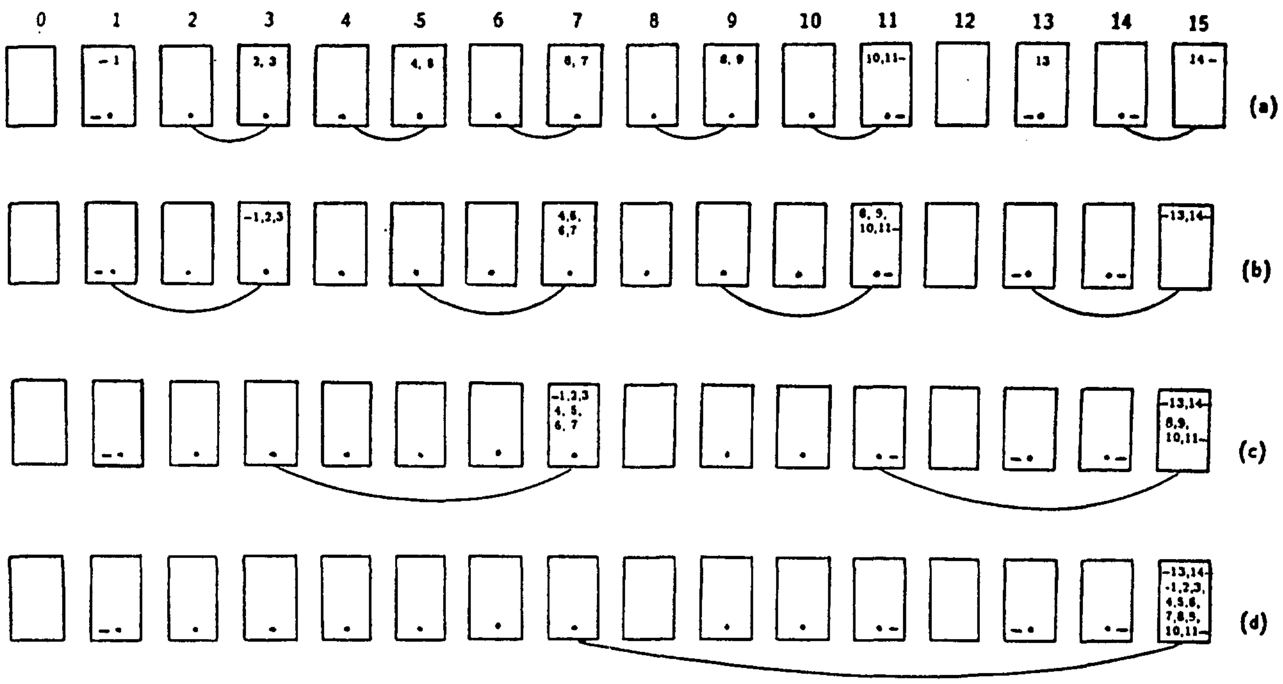


Figure 7: Stages of parallel shape recognition of two disjoint horizontal (or vertical) segments on a two-dimensional array of PEs [28]. Dots represent presence of elementary segment detected by a PE, arcs show the transfers of messages containing feature chunks, mark '-' denotes a segment bound or ending: a), b), c) merging at the second, third and fourth levels of the parallel processing. PE#15 receives full set of the short segment in the third stage of parallel processing, and the long segment in the fifth stage (d)

right. It has the full set of features satisfying Eq.(2), so it announces recognition of the circular object.

4 Involving Forks and Junctions

Endings are given the highest priority in merging to adjacent features, because endings are not counted by Eqs.(1, 2). Somewhat lower priorities should be assigned to curvature bounds, corners and horizontal or vertical nonhomogeneous segments when merging them with oblique segments when making the *primal sketch* when collecting patches or blobs using Eq.(1). When detecting regions from patches, the patches are expected to be reconstructed already from bounds, corners and curves. Patches should have a lower priority than regions when making the $2\frac{1}{2}D$ *sketch* using Eq.(2). One approach in processing shapes is to assume that the priorities of processing forks and junctions are lower than those for regions and patches, because forks and junctions form 3D objects from the $2\frac{1}{2}D$ *sketch*. Then any regions and patches are merged to forks and to junctions. Forks and junctions incur more complexity. Shapes involving forks and junctions are also amenable to a rule checking a full collection of primitive components of 3D objects as shown below.

THEOREM 3. Parallel recognition of shape involving forks and junctions is completed if the following terminations condition is satisfied:

$$L = J * 3 + F * 2 + C + 1 - R, \qquad (3)$$

where J is the mumber of junctions; F is the number of forks; L, C, R have the same meanings as in Eq.(2).

PROOF (inductive). We assume the truth of Eq.(2) for the basis step. Let Eq.(3) be the inductive hypothesis. Any single fork increments L by two giving $L + 2 = J * 3 + (F + 1) * 2 + C + 1 - R$ and reducing to Eq.(3), and any single new junction increments L by three giving $L + 3 = (J + 3) * 3 + F * 2 + C + 1 - R$ and reducing to Eq.(3). •

The lines L, bounds C, forks F, junctions J and regions R (i.e. delimited by closed curves) are counted by ancestors on each level of parallel processing. Because of a higher complexity and lowest priority of forks and junctions, Eq.(3) can be checked only at nodes detecting forks and junctions. Partial approximations of object fragments adjacent to the forks and junctions will be sent to these nodes.

THEOREM 4. Parallel recognition of shape involving any elementary features detectable by the numbers I_o and I_w of window objects (Fig. 1) is completed if:

$$L = \sum_{features_f} |features_f| * f + 1 - R \qquad (4)$$

where $|features_f|$ is the mumber of features of the order f. The *feature order* f is defined by the number I_o of window background parts minus one, i.e.:

$$f = I_o - 1 \qquad (5)$$

For example, $|features_f| = F$ for $f = 2$ is the number of forks, and $|features_f| = C$ for $f = 1$ is the number of corners or curve bounds.

PROOF (inductive). We assume the truth of Eq.(3) for the basis step (i.e., when having 0 features of orders $f > 2$). Let Eq.(4) be the inductive hypothesis. Assuming the presence of only the features of one selected order $f > 2$ and junctions J, forks F, corners of bounds C with possible regions, any new single f-order feature, $f > 2$, increments L by f yielding $L + f = (|features_f| * f + 1) + J * 3 + F * 2 + C + 1 - R$ and reducing to Eq.(4). The same is true for features of any order f. •

DEFINITION 1. An object is said to be recognized if the numbers L, C, F, J, $|features_f|$, R satisfy one of the Eqs.(1), (2), (3) or (4), and if each of its lines contains two endings, such as ending-ending, or ending-bound, or bound-bound, or ending-fork, or bound-junction, etc.

THEOREM 5. When object shape recognition is terminated, the root of the parallel computation has a complete set of features (i.e., one of Eqs.(1), (2), (3) or (4) is satisfied at the root).

PROOF. In each step of parallel processing the number of branches of any computation tree decreases because of the merging processes. Because of different priorities, Eq.(1) is checked first to detect all 2D drawings and patches. If the drawings are part of regions or are adjacent to regions and planes, then Eq.(1) is not satisfied because it does not consider regions. If image regions are considered, then the regions may be a part of 3D objects or adjacent to them, and then Eq.(2) is not satisfied because it does not consider forks and junctions formed by regions or planes. Hence, Eq.(3) must be used, or even (4). Only two cases are possible: a) A node processing one object contains a partial set of features (e.g., the shape recognition is not completed yet or an object is only partly located in the image plane), or b) a node is a root and has a complete set of features. •

THEOREM 6. Any feature or object merged and collected by a node in the last step of a completed parallel processing is continuous.

PROOF. If there is a full set of features collected by the node then all the features must make one entity. Any disjoint set of features (in the image plane) has its own full set of features (satisfying one of Eqs. (1) to (4)). •

LEMMA 3.
The complexity of the parallel shape recognition is $O(log_2(object\_maximum\_length))$ parallel stages.

PROOF. When collecting (and merging) features of an object of the maximum length N, $2^{l-1} \leq N \leq 2^l$, each new step involves 2 times nodes less, where l is the level of parallel processing. •

5 Conclusions

The step of parallel shape recognition entails collecting adjacent feature segments and merging them together according to partly known object chunks. In general, the parallel shape recognition creates a topological graph of the absolute or relative positions of features and their particular parameters. The graph nodes are labeled by feature names, and arcs represent relations on the features. Thus, the feature names and their parameters are determined and localized through the object shape.

The approach follows the *criterion of simplicity:* a generalization of the Gestalt laws saying that the least information is more likely to be the best [18]. The least information becomes representation of the most regular shapes derived from the picture and assumes the names for the shapes. The least information must be on the symbolic level, because only symbolic information is independent of possible orientations of the pattern in the field of observation [24, 26]. The symbolic representations of features is shown to be invariant in the space of observation. The approach has the property of mapping an image fragment directly into words and phrases. Mathematical calculus for the proposed parallel shape recognition can be based on the concept of the rough sets [24, 12, 29, 30].

Template matching (see [18]) is a time consuming, rotation-dependent and ambiguous. Hong, Shneier, Hartley and Rosenfeld [8] merge in parallel small feature pieces into entities on a pyramid architecture, to speed up the recognition of image edges from matched templates. They exploit good continuation of adjacent window edges to remedy the ambiguity in interpretation of edges matched by single (separate) templates. The proposed parallel shape recognition method is less ambiguous when compared to template matching (because feature names are detected in a deterministic, and not in a statistical way) and when compared to chain coding (because is resistant to variable thickness and discontinuities of objects). Parallel relaxation [17, 13] can improve the algorithms to cope with the problem of discontinuities in the analyzed figures.

Our symbolic and quantitative evaluation of window contents and merging of elementary features into wholes is also a remedy to the problem of ignoring feature continuity by pure clustering methods. For example, the Hough transform [4] in some cases may lose the track of good continuity of edge by treating collinear strips far away from each other as proximate segments. The good continuity is not lost, however, if a separate Hough transform is made for different window contents (extremely helpful when the number of background parts is 1).

Our termination conditions of shape recognition is applicable not only to the 'block world', but also to objects with curved surfaces. The up-to-date experience with the sequential implementation of the shape detection method [24-26] prove the following: there is much less ambiguity in interpretation of elementary and complex features, when compared with template matching. The method is resistant to variable thickness and discontinuities of edges. Is not sensitive to a variety of disturbances. A window is analyzed only once. Particular parameters of features are determined with great precision. Underlying subroutines have already been tested on complicated patterns consisting of lines, arcs, corners and forks, and with very good results [24-26]. The termination conditions has been successfully applied for sequential shape recognition, too.

References

[1] D.H. Ballard, C.M. Brown, *Computer Vision*, Prentice Hall, 1982.

[2] W.T. Beyer, *Recognition of topological invariant by iterative arrays*, Ph.D. thesis, MIT, 1969.

[3] P.R. Cohen, E. Feigenbaum, *The Handbook of Artificial Intelligence*, Addison - Wesley, 1982.

[4] R.O. Duda, P.E. Hart, "Use of the Hough Transformation to Detect lines and Curves in Pictures", *Communication of the ACM*, Vol. 15, No. 1, pp. 11-15, 1972.

[5] R.P. Grimaldi, *Discrete and Combinatorial Mathematics*, Addison-Wesley, 1989.

[6] D.S. Hischberg, A.K. Chandra, D.V. Sarwate, "Computing connected components on parallel computers", *Comm. ACM*, 22, 1979, pp461-464.

[7] T.H. Hong, M. Shneier, "Extracting Compact Objects Using Linked Pyramids", *IEEE Trans. on PAMI*, Vol. PAMI-6, No;.2, March 1984, pp229-237.

[8] T.H. Hong, M. Shneier, R.L. Hartley, A. Rosenfeld, "Using Pyramids to detect Good Continuation", *IEEE Trans. on Systems, Man and Cybernetics*, Vol. SMC-13, No.4, July / August 1983, pp631-635.

[9] K. Hwang, F.A. Briggs, *Computer architecture and parallel processing*, McGraw-Hill, 1984.

[10] G.F. Luger, W.A. Stubblefield, *Artificial Intelligence and the Design of Expert Systems*, Benjamin / Cummings, 1989.

[11] R.S. Michalski, J.G. Carbonell, T.M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann, 1986.

[12] Z. Pawlak, "Rough Classification". *International Journal of Man-Machine Studies*, 20, 1984, pp469-483.

[13] J.M. Prager, "Extracting and Labeling Boundary Segments in Natural Scenes", *IEEE Trans. PAMI*, v.2, No.1, pp16-27, 1980.

[14] E. Rich, *Artificial Intelligence*, McGraw-Hill, 1983.

[15] K.H. Rosen, *Discrete Mathematics*, Random House, 1988.

[16] A. Rosenfeld, "Some pyramid techniques for image segmentation", *NATO ASI series*, Vol.F25, in: *Pyramid Systems for Computer Vision*, Eds. V. Cantoni S. Levialdi, Springer-Verlag, 1986, pp261-271.

[17] A. Rosenfeld, R.A. Hummel, S.W. Zucker, "Scene Labeling by Relaxation Operations", *IEEE Trans. Syst., Man, Cybern.*, v.SMC-6, pp420-433, 1976.

[18] A. Rosenfeld, A.C. Kak, *Digital Picture Processing*, Acad. Press, 1982.

[19] M.O. Shneier, "Extracting Features from Images Using Pyramids", *IEEE Trans. Syst., Man, Cybern.*, v.SMC-12, No.4, pp569-572, 1982.

[20] Q.F. Stout, "Broadcasting in mesh-connected computers", *Proc. 1982 Conf. on Inform. Sci. and Systems*, pp85-90.

[21] Q.F. Stout, "Supporting divide-and conquer algorithms for image processing", *Journal of Parallel and Distributed Computing*, 4, 1987, pp.95-115.

[22] S. Tanimoto, *The Elements of Artificial Intelligence*, Computer Science Press, 1988.

[23] P.H. Winston, *Artificial Intelligence*, Addison-Wesley, 1984.

[24] Z.M. Wojcik, "Rough Approximation of Shapes in Pattern Recognition", *CVGIP*, 40, 228-249, 1987.

[25] Z.M. Wojcik, "A Natural Approach in Image Processing and Pattern Recognition: Rotating Neighborhood Technique, Self-Adapting Threshold, Segmentation and Shape Recognition", *Pattern Recognition*, v.18, no.5, 1985, pp299-326.

[26] Z.M. Wojcik, "An Approach to the Recognition of Contours and Line-Shaped Objects", *CVGIP*, 25, 1984, pp184-204.

[27] Z.M. Wojcik, A. Rosenfeld, "A Shape Coding Array", *Pattern Recognition Letters*, 4, 1986, pp57-59.

[28] Z.M. Wojcik, "A Parallel Shape Coding on SIMD Architecture", *Proc. IEEE Intern. Workshop on Tools for AI*, Fairfax, Oct. 1989.

[29] Z.M. Wojcik, "The Rough Grammar for Parallel Shape Coding, *Proc. ACM South Central Regional Conf.*, Nov. 1989, Tulsa.

[30] Z.M. Wojcik, B.E. Wojcik, "A Rough Grammar for a Linguistic Recognition of Image Patches", *Signal Processing*, 19, 1990, pp119-138.

Fundamental Tasks in Software Development Environments

Lars Bendix
Institute for Electronic Systems
Aalborg University
Fredrik Bajers Vej 7E
DK-9220 Aalborg
Denmark
E-mail: bendix@iesd.auc.dk

Keywords: software development environments, taxonomy, configuration management, version control, personnel management, resource control

Edited by: Johann Eder
Received: December 12, 1992 **Revised:** February 9, 1993 **Accepted:** March 1, 1993

After having established the basic terms of the field of software development, we present a conceptual framework to help establish the key tasks to be performed in this field. The field is characterized by two orthogonal concepts: Programming-in-the-Large (PitL) and Programming-in-the-Many (PitM). These concepts can be further subdivided into the tasks of Configuration Management and Version Control (for PitL), and Personnel Management and Resource Management (for PitM). The main body of this paper is dedicated to a thorough analysis of these tasks. The conceptual framework is then applied to two systems to show how to use it to evaluate and compare systems. Finally is given a discussion of the presented framework, both in its own right and with relation to other work.

1 Introduction

In this paper we will develop a conceptual framework for the tasks which are carried out in a Software Development Environment when developing software. In this context a software development environment will simply mean an environment which supports the development of software. This covers a wide span of quite different systems ranging from early and simple toolkit like systems as UNIX [22] and A Programmer's Workbench [13] to fully integrated systems as Adele [5] and Mjølner [3].

In this field many different problems are encountered such as Configuration Management, Version Control, Personnel Management, and Resource Management. The present software development environments cover some or all of these aspects to a varying degree. Initially the emphasis was on a simple integration of tools for configuration management and version control obtaining a new more powerful tool as done in RCS [26]. Later systems have put more emphasis on obtaining a uniform description and using a data base as the means of integrating more than just the tools for configuration management and version control [16]. They have also attacked the problems arising from the distribution of a project on many different workstations [25]. Or they have tried to solve problems connected to having to deal with coordination of many people, such as security schemas and management policies [18].

However, neither these early attempts nor present systems are complete and cover all of the topics. Furthermore, the lack of a fixed terminology in this field - although one is proposed by Tichy in [27] - and the fact that the systems handle the topics randomly and in isolation tends to obscure a thorough understanding of both the systems and the area. For this reason a taxonomy (a check-list) is needed to remedy the situation and to provide us with a conceptual framework to highlight the differences between software development environments. Such a framework can be useful for:

- **evaluation** of an environment. Classification according to its characteristics. Comparison of properties between different environments. Focusing on central and relevant aspects.
- **development** of a new software development environment. An overview of what it has to include. A guide to the possible choices for each topic. Focusing on central and relevant aspects.
- **understanding** better the field of software development. Focusing on central and relevant aspects.

In the next chapter we give a short, general introduction to software development environments. In chapters 3 and 4 we focus on the tasks carried out within the fields of Programming-in-the-Large (PitL) and Programming-in-the-Many (PitM) respectively. This is followed by a discussion, in chapter 5, of the usefulness of the conceptual framework and its relation to the future development of software development environments. Finally, we draw our conclusions in chapter 6.

2 Software Development Environments

Before we can give a more precise definition of what a software development environment is, we have to agree on some of the basic concepts of the field. A fixed standard terminology for software engineering has not yet emerged. Often different terms are used to cover basically the same ideas, or the same name is used for different ideas. This makes it very difficult and confusing to compare and evaluate different software development environments, and the need of some standard terminology is obvious. Here we will give some informal definitions of the basic concepts concerned and the conceptual framework will serve to give a coherent presentation for the field as a whole.

- **Component.** A component is the basic block used to put together a system. It can be either atomic or composed from other components in an aggregation hierarchy. Samples of components are, among the others, the chapter of a book, a part of a program, the blueprints of a machine, or even an entire program.
- **Version.** A component evolves while time passes. The result of making a change to a component is a new changed copy of the component. This modified component is called a version of the original one.
- **Configuration.** Components are put together to form a system. The term configuration denotes both the actual description of how to build a system and the system itself.
- **Version control.** This is the art of controlling the creation of new versions. The goals of version control are to save space, so that versions can be kept in as little space as possible, and to enforce a structure on the evolution of a component so that such evolution is observable and controllable.
- **Configuration management.** This is the art of controlling the creation and the composition of systems. The goals of configuration management are to save time in the (re)generation of a system, and to enforce a structure on the way a system is built.

In the context of software development environments the meaning of component can be given more precisely. The components involved will be modules, specifications, documentation and other material/documents related to the development and maintenance of software products.

A software development environment is an environment or tool, which supports the development and maintenance of software systems through all of the life-cycle, that is from requirements through coding and testing to documentation and user manuals. It helps to coordinate and manage the plethora of documents in various versions and it serves to integrate the many tools which handle different types of documents and perform different tasks. The most commonly performed and supported tasks in software development have to do with configuration management and version control. Also tools supporting activities for Programming-in-the-Small (PitS), like syntax-directed editors, can be considered part of a software development environment. However, they work on a more fine-grained and less abstract level and will therefore not be treated in this paper. Instead, we will include activities carried out within the field which Dart et al. [11] have coined as Programming-in-the-Many. In their paper it is defined as "tasks such as project and team management". In this paper we will investigate this term in further depth to get a better understanding of exactly what it covers.

Software development environments have undergone an immense evolution over the past 10-15 years. Early environments being primarily conceived as the operating system enhanced with a couple of specialized tools for the handling of configuration management and version control. By means of ordinary text-files the tools were integrated in a very loose way with each other and with the rest of the environment (or operating system).

Later environments saw an emerging notion of some kind of simplistic data base to contain the components and to act as the medium of integration. Components were no longer considered unstructured text-files, but rather black boxes with an internal (hidden) structure for which there could be given an external and uniform description. Furthermore, distribution became a subject of interest.

Present environments have elaborated more on the idea of the data base as the central repository of components. The black boxes have been "opened up" and the emphasis is now more on finding a common representation for their internal structure. Integration is obtained by the sharing of this common description among different tools and by developing an elaborated attribute schema to contain additional information about the components. Finally they introduce the concept of knowledge. There is a growing awareness of the fact that the data base is a repository of knowledge, which can be exploited in an intelligent way to automatically perform certain tasks such as extraction of dependency relations between components.

Future environments could see this development carried on to pure knowledge bases with associated planners to relieve the programmer from tedious tasks. We should also see the arrival of sophisticated query systems so the programmer can be integrated as more than just a source of information. With the arrival and wide diffusion of powerful workstations, concepts like communication and distribution will become increasingly important in order to coordinate the work of the individuals of a project group. Finally we should see a more conscious notion of project management being integrated in the software development environment.

For a more elaborated presentation and discussion of the evolution of especially configuration management and version control in software development environments see [2].

In the following two chapters we present our conceptual framework. At the top level we want to make a division between tasks and activities related to Programming-in-the-Large and tasks and activities related to Programming-in-the-Many. Many other distinctions could have been made such as between the types of environment (language-centred, structure-oriented, tool-kit or method-based) as have been done in [11].

However, we find it much more fruitful to emphasize the distinction between properties related to the manyfold of components and versions (PitL), and properties related to the manyfold of persons and storage locations (PitM). The former has to do with the management and control of systems and are mainly problems related to single programmers. The latter has to do with the organization and management of people and ressources of a project.

Of course this distinction cannot be clear-cut as the different tasks that have to be performed to some degree are interrelated (like a configuration of a system including modules produced by others and located on different servers). However, the advantage of our approach is that the focus is put on the more abstract level of distinguishing the different tasks and activities that have to be supported by a complete software development environment for it to constitute the ideal environment for supporting the development of software. In the construction of the framework we thus take the programmer's perspective - not his perception of the environment he uses, but rather his perception of the tasks and activities he has to perform to develop and maintain a large software system (build systems, save versions (PitL) - communicate, adhere to methods, cooperate (PitM)).

3 Programming-in-the-Large

The original definition of PitL as given by DeRemer and Kron [12] is that it has to do with:

- structuring a large collection of modules to form a system
- describing module interconnectivity
- enforcing module disconnectivity

Based on this definition they give the proposal for a separate Module Interconnection Language to solve these problems. Much work har been done to get a better understanding of the proper mechanisms for Module Interconnection Languages. For a survey of this research see [20]. Results from this research have migrated into programming languages proper, most notably Ada [1] and Modula-2 [28], to the extent that modern programming languages have little or no need for a Module Interconnection Language on top of them. Both for this reason and because it gives more flexibility in the configuration management task [7] the latter two of the above problems can today be considered as belonging to the world of PitS. This leaves primarily the problem of structuring the modules to form consistent systems, which is basically what Configuration Management is about.

Another problem which we consider to belong to PitL, but which is not treated in the early paper by DeRemer and Kron, is that of versioning. Especially for todays systems which have to be maintained and further developed for long periods of time and possibly for different hardware and operating systems, there develops a manifold of versions of each module. These versions are highly related to each other and yet they may exhibit quite different characteristics. The safe and easy administration of these versions is what is handled by Version Control.

3.1 Configuration Management

During the development and maintenance of a large and complex software system, it is often necessary to (re)compose it in different ways. Both because its components evolve and because there may be the need of configuring the system differently for different hardware or different operating systems or different clients.

Configuration management is about the composition of systems from the multiplicity of possible modules and the two main objectives are to save time and to add structure. Saving time concerns minimizing the cost of (re)composing a system, as well in terms of CPU time as in terms of human and elapsed time. Adding structure serves to introduce additional constraints and abstractions, thereby making system composition more flexible and controllable.

In the process of creating a software system three different steps can be distinguished: description of its structure, selection of the involved components, and instantiation of the description using the selected components.

The description can be considered a generic template which gives the overall structure of the system without stating explicitly the contents of this structure. The next step is to transform this generic structure to a specific one by resolving - whenever necessary - which component to use for each "slot" of the structure. Finally, because we deal with software systems, we have to apply the usual compile-link-load cycle to the specific structure to obtain an executable system.

3.1.1 Description of systems

The first thing we have to talk about for configuration management is how to describe the structure of the systems we want to generate. A description of a configuration is the formal representation of the programmer's knowledge about how to compose a system. It may serve many purposes such as consistency checking, type checking, information hiding etc. However, here we will just consider the configuration management aspect where it is used to make information available for the configuration management system such that the task of generating systems can be automated.

There is a wide range of different ways to describe systems spanning from verbose scriptlike ways to almost automatic systems. The extreme would be a system where the command "build editor" would be enough to generate the editor. However, this requires that the configuration management system is in possession of perfect knowledge - a situation which we still have to arrive at.

Early systems like Make [14] are characterized by rather verbose centralized descriptions.

These descriptions contains both structural information about the dependencies between modules and operational information about how to compile, link and load the modules together. Some systems support ways of stating default derivations (like how to derive a C object code module from its source code) to make descriptions less verbose in the operational part. The advantages of these systems are that automating the system generation makes it possible to avoid errors and omissions, and that the system detects when updates are superfluous. The systems are very general in the way they work and their usage is in fact not restricted to software configuration. However, the generality has the drawback that there is little or no information available to support e. g. consistency checking of configurations. By far the most serious drawback, though, is that the dependency relations between the modules are considered a global property [7]. This implies a rather static and inflexible view of configurations, which are not in line with many of the actually used development methods such as prototyping.

As stated above, the introduction of module interconnection languages had more that just configuration management purposes. Being primarily a means for providing the interface part known from modern modular programming languages, they usually contain only structural information and no operational information. However, as they are normally used in monolingual environments, the knowledge can be hard-coded into special configuration management tools. Unfortunately the usage primarily in a monolingual setting obscures one of the important objectives of a module interconnection language, namely that of permitting the safe composition of a system from multilingual components. In a configuration management perspective the big step forward is that of considering the dependency relations between components a local property. Moving away from the centralized description, they give locally for every (version of a) module an external description of important aspects of its contents. This opens up for a more dynamic and flexible view of a configuration where the relevant information is placed at the source of its origin and therefore is more easily maintainable.

In some recent systems (Adele [6] and DSEE [17]), the construction of the configuration description has been completely automated as regards the structural part. This has been possible because they support languages in which there are primitives for describing dependency relationship (like statements), which can be automatically extracted and used by the configuration management system. In fact Adele requires that supported languages have a notion of interface- and implementation-part. This has the consequence that the programmer has only to worry about dependencies in the context of programming the single module. Furthermore, the systems contains knowledge about how to use standard conventions (like extensions to filenames) to deduct the proper description of object-modules and executable systems. We are thus close to the ideal situation where the structural information is automatically extracted from inside the modules, and the operational information is present as a combination of a knowledgeable tool and a more formal typing notion of the modules.

So in summary for description of configurations, we have to consider if:

- it is done manually or automatically
- it has a global or a local view
- it contains only system structure or also build instructions
- it is explicitly (such as text) or implicitly (such as graphics) represented

3.1.2 Selection of components

Given the structural description of a configuration, the next step is to "populate" it with components. Except for very simple systems, the description has to be considered a template for which we have to find the proper components to "inhabit" every slot. The reason for the existence of many potential candidates for every slot is versioning in a broad sense. Versioning because of bug-fixes, because of further development of a component, because of adaptations for different hardware or clients, because of different implementations of time/space trade-offs, etc.

There are conceptually seen basically two different kinds of versioning/selection for which we need appropriate ways of resolving which components to choose.

The first is versioning because of continuous development of a component. For this kind of versioning we would like to be able to make selections based on some sort of state (such as experimental, tested, released) or based on temporal properties. These selections can be done either dynamically ("latest tested version") or statically ("version 3.1"). Care has to be exercised when using dynamic selection as a later selection using the same criteria will not necessarily yield the same result. Also care has to be taken to distinguish dynamic selection from static selection. For instance is a statement like "latest released version" dynamic in nature, as a tested version may transform into a released version, whereas a statement like "before todays date" is static.

The second kind of versioning arises from the existence of variants and the reuse of the same component in differing contexts. Here the need is not so much of selection on state or temporal properties as that of selection on (user-defined) functional properties and the existence of choice propagation. If we are building a customized system for Acme Inc., we want to be able to state this as a selection rule. Furthermore we want (at least some of) the selection rules to be propagated into the further selection of components on which the present selection depends.

The general problems encountered in selection is those of how the set-up is and how the selection mechanism works. We need to consider whether the set of attributes available is pre-defined or if it is possible to have user-defined attributes, too. Furthermore, if the set of values of the pre-defined attributes is extendible or not. Most systems introduce some sort of parallel to the scope-rules of programming languages. They have a hierarchy of importance where a component in the users workspace can have precedence over the project version of that component. Other systems use a schema of standard/default indications for the same purpose.

This leads to another point of great importance, namely that of ambiguous or empty selection sets. From the existing configuration management systems there seems to be no clear line in how this problem is solved. In the presence of an empty selection it is obvious to signal an error, but when does a selection become empty. Will for instance a selection-rule stating "hardware=SUN" become empty if no components have the attribute "hardware" and it is set to "SUN". Or will the absence of the attribute be interpreted as the independence of hardware, so the component will be selectable? The same kind of problems arise when the selection-rules are not exhaustive, thus leaving two or more components to choose from with no explicit selection-rule to do this choice. Also here there is a wide variety of solutions. Some use a schema of choosing the newest component possible. Others rely on standard/default indications of preference or some other preset strategy of scope-rules. And yet others use a selection schema introducing a hierarchy in the strength of the selections ranging over imperative, exclusive, conditional, and default selections.

In summary when considering the selection of components to populate the given description, we have to pay attention to:

- if choices are static or dynamic
- if there is propagation of selection-rules
- if the attribute schema is extendible
- the existence of scope-rules
- the resolution of empty/ambiguous selection sets

3.1.3 Instantiation of configurations

The third step in generating a software system is that of instantiating the configuration. In the two steps we have described the system and chosen the components respectively. Now we have to assemble the specified system and dealing with software this is done by compiling the modules and linking and loading them together into an executable system.

The degree to which the programmer is aware of the presence of the object-code varies much. In systems like DSEE, everything is specified in terms of source-code modules and the binaries are considered internal results of system application of tools to source-code. Other systems like Make have the programmer worry about specifying each and every detail in transforming the source-code modules into an executable load-image.

The question of when the instantiation of a new configuration is executed and has effect is another key point. Most systems implement a schema of lazy effectuation - a schema where systems are only generated by explicit request. This has the advantage that the programmer has full control over when new systems are generated and can be aware of the effects of a dynamic selection. However, it also means rather long idle-times for the programmer if a lot of work has to be done to (re)build the system. For this reason some systems implement a busy effectuation schema trying to anticipate the requests. This gives very fast response times (sometimes even zero) for the instantiation of configurations. The drawback, however, is that a lot of work (CPU-time) can be consumed on instantiating configurations that are never requested.

If an entire configuration had to be (re)instantiated every time even a small change was made to some module, it would lead to disastrous performance problems. All systems implement some optimalization mechanisms to handle this problem. The best known technique is to compute the least update necessary to keep the configuration consistent. This causes a compilation only of modules for which no binary exist, and a recompilation only of modules that have been changed (or that have had effective changes) and modules that depend on these. Another approach is to keep pools of binaries to avoid compilation because of missing binaries. For large and fast evolving systems this can lead to serious space problems, which can, however, be solved (partially) by using paging-like algorithms for discarding binaries unlikely to be used. Finally the arrival of workstations supporting multiple processes and connected in networks have lead to new ways of obtaining performance gains. In these systems the work of instantiating configurations can be performed as background jobs making use of the computers idle-time or of excess CPU-time on other computers in a network.

Summarizing the key points to pay attention to for the instantiation of configurations, we have the following:

- orientation (source- or object-code)
- effectuation (lazy or busy)
- optimalizations (least update, pools, distributed or background processing)

3.2 Version Control

During its development and maintenance a component will change slightly. For many reasons it is convenient to keep, or to be able to regenerate a component also in its intermediate forms and not only in its present one. In other cases there may be distinct but related components which we want to consider as variants of each other and belonging to the same "family".

Version Control aims at managing the history of development of a component and the two main objectives are to save space and to add structure. Saving space is mostly a technical matter, in which all the versions are compressed in as little space as possible. More important, however, is it that version control makes it easier and more convenient to understand and record the evolution of a component and the interrelations between the manifold of versions.

When trying to manage and maintain a collection of versions and variants of components, there are three important notions: the underlying model for the evolution, the employed attribute schema for the components, and finally how to merge components.

The underlying model is an important instrument to serve as a conceptual framework for structuring and restricting the way components can evolve. The attributes associated with the components are useful in the context of version control for distinguishing the different versions, but also in a larger context where they serve as an external description of some internal (private) properties of the components. In the context of parallel development (variants), merging is an often performed task to incorporate corrections/changes to one variant into other variants.

3.2.1 Model

One of the most important instruments when trying to manage the manifold of versions is to have a good model. The existence of such a model makes it possible to create abstractions to suppress irrelevant details and to support a superimposed structure.

The primary way of supporting abstractions is to consider the set of related versions and variants as one entity. Usually such an entity is called a family, and the members share some important characteristics (such as functionality) while others (such as implementations) may be different. Ideally the members of a family should be interchangeable and only when special characteristics are needed (like performance, special hardware, bug-fixes) it will be necessary to "open up" the family and choose a specific member.

Inside a family it is necessary to put a structure on the way its members can evolve from each other. Such a structure is a help to guide the evolution in a sound direction and to enforce restrictions to avoid unwanted or unsafe development. Furthermore the structure helps in providing a conceptual framework to make evident the history of previous development. As such the model serves to understand previous development and to guide future development.

The most widely used structure is that of a tree. Development is basically considered to be linear, but in some points it can branch into a parallel development. The exact nature of the tree (one/more branches, branches on branches, etc.) differ from system to system, although there seems to be a general agreement on allowing only one level of branching.

There is, however, one big problem with the tree-model, and that is the treatment of variants. Variants have a nature slightly different from that of versions. Variants are developed in parallel lines which for some aspects (like supported hardware) are independent from each other, but for other aspects (like inclusion of common bug-fixes) they are interrelated. This indicates the need of being able to state relations between different branches of development. This model can be realized by a graph. However, a general graph is inconvenient from the point of view of controlling the evolution, as it is too permissive. Thus restrictions (such as acyclic) are usually put on the nature of the graph. The graph also supports the situation where diverging lines of evolution have to be merged into one common (see section 3.2.3), which is not a very unusual situation in software development.

In some cases the distinction between variants and revisions can be important for the organization of the whole project. In such circumstances we should look for mechanisms to support a model where there is orthogonality between variants and revisions not just on the level of individual components but on a project-wide basis. There are different approaches to this [21] [8], but they have all in common that their organization emphasizes the entire project over its individual components. Consequently, the terms variant and revision in this case span the whole project and are orthogonal to each other.

In summary we have to look out for the following characteristics of the model for Version Control:

- the support for abstraction
- the support for structure (tree, graph, orthogonality, restrictions)
- how variants are treated

3.2.2 Attribute schema

For version control and other purposes an attribute schema is a necessity. For structural ends it serves to unveil the abstractions made in the model and to give an extra layer of relations between components not present in the model. Informative ends see the attributes employed in providing a way to state important history and descriptive information connected with each component.

The model for version control helps in managing the manifold of versions by providing abstractions to compact the abundance of information into a manageable number of uniform units. The attributes help us to go the other way. To uncover the diversity of the components in a unit and to help in "naming" or selecting amongst them. They thus work to support and make manifest the model (such as by version numbering), but they also add to the structural model. In the absence of relations, the attributes can be used to supply an extra structure to say how the components in a unit interrelate and possibly relate to other units.

An important aim of version control is to keep information other than the structure to help understand the historical development of the components. Again the attributes are able to store the needed information in a compact and concise way, so it can be used both for querying and for selecting purposes. In general, attributes are convenient for giving short, uniform information about heterogeneous components. It is a way to give a uniform external description of properties (like used language) of the component that might not even be deducible from the contents of the component (like state).

Summarizing the treatment of attributes, the important points are:

– support for structural purposes

– support for informative purposes

3.2.3 Merging of components

The merging of different versions or variants is a very important and very often performed task in the development and maintenance of software systems. Conceptually all the components of one family follow one single line of development. There may be divergences for some periods of time, but the general flow of development is common. For versions branching off into parallel development, it is usually a temporary solution and one may soon want to integrate the changes into the main line of development. Variants are more long-lived, but in general there will be the need of applying the changes made to one line of development (like bug-fixes) also to some or all of the other lines.

The support for merging is not present in all systems. Some systems consider it to be outside of the scope of the environment and to be solvable by means of other tools. For other systems (mainly non-text based) the problem is that the merging of two abstract syntax trees is not a well understood and easily solved problem. For this reason they have concentrated their efforts on other topics and leave the programmer with no support - if not to extract text-copies and to re-enter the merged copy into the environment.

Many (text-based) systems, however, provide an integrated mechanism for merging. The advantage of doing the merge inside the environment is that it is possible to automatically record the fact that a component is the result of a merge of two other components. If it is done outside the control of the environment, the programmer has to supply the necessary information if there is any support at all even for recording this type of information.

The point to consider for merging of components is if it is an internally supported operation, or if it has to be performed outside of the control and support of the environment.

4 Programming-in-the-Many

In this chapter we will look at the problems caused by the big organization around software development. Programming-in-the-Many is aimed at how to manage many people having different tasks, working on different computers, using different tools to produce slightly different products. As projects get bigger and live longer, it becomes increasingly more difficult to contain these many people and the many physical resources within the context of one single project. We subdivide this topic into how to manage the many persons involved and how to organize the many resources at disposal.

Personnel Management is about how to systemize and formalize the interrelations between the many different persons attached to a project. To compute their tasks successfully they have to collaborate, coordinate and communicate. Often they have to adhere to some kind of formal development method and there may be pre-established patterns of communication such as for the handling of bug-reports. Furthermore, we need to protect the various parts of the project from unauthorized access.

The growth in easily accessible resources has made it more and more common that each programmer has his own personal workstation. Furthermore, big and long-lived systems are often implemented on several different target computers. How to create order out of this chaos of often non-homogeneous and distributed resources is what Resource Management is dealing with. We need to give a uniform view of the resources and we need to make the distribution of them transparent.

4.1 Personnel Management

As the size of software development projects grow, it becomes necessary to involve more people and to specialize them. On big projects there can be assigned several hundred persons, and during the life-time of the project (including maintenance) even more people may contribute to the project in a shorter or longer period of time. With todays big multi-national companies it is not unusual to see a project also being distributed in place even to different continents.

This causes severe problems in coordinating these people in a disciplined way. The problems can be approached in two ways: a formalized method of development to assure the documented collaboration and communication; and the implementation of security measures to protect against access violations. In short, we want to organize the interpersonal relations in a formalized framework, where they can be documented and controlled in order to be able to successfully manage software development projects.

4.1.1 Method

Organizing big projects is a difficult and complex task. We have to assure that all parts of the project is being taken care of, and that no part of it is unintentionally being worked on by several persons. For this reason we need methods to break down the project into smaller pieces and ways to assign these pieces to certain persons or group of persons (who again might break it down into even smaller pieces).

In the context of software development several formal methods have been defined. They range from rather rigid and simplistic ones like the waterfall model [23] to more anarchistic and fuzzy ones like rapid prototyping [15]. Some methods are aimed at a specific programming paradigm and can best be used under this restriction [10], while others are more generally applicable no matter which programming paradigm is used [9]. However, they all share the distinction between several different phases of the development like: requirements, specification, coding, testing and so on. The main difference between the methods is their approach to the execution of these phases. In for instance the waterfall model they have to be executed in a strictly sequential order one after the other. Others consider it to be a cycle that has to be iterated - maybe indefinitely - and where the phases may be carried out in parallel. In yet others the phases interact in arbitrary ways to lead towards the common goal - the right system.

One thing on which all the methods agree is the importance of communication between the different phases - or the exchange of information. And that this communication has to be documented and persistent. Documentation is usually obtained by using standard formulas in exchanging information such as specification schemas and bug reports. This has the advantage of keeping a high degree of written information exchange. Written information is easily made persistent and the (semi-)formal way in which it is stated makes it easy to construct query-systems to facilitate the fast retrieval of any wanted piece of information. A high degree of written and persistent information gives a good possibility of controlling the project.

One point on which the methods differ is the extent to which they incorporate maintenance as an integral part of their model for the development. Some methods do not consider maintenance at all and we have to fake it considering a sequence of projects where the requirements for the new project is that for the old one plus bug corrections plus enhancements/changes. Other methods have maintenance as an integrated part of the model; some methods have it as a transition to a special maintenance phase, while others has it as a normal phase of the continuous development of a system. The maintenance phase has become increasingly important as it can consume a major part of the resources of todays software development projects [24].

All the methods are useful for controlling software development. However, as the computer is also the main tool for all the people working on the project, then we get the maximum benefit when the method is also supported by the computer. The ideal would be that all information exchange was done by means of the computer media itself. In this way all the information about the project would be on-line and easily combinable in different ways in different contexts. Furthermore, because of its storage capacity and computing power, the computer is also the perfect tool to impose the methods and to control that they are being followed.

To sum up, we must look for:

- support for different methods (work breakdown)
- support for communication/documentation
- support for maintenance

4.1.2 Security

People working together on big projects need to collaborate and while collaborating also to share things among them, such as the modules of the system. However, not all information and all things should be shared between everybody - and not in the same way. For example, while a programmer works on a new version of a module, he wants that version to be private to himself so nobody else can use it by accident or make changes to it without his knowledge. When he has finished the version, he might make it accessible for the people responsible for the testing in a read/execute mode. When these in turn have finished the testing, they make it accessible to all the other programmers in execute mode - other people like the release people will not have access to the components of the system, but only to the compound system itself.

When talking about access problems we have to consider that there are two types of illegal accesses. There are illegal accesses done with an ill-will intention and illegal accesses done by accident such as by accessing a wrong version. Within the project most illegal accesses will be of the second type, while the first type will be most common for people from outside the project - but maybe still within the same company. However, in some very special cases - e. g. when a programmer is away for a longer period - it can be necessary to circumvent the security system to avoid waiting for the accessibility of some important information.

The most common ways to fight illegal accesses are either to put access rights on all single files or to partition the project information and assign rights to each. When we talk about putting access rights on each single file, this can be done in many different ways. Usually there exists a way to group other users into precisely defined groups for which we can grant different rights. This can be done as in UNIX-based systems where the division is between user, group and others, or it can be done as in other systems where it is possible to specify for each file a list of users which are allowed to access it. The kind of access granted can vary from a pure execute permission to an all-out write permission. The other way to grant permissions is to partition all the projects files into more or less related partitions and for each one assign access permissions to different groups of people. Probably a mixture of the two approaches will be the most efficient way of implementing security schemes, even if it may imply compatibility problems between permissions granted under each single scheme.

Rounding up, we can say that security measures fall into two groups:

- security against illegal access.

- possibilities of partitioning the project.

4.2 Resource Management

The introduction of the personal computer - and thereby the easy access to computing resources - has had an immense impact on the way in which software development is conducted. Now people are not any more connected to a central mainframe via terminals, but are using workstations interconnected in local networks. Furthermore, the developed systems tend to get bigger and more costly. To amortize the high costs, such systems tend to have longer lifetime and be implementated (and maintained) on many different computers. For these reasons we see a growing trend towards a distribution of the development (and especially the maintenance) efforts in space and time.

The distribution in space is a result of splitting up the project on many computers. This division has many advantages in the form of increased computing power, storage capacity, flexibility and so on. However, it also poses us the task of making the distribution transparent. Otherwise the drawbacks in form of overhead in the manual management of the partitioning will outweigh the advantages.

We can consider the problems about maintaining (and developing) many variants of the same system as a question of distribution in time. We have to manage many systems created at different points in time and maybe having slightly different characteristics. This creates problems of many different kinds. We have to be able to regenerate old systems to handle bug correction. We have handle updating the common part of different variants. We need to know exactly what was shipped with each and every release (version of software, manuals and documentation). We must keep track of which clients have what system. And the list can be continued.

4.2.1 Workspaces

The workspace is a way to solve the problem about the spatial distribution of a project. When the information (program, documentation, test) is divided on several computers, there exist the task of assisting the programmer in obtaining/maintaining easy access to it. The ideal would be to make the distribution transparent so that he does not have to care about e. g. network protocols. And to give him a uniform view of the project such that he does not have to fight idiosyncrasies of different operating systems, environments, or - more generally - applications.

The requirement to the programmers environment should be that he in his normal work does not have to deal with the problems caused by distribution. It should be possible for him to make scripts (or to get ready-made ones) handling the differences in interacting with different systems. And he should be able to define a profile telling the physical position of the files he is using but which is not residing on his own file server.

Furthermore, we must have some mechanisms to handle the dangers of sharing the same files. The updating of a module without the knowledge of the programmer who is responsible for it can cause very unpleasant surprises. Mechanisms that can notify us in case the module we are working on also being updated by someone else. This could be to tell us about the existence of a never version than the one we used to work on. Or that somebody is trying to update the version we are currently working on. Or that somebody is already working on the version we want to work on, such that there might be a potential conflicting update in the future.

4.2.2 Time-machines

In the science fiction literature time-machines are used to take the reader on a travel in time. Either to go back in time to see exactly how some historic event happened, or into the future to give him a glimpse of how some actions can effect the future. This is much the same functionality we want to obtain from the time- machines when they are applied to software development. we want to be able to go back in time and locate the components of a specific system and regenerate an exact copy of it. Likewise we would like to make some changes, see how they influence on the systems (future) behaviour and go back and undo some actions creating undesired effects.

Maintenance of old variants is the main reason for having to go back in time. When a client calls us about a bug, we must be able to generate an exact copy of his system to be able to understand the nature of its erroneous behaviour. And when we have corrected the bug - or in other ways want to update the system - we need to know how the system is affected. It is especially important that the time-machine takes into consideration more than just the modules of the system. Albeit the most important part, we also need to be able to find the corresponding specifications, tests, documentation and so on, so also these can be consulted and updated in a correct way.

The further development of a system can also benefit from the time-machines ability to go forth and back in time. When we have to make changes to a system, we can make experiments to see the effects, and if some of the effects are undesired we have the possibility to go back in time and undo them (much the same as in an editor). This can be used when we have to make a common change to different variants when we cannot do it in the common part (or to discover that it has side-effects for some variants). The facility can also be used to make exploratory programming where we by successive trail-and-error arrive at the desired result.

However, the most important property of the time-machine is that it makes our experiments documented. And that it lets us work in precisely defined and consistent contexts.

5 Discussion

When presenting a conceptual framework for software development environments, many different approaches can be taken. One such is the one of [11] where they group a collection of systems into different categories having different characteristics and properties. This approach has in our opinion the serious drawback of only focusing on the technical topics which are captured by existing (or by their collection of) systems. As a classification of existing systems it does well. The problem is that such an approach only allows us to construct a framework which spans the known and already implemented. When we want to make a proper framework for the field of software development as such, we find that it is crucial for its generality and completeness to start from a dissection of which tasks have to be performed in this field. Such a study could take its starting point in [19], which is a presentation of the different maturity levels in the organization of the software development process. In this way we can capture also properties which are as yet poorly or not at all supported. That is why we focus on the functionality that has to be offered in the area of software development rather than giving a technical presentation of existing systems.

There may be raised objections to our classification of topics, saying that we have left out things like PitS, integration, and user interface. It is true that these are important topics and that we do not treat them independently. Such topics must also be a part of a complete evaluation of a software development environment. Our focus in this paper, however, is only on the tasks which have to be carried out and which should therefore be properly supported. We think that PitS belongs at the level of individual tools, which we do not consider in this paper. Integration is a "behind-the-screen" implementation issue, which is of no concern to the user - except when it "shows up on the surface". User interface is too diffuse a topic to be treated in its own right as it affects most of the other topics - especially workspaces - and are treated there. These are all important topics, but they should be treated separately in another setting as they are outside the scope of this paper.

A more serious objection is that there is no separate treatment of what is the conception of the basic unit. That there is not elaborated on what is the view of the component (internal/external description, common representation etc.). Again our point of view is that this is not the proper place for any detailed treatment of that topic. It is something which belongs mainly to the programming language field. It has obvious relations to some of the tasks in software development and is duely treated under these tasks. We want, however, to keep our framework language-independent and thus in the context of software development environments the treatment given in chapter 2 suffices from the point of view of the user.

Our framework is very suitable for evaluating and comparing software development environments. It distinguishes a number of important, orthogonal topics and for each topic it lists a number of parameters which have to be considered. This provides a good foundation for evaluating how suitable a specific software development environment is for a given organization. In the presented framework, all the topics are stated on an equal footing. In order to use the framework in an actual evaluation the relative importance of each topic has to be adjusted to the requirements of the given organization. In such a customization some of the listed topics may be left out because they are not relevant in the given context. The use of our framework ensures, however, that we start with a complete list of relevant topics.

The most important innovation of this conceptual framework is the attempt to establish PitM as a dual to PitL and equally as important. Much more than just a question of management, PitM also has to deal with things like communication, cooperation, and homogeneous views of resources. As pointed out by Bazelmans [4] the lack of a well defined method for project use can be a hindrance for the widespread and coordinated use of many useful tools. This framework will hopefully help to avoid this.

It is always difficult to guarantee the completeness of a framework of this nature. It will inadvertently be limited by the depth of our insight into and understanding of the field in question. The part of our framework which covers PitL builds on other fields with a long and rich background in research. This provides us with a solid foundation and makes it possible to refer to actual implementations of the discussed concepts. The other part which covers PitM is potentially more problematic because there is not much prior experience to base upon. We draw on work from many different fields like capability maturity models, software process modelling, and computer supported cooperative work. The relevant parts from these fields are brought together under the term of PitM and then categorized in a systematic way such that it is suitable as a framework. The field is still young, so this part of our framework should not be considered to constitute the final word. It is, however, complete with respect to the present level of knowledge.

6 Conclusions

In this paper we have presented a conceptual framework to use for the study of software development environments. The field is divided into two main topics, Programming-in-the-Large and Programming-in-the-Many, which are again divided up into subtopics, Configuration Management, Version Control and Personnel Management, Resource Mànagement respectively. Each topic is analyzed in details both to give a better understanding of the tasks which have to be performed within each subtopic, and to present some of the alternative ways to solve them.

The innovative in our approach with respect to other surveys of the field, is that we apply the perspective of the user of the environment and the tasks he has to perform to solve his assignment. Other taxonomies take the perspective of the environment and are therefore neglective to tasks which are poorly or not at all supported in todays environments.

Finally, we show the importance of such a framework. On the theoretical level it can be used as an introduction to the field of software development helping us to focus on important topics and to systemize our insight. On the more practical level it can be used both to evaluate and to construct environments.

Acknowledgement: This research was supported, in part, by grant no. 9400911 from the Danish Natural Science Research Council.

References

[1] ADA Reference Manual, Proposed Standard Document, United States Department of Defense, July 1980.

[2] Vincenzo Ambriola, Lars Bendix, and Paolo Ciancarini: The Evolution of Configuration Management and Version Control, Software Engineering Journal, Volume 5, Number 6, November 1990.

[3] Lars Bak, Claus Nørgaard, Elmer Sandvad, Jørgen Knudsen, and Ole Madsen: An Overview of the Mjølner BETA System, in Software Engineering Environments, Volume 3, Ellis Horwood Limited, 1991.

[4] Rudy Bazelmans: Evolution of Configuration Management, ACM SIGSOFT Software Engineering Notes, Vol. 10, No. 5, October 1985.

[5] N. Belkhatir, and J. Estublier: Experiences with a Data Base of Programs, ACM SIGPLAN Notices, Vol. 22, No. 1, January 1987.

[6] N. Belkhatir, J. Estublier, and W. L. Melo: Adele2: A Support to Large Software Development Process, in proceedings of the International Conference on the Software Process, Redondo Beach, CA, October 1991.

[7] Lars Bendix: Configuration Management and Version Control Revisited, PhD dissertation, Institute for Electronic Systems, Aalborg University, Denmark, October 1995.

[8] Brian Berliner: CVS II: Parallelizing Software Development, in proceedings of the USENIX Winter 1990 Conference, Washington D.C.

[9] Barry W. Boehm: A Spiral Model of Software Development and Enhancement, IEEE Computer, May 1988.

[10] G. Booch: Object-Oriented Design with Applications, Benjamin/Cummings, 1991.

[11] Susan A. Dart, Robert J. Ellison, Peter H. Feiler, and A. Nico Habermann: Software Development Environments, IEEE Computer, November 1987.

[12] Frank DeRemer, and Hans H. Kron: Programming-in-the-Large versus Programming-in-the-Small, IEEE TSE, Vol. SE-2, No. 2, June 1976.

[13] T. A. Dolotta, R. C. Haight, and J. R. Mashey: The UNIX Time-Sharing System: The Programmer's Workbench, The Bell System Journal, 57 (6), July-August 1978.

[14] S. I. Feldman: Make - A Program for Maintaining Computer Programs, Software - Practice and Experience, Vol. 9, 1979, p. 255-265.

[15] Christiane Floyd: A Systematic Look at Prototyping, in Budde Kuhlenkamp, Lars Mathiassen, and Heinz Zllighoven, editors, Approaches to Prototyping, Springer Verlag, 1984.

[16] A. Nico Habermann, and David Notkin: Gandalf: Software Development Environments, IEEE TSE, Vol. SE-12, No. 12, December 1986.

[17] David Lubkin: Heterogeneous Configuration Management with DSEE, in proceedings of the 3rd International Workshop on Software Configuration Management, Trondheim, Norway, June 1991.

[18] N. Minsky, and A. Borgida: The DARWIN Software-Evolution Environment, ACM SIGPLAN Notices, Vol. 19, No. 5, May 1984.

[19] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber: Capability Maturity Model for Software, Version 1.1, Technical Report CMU/SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, February 1993.

[20] Ruben Prieto-Diaz, and James M. Neighbors: Module Interconnection Languages, The Journal of Systems and Software, No. 6, 1986.

[21] Christoph Reichenberger: Voodoo: A Tool for Orthogonal Version Management, in proceedings of the Fourth International Workshop on Software Configuration Management, Baltimore, MA, May 1993.

[22] Dennis M. Ritchie, and Ken Thompson: The UNIX Time-Sharing System, Communications of the ACM, Vol. 17, No. 7, July 1974.

[23] Winston W. Royce: Managing the Development of Large Software Systems, IEEE WESCON, August 1970.

[24] Ian Sommerville: Software Engineering, 4th edition, Addison-Wesley Publishing Company, 1992.

[25] D. C. Swinehart, P. T. Zellweger, R. J. Beach, and R. B. Hagmann: A Structural View of the Cedar Programming Environment, ACM TOPLAS, Vol.8, No. 4, October 1986.

[26] Walter F. Tichy: RCS - A System for Version Control, Software - Practice and Experience, Vol. 15(7), July 1985.

[27] Walter F. Tichy: Tools for Software Configuration Management, in proceedings of the ACM Workshop on Software Version and Configuration Control, Grassau, Germany, January 1988.

[28] N. Wirth: Programming in Modula-2, 3rd edition, Springer Verlag, 1985.

A Sound and Complete Axiomatization of Functional Dependencies: A Formal System With Onty Two Inference Rules

Mirko Maleković
University of Zagreb
Faculty of Organization and Informatics
Pavlinska 2, 42000 Varaždin, Croatia

Keywords: completeness, formal systems, functional dependencies, inference rules, soundness

Edited by: Rudolf Murn
Received: December 12, 1994 **Revised:** August 9, 1995 **Accepted:** September 26, 1995

A formal system for functional dependencies is introduced. The formal system contains only two inference rules: (I) Reflexivity and (II) Generalized transitivity.

1 Introduction

Various sound and complete formal systems for functional dependencies have been studied in the literature ([Armstrong 74], [Beeri et al. 77], [Maier 83], [Pichet and Delobel 79], [Saxena and Tripathi 89], and [Ullman 88]. All the formal systems, as mentioned above, have at least three inference rules. In this paper, we present a new formal system for functional dependencies; the system has only two inference rules: reflexivity and generalized transitivity.

The paper is organized as follows. In the next section, we describe a new formal system for functional dependencies with only two inference rules. Correctness of the formal system is also proved. Conclusions are given in Section 3.

2 A Formal System with Two Inference Rules

In this section we describe a formal system which contains two inference rules. The system is called FS_2-system . FS_2-system for functional dependencies consists of two inference rules:

(I) $\vdash X \rightarrow Y$ if $Y \subseteq X$
(reflexivity)

(II) $X \rightarrow Y, YZ \rightarrow W \vdash XZ \rightarrow W$
(generalized transitivity)

In the following proposition we assert that FS_2-system is sound and complete.

Proposition 1 *FS_2-system is sound and complete.*

Proof

Because it is very easy to prove that FS_2 is sound, we only prove that FS_2 is complete.

We shall first introduce an entailment relation, denoted $\succ$. Let FS be a formal system for functional dependencies and $IR : f_1, .., f_n \vdash g$ be an inference rule for functional dependencies. We shall say that FS entails IR, denoted $FS \succ IR$, if $f_1, .., f_n \vdash_{FS} g$, where $f_1, \dots, f_n \vdash_{FS} g$ indicates that g is derivable from $f_1, \dots, f_n$ in FS.

In addition, let FS_1 and FS_2 be two formal systems for functional dependencies. We shall say that FS_1 entails FS_2, denoted $FS_1 \succ FS_2$, if $FS_1 \succ IR$, for each inference rule IR in FS_2.

We have the following auxiliary proposition.

Proposition 2 (Completeness and Relation $\succ$) *Let FS_1 and FS_2 be two formal systems for functional dependencies. FS_1 is complete for functional dependencies if FS_2 is complete for functional dependencies and $FS_1 \succ FS_2$.*

Proof

The proof follows immediately from the definitions of completeness and relation $\succ$.

It is well-known that Armstrong's formal system, denoted AS, is complete for functional dependencies. AS has the following inference rules:

$as_1: \vdash X \rightarrow Y \quad \text{if} \quad Y \subseteq X$
(reflexivity)

$as_2: X \rightarrow Y \vdash XZ \rightarrow YZ$
(augmentation)

$as_3: X \rightarrow Y, Y \rightarrow Z \vdash X \rightarrow Z$
(transitivity)

Now we shall prove that $FS2 \succ AS$.

Because (I) is as_1, we have to prove that

$$FS_2 \succ as_2 \quad \text{and} \quad FS_2 \succ as_3$$

First, we prove $FS_2 \succ as_2$. We would like to show that

$$X \rightarrow Y \vdash_{FS} XZ \rightarrow YZ$$

The proof is as follows:

(1) $X \rightarrow Y$ hypothesis
(2) $YZ \rightarrow YZ$ (I)
(3) $XZ \rightarrow YZ$ (1), (2), and (II)

Now we give the proof of $FS_2 \succ as_3$. We would like to show that

$$X \rightarrow Y, Y \rightarrow Z \vdash_{FS_2} X \rightarrow Z$$

We have the following proof:

(1) $X \rightarrow Y$ hypothesis
(2) $Y \rightarrow Z$ hypothesis
(3) $X \rightarrow Z$ (II)

Because we have established $FS_2 \succ AS$, and as AS is complete for functional dependencies, we obtain, by Proposition 2 (completeness and relation $\succ$), that FS_2 is complete for functional dependencies.

3 Conclusions

We introduced a sound and complete formal system FS_2 for functional dependencies. $FS2$-system consists of only two inference rules: (I) reflexivity and (II) generalized transitivity. Completeness of FS_2 is proved.

An interesting open problem remains to solve: Is there a complete and sound formal system for functional dependencies which contains only one inference rule?

Acknowledgements

I would like to thank the anonymous referees for their valuable comments and suggestions.

References

[Armstrong 74] W. W. Armstrong, Dependency structures of database relationships, Information Processing 74, North Holland Pub. Co. Amsterdam (1974), pp. 580-583.

[Beeri et al. 77] . Beeri, R. Fagin, and J. H. Howard, A complete axiomatization for functional and multivalued dependencies in database relations, Proc. ACM SIGMUD Int. Conf. Management of Data, Toronto, Canada (1977), pp. 47-61.

[Maier 83] D. Maier, The theory of Relational Databases, Computer Science Press, Rockville, Md., (1988.).

[Pichet and Delobel 81] . Pichet and C. Delobel, Designing third normal form relational database schema, RR No. 149, January (1979), Report de Recherche de l'Universite' de Grenoble.

[Saxena and Tripathi 89] P.C. Saxena and R.C. Tripathi, Cancellation Law and a Complete Axiomatization of Functional Dependencies in Relational Database, Computers and Artificial Inelligence, 8,4 (1989.), pp. 347-356.

[Ullman 88] J.D. Ullman, Principles of Database and Knowledge Base Systems, Volume I, Computer Science Press, (1988.).

CYBERNETICS & HUMAN KNOWING

Cybernetics and Human Knowing is a quarterly international multi- and interdisciplinary journal on *second order cybernetics* and its relation and relevance to other interdisciplinary approaches such as semiotics.

The journal is devoted to the new understandings of the self-organizing processes of information in human knowing that have arisen through the cybernetics of cybernetics, or second order cybernetics. This new development within cybernetics is a nondisciplinary approach. Through the concept of self-reference it tries to explore: the meaning of cognition and communication; our understanding of organization and information in human, artificial and natural systems; and our understanding of understanding within the natural and social sciences, humanities, information and library science, and in social practices as design, education, organization, teaching, therapy, art, management and politics.

Because of the interdisciplinary character articles are written in such a way that people from other domains can understand them. Articles from practitioners are accepted in a special section.

Subscription Information

For subscription send a check in DDK or USD, your name and address to *Cybernetics & Human Knowing*, v/Søren Brier, The Royal School of Librarianship, Aalborg Branch, Langagervej 4, DK–9220 Aalborg Øst, Denmark; or pay on Giro account no. 2 84 29 04 to Cybernetics & Human KNowing, Denmark or by Diners Club Card. The amounts are: 320 DDK (individual) and 640 DDK (institutional) for Europe and 360 DDK or 60 USD (individual) and 720 DDK or 120 USD (institutional) for the rest of the world.

Editor and Editorial Board

Information for Authors

To facilitate editorial work and to enhance the uniformity of presentation, authors are requested to send *three* copies of the paper to the Editor, and to prepare the contribution in accordance with the conventions summarized below. Manuscripts will not be returned except for editorial reasons. The language of publication is English. Authors provide printed *double-spaced* manuscripts with wide margins. The following information should be provided on the first page: the title, the author's name and full address, a title not exceeding 40 characters including spaces and a *summary/abstract* in English not exceeding 200 words. Tables, diagrams, reference lists, illustrations and illustration captions should be presented on separate sheets.

Reference to a publication should preferably be made of the author, the year of publication and, when necessary, the page numbers (in parentheses).

Endnotes should be typed double-spaced on a separate sheet and numbered consecutively. They should be as few and as short as possible and should include no reference material, as this should be given in the reference list.

Drawings, graphs, figures and tables must be reproducible originals. They should be presented on separate sheets. Authors will be charged if their illustrations have to be re-drawn.

The Editors reserve the right to correct, or to have corrected, non-native English prose, but the authors should not expect this service. The Journal has adopted U.S. English usage as its norm (this does not apply to other native users of English). Authors are advised to use italics for emphasis, quotations, author's names etc. Accepted papers should be delivered on disc in Wordperfect.

A Look into Cybernetics & Human Knowing

From the Editor of C&HK I received a couple of the journal numbers from which I could choose some *new* matters for the readers of *Informatica.* Let me look into Vol. 3 (1995) No. 1 (the marker used for citations will be **3** (1995) 1, and so on).

—CONTENTS of **3** (1995) 1: This issue is dedicated to cyber-semiotics: the integration of knowledge from second-order cybernetics and the triadic semiotics of C.S. Peirce to a broader framework for understanding information and communication. Papers are: Søren Brier, *Cyber-Semiotics: On Autopoiesis, Code-Duality and Sign Games in Bio-Semiotics;* Jesper Hoffmeyer, *The Swarming Cyberspace of the Body;* Lawrence S. Bale, *Gregory Bateson, Cybernetics, and the Social/behavioral Sciences.* A paper concerning praxis, entitled *A (Cybernetic) Musing: Control 1* was contributed by Ranulph Glanville. Book Reviews presents Robert Theobald's *Turning the Century: Personal and Organizational Strategies for Your Changed World,* reviewed by Tetsunori Koizumi; Asghar T. Minai's *Aesthetics, Mind, and Nature—A Communication Approach to the Unity of Matter and Consciousness,* reviewed by Michaela Ulieru, etc.

Citations from Cybernetics & Human Knowing

Let us list some most challenging citations from C&HK which characterize the aim of the journal.

—[**1** (1992) 1, pp. 5–6, *Rodney E. Donaldson,* Cybernetics & Human Knowing: one possible prolegomenon.] Aspects of Humberto Maturana's brilliant analysis of the difference between transcendental and constitutive ontologies have been intuited by many wise women and men in past millennia and in a variety of cultures—but not by very many, and not with the clarity and scientific sophistication which Maturana's cybernetics offers.

Heinz von Foerster, the father of the idea of second order cybernetics, refers to it as "[a] turn form looking at things out there to looking at itself". ... —once we recognize that perception is an activity and not a passivity—the notions of "communication" and "control", as well as "information", either require redefinition or become quite quietly obsolete. ... We also learn that the past and future are stories we tell ourselves in the present ...

—[**1** (1992) 1, p. 14, *Heinz von Foerster,* Ethics and Second-order Cybernetics.] ... in 1931, Kurt Gödel, then 25 years of age, published an article whose significance goes far beyond the circles of logicians and mathematicians. The title of this article I will give now in English: *On formally undecidable propositions in the Principia Mathematica and related systems.* What Gödel does in his paper is to demonstrate that logical systems, even like those so carefully constructed by Russel and Whitehead, are not immune against undecidables to sneak in.

[P. 17] When the language switches to the track of function it is dialogic. There are of course these noises; some of them may sound like "table", some others like "chair", but there need not be any tables or chairs. These noises are invitations to the other to make some dance steps together. The noises "table" and "chair" bring to resonance those strings in the mind of the other which, when brought to vibration, would produce noises like "table" and "chair": language in its function is connotative.

In its appearance, language is descriptive. When you tell your story, you tell it as it was: the magnificent ship, the ocean, the big sky, and the flirt you had, that made the whole trip a delight. ... The right question is: With whom are you going to dance your story, so that your partner will float with you over the decks of your ship, will smell the salt of the ocean, will let the soul expand over the sky, and there will be a flash of jealously when you come to the point of your flirt.

In its function, language is constructive, because nobody knows the source of your story. Nobody knows and ever will know how it was: because as it was is gone for ever.

—[**1** (1992) 1, p. 22, *Ernst von Glasersfeld,* Why I Consider Myself a Cybernetician.] I had also come across *Claude Shannon's* theory and in the first two pages of his famous paper on *The mathematical theory of communication*[1], he mentions that *meaning* does not travel from a sender to a receiver. The only thing that travels are changes in some form of physical energy, which he called "signals". More important still, these changes in energy are signals only to those who have associated them with a code and are therefore able, as senders, to *encode* their meanings in them and, as receivers, to *decode* them. Too often, in discussion on communication, it is overlooked that the initial code of a particular communication system cannot be established *within* that system but has to be arranged by other means. The communication system we call "natural language" is no different in that regard.

[P. 23] Even monolinguals, when they grow up, sometimes discover that what they thought those other were doing is not what *they* thought they were doing. So they may become aware of discrepancies between their use of certain words and other people's. But since they have to interact not only with things but also with other speakers of the language, they adapt their meanings as best they can to the meanings they believe others to have in their minds. Quite often this leads to the feeling that one "sees things their way". But, as most of us discover, the need for adaptation never ends. In fact, as you advance to old age, you realize how much you are alone in your conceptual world.

On the strength of all this, *I* came to believe that the meanings we attribute to words and phrases, and to whole speeches and texts, are meanings, or built up of meanings, that we ourselves have generated in our own experience. They are the result of "self-regulation"—and the study of self-regulation is an integral part of cybernetics.

—[**1** (1992) 1, pp. 31–32, *Ole Thyssen,* Ethics as Second Order Morality.] Heinz von Foerster[2] has developed the idea of a second order cybernetics. While first order cybernetics deals with observed systems, second order cybernetics deals with observing systems. This involves a shift in perspective. First order cybernetics places an observer outside the observed system, while second order cybernetics places the observer inside a system, which embraces both the observer and the system which he observes and which observes him. Instead of a privileged observer, who monopolizes the rationality of observation, we get a variety of observers, who observe each other without any one having priority. ...

Any observation takes place from a point which is invisible for the observer, while he observes. As von Foerster points out, any observation has a blind spot (von Foerster 1984, p. 289). Blindness is a condition of seeing. We are metaphysicians when we make choices which cannot be made and which nevertheless are made. We create philosophical systems on the basis of arbitrary decisions and try, perhaps, to make them socially obligatory. This happens when, e.g., a research elite decides what is good and what is bad research. Persons and institutions get their identity by making metaphysical choices, because such choices show (unfounded and arbitrarily) who they are and how to act.

[P. 43] Of course a subsystem is unable to transgress its boundary. It operates internally, and it uses its medium to define its operation. ...

—[**2** (1993) 2, p. 4, *Massimo Negrotti and Lars Qvortrup,* Introduction to the Theme: the Theory of the Artificial.] ... artificial devices are built in order to "mimic" something outside the device, be it something in the natural, the psychic or the social world. But what does it mean to "mimic"? ... It is plausible to think that the concept of artificial concretely includes two poles (nature and technology) and generates new realities and not only metaphors. A children's game, a theatrical piece, a painting or a symphony are good examples of artificial realities in the sense that they try to reproduce situations, feelings or ideas by means of 'technologies' like social rules, language rules, color rules or acoustic rules. What kind of objects do they set up? Surely they generate artificial realities, but not in the sense of something false: rather they establish new levels of reality (or recombine other levels of reality in a new one) strictly depending, as in any other case of the artificial, on two factors: the 'selections' of the author and the technological tools adopted to reproduce them.

In this sense, man is much more involved in doing the artificial than in doing something natural and this is, perhaps, his very peculiarity as

[1]Bell Systems Technical Journal, **27**, 379–423, 623–656.

[2]Heinz von Foerster, *Observing Systems*, Intersystems Publications, Salinas, CA, 1980 (Second edition 1984).

compared to lower living systems, though not to all of them. Since to *reproduce* is, *coeteris paribus,* more easy than to *produce ex novo,* humans spend a lot of time reproducing the world, both the external and the internal one, thanks to the various technologies invented during the centuries.

—[2 (1993) 2, pp. 7, *Lars Qvortrup,* Orders of Artificiality.] ... if C is an internal element of the system, then the system is a natural (or social) system, which means that it produces its own conditionality. Thus, the system is an autopoietic system. If, however, C is an external element in relation to the system conditioned by C, then the system is artificial (or at least artificial in condition to C). Such a system does not produce its own condition: it is heteropoietic system. For example, a machine is a heteropoietic system: it may or it may not produce its own elements and relations, but it does not produce its conditionality.

[P. 12] According to Luhman, meaning is at one and the same time produced by the system and externalizing itself in relation to the system. "Communication systems develop a special way to deal with complexity, i.e., introducing a representation of the complexity of the world into the system. I call this representation of complexity 'meaning' ", Luhman writes[3] ...

[P. 13] *Social systems:* Communicate through meaning *as if* meaning is a common denominator. I.e. system A communicates with system B through system A's meaning-conditionality, as if it were also the meaning-conditionality of system B. One person, A, understands other person, B, as if this other person were identical with person A, or, more precisely, if person B shared person A's meaning system. We understand each other egocentrically.

—[2 (1993) 2, p. 22, *Massimo Negrotti,* Towards a Theory of the Artificial.] Artificial is no longer a mere adjective: it is also to be conceived as a substantive. It means that there are artificial objects just as there are natural or technological ones. Artificial is an object which does not wholly overlap either with an object of nature or with an object of technology: it is bound to swing between nature and technology. Far from overlapping with all which is simply "non-natural", this concept is closely dependent on the natural object which it aims at reproducing. The artificial constitutes an original reality (a mix of technology and nature), but is always such "with respect to something else", without which it would turn out to be meaningless as artificial and should be defined as pure technological object, as a pure machine.

—[2 (1994) 3, p. 3–15, *Ervin Laszlo,* Ψ-Field Memory: The Missing Factor of Order.] (A comment: for the sake of readability the markers of skipped text parts are omitted.) ... we group the major kinds of systematic realities under the heading of "alternative universes". There are four of them: the mystical μ-universe, teleological τ-universe, random-change ϕ-universe, self-forming ψ-universe. The origins of the μ-universe go back to the dawn of human intellectual history, with roots in both Eastern and Western thought. It was Plato who introduced it into systematic philosophy. Aristotle objected to this postulate and proceeded to outline the historically powerful variant of the τ-universe. In Aristotle's conception matter is inert and formless. To account for the world's known properties we must assume action of four distinct "causes", including the final cause which makes the universe basically teleological.

The ϕ-universe has an especially rich history, first within philosophy and then in science. Its origins can be traced to the atomism of Democritus and Leucippus. After a career in philosophy, atomism penetrated into science. Scientific atomism viewed the world as the precise and predictable concourse of atoms (or more basic particles) in space and time. The laws of motion that determine all things in the universe are dynamic, invariant and universal. Probed with more precise instruments, the universe refused to behave like a precise mechanism. The uncertainties that came to light suggest not merely limitations of knowledge but basic indeterminacies at the heart of reality. The breakdown of determinism in the ϕ-universe was presaged in Boltzmann's statistical mechanics more than a hundred years ago. The probabilistic variant of the ϕ-universe remains the main reality for contemporary science to this day.

Something is missing from the ϕ-universe—a factor that would introduce the necessary gui-

[3] N. Luhman *Essays on Self-Reference,* New York, 1990; p. 146.

dance or direction into the random concourse of particles of matter. In the hypothesis put forward on these pages the holographic field that acts as the natural memory of the universe is termed the ψ-field. The hypothesis concerns the existence of a mnemic field that conserves the wave-transform of all configurations of matter that arise in the universe and, in the inverse transform, feeds the conserved pattern back to the corresponding configurations. The basis of the ψ-field is the energy potentials believed to be associated with all particles and configuration of particles of matter. The new theory views particles as quantized localizations of the quantum probability field: it postulates that the total wave-field is made up of the energy field frequencies plus Planck's constant. Each particle is described by a probability function in space-time. Living in a ψ-universe, reality is not what commonsense and even science thought it was.

—[**2** (1994) 4, p. 11, *Lloyd Fell* and *David Russel*, Towards a Biological Explanation of Human Understanding.] ... we do not think that the meaning of the words or the "body language" has been transferred from one person to the other. That kind of explanation has lead to the idea that we live in an "information age" and we come together to exchange information rather than interact. ... Our re-framing of the basic biology means that we do not regard cognition as an information-processing operation, but as a constitutive mechanism of living things. ... From the biology of Maturana and Varela we can say—as Mingers (1991)[4] has done—that language is essentially connotative rather than denotative. ... The fact that we often reach the agreement about the meaning of a word or scientific concept is a testament to our ability to reach agreement, not a proof that such an entity exists in reality. We would say that the meaning of anything lies in the relationship which we make with it. Therefore we are saying that meaning is not transferable–it is formed individually in the course of conversation.

—[**2** (1994) 4, p. 41, *Asghar T. Minai*, Information and Aesthetics.] Since the ultimate reality for me is "information" then the usual metaphysical problems arise; from the one how did the existing many derive? Unity has no differentiations. Following Hegel, Nietzsche, Heidegger, Derrida, I emphasize the "difference" principle that is associated with chance, the irrational, the spontaneous and the individual aspects of reality as opposed to the necessary, the rational, the formal, and the universal aspect of things.

—[**3** (1995) 1, p. 5, *Søren Brier*, Cyber-Semiotics: On Autopoiesis, Code-Duality and Sign Games in Bio-Semiotics.] The idea is to make computer manipulate signals which to humans have symbolic value in a logic syntax describable in algorithms in such a way that it becomes meaningful to other language users. But although there is a lot of talk of semantics and symbols in Cognitive Science, the concept of information seems to be like Wiener's (1961)[5] combination of Shannon's statistical information theory and Boltzmann's probabilistic understanding of thermodynamics (Brier 1992)[6]. This is again combined with a linguistic theory which claims that the semantic content that symbols of a sentence represent can be determined through truth tables. The basic reality of a sentence is seen as a logical structure with semantically empty symbols. The semantical content can then be "poured into the symbols" by their capacity to refer to things, and determined through the truth tables. Reason and the working of language in communication is seen, roughly, as fitting the model of formal logic. The core of meaning, intelligence and reasoning is seen as logical algorithms.

From a biological cybernetic and semiotic point of view this theory overlooks various fundamental characteristics of biological systems: self-organization, closeness, complexity, autonomy, self-interest in survival, life history and intentionality. The mechanistic idea of reason and knowledge—and of logic, of course—has led to a simplistic understanding of how meaning functions in both language and in practice. The mechanicists hope that the causal interaction of symbols can be explained through their syntactic relations.

—[**3** (1995) 1, p. 16, *Jesper Hoffmeyer*, The Swar-

[4] J. Mingers (1991), *The Cognitive Theories of Maturana and Varela*, Systems Practice **4** pp. 319–338.

[5] N. Wiener, *Cybernetics or Control and Communication in the Animal and the Machine*, The M.I.T. Press and John Wiley & Sons, New York (Second edition).

[6] S. Brier, *Information and Consciousness: A Critique of the Mechanistic Foundation for the Concept of Information*, C&HK **1** (1992) 2/3.

ming Cyberspace of the Body.] If the 'observation of observing systems' is the core of second order cybernetics, then certainly biology is a kind of second order cybernetics, for living systems are definitely observing systems.

[P. 17] DNA does not contain the key to its own *interpretation.* In a way the molecule is hermetic. In the prototype case of sexually reproducing organism, only the fertilized egg 'knows' how to interpret it, i.e., to use its text for the construction of the organism. The interpreter of the DNA message is buried in the cytoskeleton of the fertilized egg (and the growing embryo). This in turn is the product of history, i.e., of the billions of molecular habits acquired through the evolution of the eukaryotic cell (Margulis 1981)[7] in general and the successive phylogenetic history of the species in particular. (It took evolution two billion years to produce this marvelous entity, the eukaryotic cell. Having accomplished this deed, evolution spent only one and a half billion years producing all the rest.)

[P. 22] There is a growing awareness among immunologists that the separation of the immune system from the rest of the body, and especially from the brain, is illusory. Not only the nerve fibers branching into the organs of the immune system, thymus, lymph glands, bone marrow and spleen, but more importantly, a major conceptual shift in neuroscience has been wrought by the realization that brain function is modulated by numerous chemicals in addition to classical neurotransmitters. Many of these informational substances are *neuropeptides.* Their number presently exceeds 50 and most, if not all, alter behavior and mood. We now recognize that *their semiotic specificity resides in receptors* rather than the close juxtaposition occurring at classical synapses.

—[**3** (1995) 1, p. 28–29, *Lawrence S. Bale,* Gregory Bateson, Cybernetics, and the Social/behavioral Sciences.] System theory emerged out of the need to map and explain biological phenomena that cannot be suitable understood using the classical mechanistic model of reality. "The analytic, mechanistic, one-way causal paradigm of classical science" (Bertalanffy, 1968, p. xxi)[8], as Austrian biologist Ludwig von Bertalanffy describes it, assumes that reality can be quantifiably analyzed; that a whole can be understood in terms of its parts; and that the nature and function of a substance or an organism can be comprehended by reducing it to its material, externally observable components.

...the successes garnered by the classical scientific paradigm revealed its inadequacies. As refined tools have opened wider panoramas of research, exhibiting data of increasing complexity, science has been driven to search for new ways of conceptualizing reality. In short, the classical paradigm of science has proven inadequate to the task of mapping the natural world. It is particularly inadequate when applied to describing and explaining the multivariable processes of human interaction, e.g., communication, and human-kind's intricate interrelationship with local and global ecological systems.

[P. 38] Note how the cybernetic paradigm shifts the focus of our discourse away from: discreet material substances, one-way causality, structure, and summativity. Rather, a cybernetic explanation focuses on: process and behavior, dynamic or animated organization, circular or more complex than circular causality, the mutual causal loops of feedback cycles, interaction between multiple variables, and emergent morphogenesis. Hence, the cybernetic stability of a system must be understood not as an inactive structure, but as a pattern of events—an animated organization of exchanges and transformations within the system's parameters. Hence, it is not the characteristics of the "parts" alone that are basic to any whole. Rather, it is the manner in which the system's differentiated components are interrelated that gives them their distinctive properties. Furthermore, within more complex systems the "differentiated parts" exhibit properties which they owe specifically to being components of a larger whole.

Is this presentation instructive for the readers and authors of *Informatica*? I hope so, thanking Søren Brier, the editor, giving me the opportunity to write this report.

A.P. Železnikar

[7] L. Margulis, *Symbiosis in Cell Evolution: Life and Its Environment on Earth,* Freeman, San Francisco 1981.

[8] L. von Bertalanffy, *General Systems Theory,* George Braziller, New York 1968.

Report: ISCA '95

The 22th Annual International Symposium on Computer Architecture was held in Santa Margherita Ligure, Italy, on June 18-24, 1995. Last time this symposium was organized in Europe in 1983. Program Committee, led by James E. Smith, selected 37 out of 180 submitted papers from France, Germany, Japan, Spain, Sweden, Switzerland and USA. Papers were scheduled in the following sessions: Multiprocessors and Applications, Cache Coherence, Interconnect Technology and I/O, Instruction Level Parallelism, New Microarchitectures, Managing Memory Hierarchies, Interconnection Network Routing, Novel Memory Access Mechanisms, Branch Prediction, System Evaluation, Instruction Fetching, Caches, and Processor Architecture.

Pre-conference tutorials were:
- High Performance Microprocessors
- Introduction to Parallel Computing
- Introducing Parallel Databases for OLTP and Query Processing to Computer Architects
- Reliable, Parallel Storage Architecture: RAID and Beyond
- Parallelizing Compilers and Beck-end Optimizations
- Distributed Shared memory: Concepts and Systems
- A Methodology and Tool for Performance Prediction and Evaluation of Complex Computing Platforms
- Continuing to Exploit Instruction Level Parallelism: Issues, Bottelnecks, and Enabling Conditions

In addition, the following workshops were organized:
- Undergraduate Computer Architecture Education
- Pre-Hardware Performance Analysis Techniques: How Good are We at this?
- Fifth Workshop on Scalable Shared-Memory Multiprocessors

There were almost 300 participants (from 20 countries) among which also researchers from Computer Systems Dept. of the Jožef Stefan Institute, Ljubljana, Slovenia. The next ISCA will be in Philadelphia, USA.

Jurij Šilc

Report: EURO-PAR'95

The EURO-PAR'95 conference was the first in a series of annual EURO-PAR conferences, formed by merging the PARLE and CONPAR-VAPP conference series on parallel processing, with the intent to create the main annual scientific event on parallel processing in Europe. The scope of the series covers the full spectrum of parallel processing, ranging from theory to design and applications. The objective is to provide a forum to promote the development of parallel computing, both as an industrial technique and as an academic discipline, extending the frontier of both the state of the art and the state of the practice.

The EURO-PAR'95 conference was organized in Stockholm by the Swedish Institute of Computer Science and the Department of Teleinformatics at KTH. The call for papers resulted in 196 submissions; the selection process resulted in 50 regular papers and 11 posters being accepted for presentation on one of the following sessions: Parallel Algorithms, Language Implementation, Compiling Techniques, Semantics and Tools, Loop Parallelization, Architecture Design, Interconnection Networks, Cache Systems, Load Balancing, Scheduling, Fault Tolerance and SIMD Arrays, Applications, and Posters. The presentations included *A model for efficient programming of dynamic applications on distributed memory multiprocessors* (A.Erzmann et al.), *Scheduling master-slave multiprocessor systems* (S.Sahni), *Using knowledge-based techniques for parallelization on parallelizing compilers* (Ch.-T. Yang et al.), *Efficient solutions for mapping parallel programs* (P.Bouvry et al.), *Efficient run-time program allocation on a parallel computer* (J.Šilc, B.Robič).

Three invited talks were given during the conference by Greg Papadopoulos (*Mainstream parallelism: Taking sides on the SMP/MPP/Cluster Debate*), Björn Enquist (*The Oz programming model*), and Gert Smolka (*Parallelism in computational algorithms and the physical world*).

Conference proceedings are published by Springer-Verlag in the *Lecture Notes in Computer Science* series, vol. 966.

The next EURO-PAR conference will take place in Lyon, France, in August 1996.

Borut Robič

Announcement of the forthcomming issue of INFORMATICA on the topic:

MIND <> COMPUTER

[i.e. Mind NOT EQUAL Computer]

In this special issue we want to reevaluate the soundness of current AI research positions (especially the heavily disputed strong-AI paradigm) as well as pursue new directions aimed at achieving true intelligence. This is a brainstorming special issue about core ideas that will shape future AI. We are interested in critical papers representing all positions on the issues.

The first part of this special issue will be a small number of invited papers, including papers by Winograd, Dreyfus, Michie, McDermott, Agre, Tecuci etc. Here we are soliciting additional papers on the topic.

TOPICS: Papers are invited in all subareas and on all aspects of the above topic, especially on:

- the current state, positions, and advancements achieved in the last 5 years in particular subfields of AI,
- the trends, perspectives and foundations of natural and artificial intelligence,
- strong AI versus weak AI and the reality of most current "typical" publications in AI,
- new directions in AI.

TIME TABLE AND CONTACTS: Papers in 5 hard copies had to be received by May 15, 1995 at one of the following addresses (please, no e-mail/FAX submissions):

North & South America:
Marcin Paprzycki
`paprzycki_m@gusher.pb.utexas.edu`
Department of Mathematics and
Computer Science
University of Texas of the Permian Basin
Odessa, TX 79762, USA

Asia, Australia:
Xindong Wu
`xindong@insect.sd.monash.edu.au`
Department of Software Development,
Monash University
Melbourne, VIC 3145, Australia

Europe, Africa:
Matjaz Gams
`matjaz.gams@ijs.si`
Jozef Stefan Institute, Jamova 39
61000 Slovenia, Europe

E-mail information about the special issue is available from the above 3 contact editors.

The special issue will be published in late 1995.

FORMAT AND REVIEWING PROCESS: Papers should not exceed 8,000 words (including figures and tables but excluding references. A full page figure should be counted as 500 words). Ideally 5,000 words are desirable.

Each paper have been refereed by at least two anonymous referees outside the author's country and by an appropriate subset of the program committee.

When accepted, the authors will be asked to transform their manuscripts into the Informatica LaTeX style (available from `ftp.arnes.si`; directory `/magazines/informatica`).

More information about Informatica and the Special Issue can be accessed through URL: `ftp://ftp.arnes.si/magazines/informatica`.

TIME-96: Third International Workshop on Temporal Representation and Reasoning

Key West, Florida, USA May 19-20, 1996

CALL FOR PAPERS

The purpose of this workshop is to bring together active researchers in the area of temporal representation and reasoning in Artificial Intelligence. Through paper presentations and discussions, the participants will exchange, compare, and contrast results in the area. The workshop is planned as a two day event to immediately precede FLAIRS-96 (Ninth Annual Florida Artificial Intelligence Research Symposium, May 20-22). Workshop participants are also encouraged to submit papers to FLAIRS and attend the conference. The workshop will be conducted as a combination of paper presentations, a poster session, an invited talk, and panel discussions. The format will provide ample time for discussions and exchange of ideas. Submission of high quality papers describing mature results or on-going work are invited for all areas of temporal representation and reasoning, including, but not limited to:

temporal logics and ontologies
temporal languages and architectures
continuous versus discrete time
point versus interval representations
expressive power versus tractability
belief and uncertainty in temporal knowledge
temporal databases and knowledge bases
temporal learning and discovery
reasoning about actions and events
time and nonmonotonism
time and constraints
time in problem solving (e.g. diagnosis, qualitative physics,...)
multiple agents, communication, and synchronization
applications

To maximize interaction among participants, the size of the workshop will be limited. Accepted papers will be invited for full presentation or a poster presentation. All submissions must be received by December 5, 1995. Notification of acceptance or rejection will be sent to the first author (or designated author) by February 26, 1996. Prospective participants should submit 5 copies of a 6-8 page paper (indicating the selected areas) to:

TIME-96
Luca Chittaro and Angelo Montanari
Dipartimento di Matematica e Informatica
Universita' di Udine
Via delle Scienze, 206
33100 Udine - ITALY

Or preferably, submit by anonymous FTP a postscript version of the paper to:

`bach.dimi.uniud.it` (158.110.1.140)
directory: `pub/time96`

After submitting electronically, please communicate the title, area, and authors of the paper by sending an e-mail to time96@dimi.uniud.it (this address can also be used to obtain further information). The TIME-96 workshop has also a web page at: http://www.seas.smu.edu/ mario/time-workshop/welcome.html

1 Publication

All accepted papers will be published in the workshop proceedings. As well, a selected subset of the papers will be invited for inclusion (subject to refereeing) in a book or in a special issue of a journal.

2 Organization

GENERAL Chairs

Scott Goodwin and Howard Hamilton, University of Regina, Canada

PROGRAM COMMITTEE Chairs

Luca Chittaro and Angelo Montanari, Universita' di Udine, Italy

PROGRAM COMMITTEE

Frank Anger, University of West Florida, USA Fahiem Bacchus, University of Waterloo, Canada Mark Boddy, Honeywell Systems and Research Center, USA Luca Chittaro (co-chair), Universita' di Udine, Italy Jan Chomicki, Kansas State University, USA Philippe Dague, Universite Paris-Nord, France Tom Dean, Brown University, USA Mark Denecker, K. University Leuven, Belgium Jennifer Elgot-Drapkin, Arizona State University, USA Marcelo Finger, Imperial College, UK Michael Fisher, Manchester Metropolitan University, UK Dov Gabbay, Imperial College, UK Malik Ghallab, LAAS-CNRS, France Anthony Galton, University of Exeter, UK Michael Gelfond, University of El Paso, USA Michael Georgeff, Australian AI Institute, Australia Peter Haddawy, University of Wisconsin-Milwaukee, USA Pat Hayes, University of Illinois at Urbana-Champaign, USA Peter Ladkin, Universitaet Bielefeld, Germany Gerard Ligozat, Universite Paris XI, France Angelo Montanari (co-chair), Universita' di Udine, Italy Robert Morris, Florida Institute of Technology, USA Bernhard Nebel, Universitaet des Saarlandes, Germany Don Perlis, University of Maryland, USA Han Reichgelt, University of the West Indies, Jamaica Raymond Reiter, University of Toronto, Canada Mark Reynolds, Kings College, UK Maarten de Rijke, CWI, The Netherlands Erik Sandewall, Linkoping University, Sweden Marek Sergot, Imperial College, UK Murray Shanahan, Imperial College, UK Peter van Beek, University of Alberta, Canada Andre Trudel, Acadia University, Canada

ORGANIZING COMMITTEE

Peter Haddawy, University of Wisconsin-Milwaukee Robert Morris, Florida Institute of Technology Andre Trudel, Acadia University Peter van Beek, University of Alberta

SPONSORING ORGANIZATIONS

Sponsorship for TIME-96 is being sought from the American Association for Artificial Intelligence (AAAI), Florida Artificial Intelligence Research Society (FLAIRS), and the University of Udine.

3 Summary of Important Dates

December 5, 1995 – Submission deadline
February 26, 1996 – Notification of acceptance
April 15, 1996 – Camera-ready copy deadline
May 19-20, 1996 – TIME-96 Workshop
May 20-22, 1996 – FLAIRS-96 Conference

Machine Learning List

The Machine Learning List is moderated. Contributions should be relevant to the scientific study of machine learning. Mail contributions to `ml@ics.uci.edu`. Mail requests to be added or deleted to `ml-request@ics.uci.edu`. Back issues may be FTP'd from `ics.uci.edu` in `pub/ml-list/V<X>/<N>` or N.Z where X and N are the volume and number of the issue; ID: anonymous PASSWORD: <your mail address> URL- `http://www.ics.uci.edu/AI/-ML/Machine-Learning.html`

First Workshop on Numerical Analysis and Applications

Russe, Bulgaria

June 24–27, 1996

Organizers

University of Russe, Association of Bulgarian Mathematicians - Russe

Co-organizers

Institute of Mathematics and Center for Informatics and Information Technologies of the Bulgarian Academy of Sciences, Technical University of Gabrovo, Technical University of Sofia

Traditionally every 4 years a Conference on Numerical Analysis and Applications is organized in Bulgaria. The present workshop is meant to support this tradition and to serve as an intermediate meeting between these conferences. We would like to give an opportunity for mathematicians and applied scientists to discuss topics of common interest.

The workshop will have three tracks:
1. Numerical linear algebra.
2. Numerical methods for differential equations.
3. Numerical modelling.

Preliminary list of Invited Speakers

R. Bisseling (Netherlands), L. Brugnano (Italy), S. K. Godunov (Russia), A. Griewank (Germany), A. Hadjidimos (USA), S. Hammarling (UK), W. Hofmann (Germany), A. Karageorghis (Cyprus), Yu. A. Kuznetsov (Russia), R. Maerz (Germany), W. T. Pickering (UK), R. Plemmons (USA), I. V. Puzynin (Russia), G. I. Shishkin (Russia), T. Szulc (Poland), E. E. Tyrtyshnikov (Russia), W. Varnhorn (Germany), V. V. Voevodin (Russia), Z. Zlatev (Denmark).

Organizing committee

L. Vulkov (Chair), P. Yalamov (co-Chair), A. Andreev, P. Ivanova, I. Lirkov, M. Paprzycki, V. Pavlov, S. Romanova, T. Todorov, K. Zlateva.

We would like to invite all interested individuals to ORGANIZE a MINISYMPOSIUM related to one or more of the conference tracks. Please send a minisymposium abstract (approximately one page) and a list of 4-8 speakers to one of the addresses listed below. The deadline for proposals is December 31, 1995.

A general call for papers and more details about the meeting will be provided in the future announcements.

For more information, please, contact:
Plamen Yalamov, Dept. of Mathematics, University of Russe, 7017 Russe, BULGARIA, yalamov@iscbg.acad.bg
Marcin Paprzycki, Dept. of Mathematics and CS, UT Permian Basin, Odessa, TX 79762, USA, paprzycki_m@gusher.pb.utexas.edu

THE MINISTRY OF SCIENCE AND TECHNOLOGY OF THE REPUBLIC OF SLOVENIA

Address: Slovenska 50, 61000 Ljubljana, Tel.: +386 61 1311 107, Fax: +386 61 1324 140.
WWW:http://www.mzt.si
Minister: Prof. Rado Bohinc, Ph.D.
State Secretary for Int. Coop.: Rado Genorio, Ph.D.
State Secretary for Sci. and Tech.: Ciril Baškovič
Secretary General: Franc Hudej, Ph.D.

The Ministry also includes:
The Standards and Metrology Institute of the Republic of Slovenia
Address: Kotnikova 6, 61000 Ljubljana, Tel.: +386 61 1312 322, Fax: +386 61 314 882., and
The Industrial Property Protection Office of the Republic of Slovenia
Address: Kotnikova 6, 61000 Ljubljana, Tel.: +386 61 1312 322, Fax: +386 61 318 983.

Scientific Research and Development Potential. The statistical data for 1993 showed that there were 180 research and development institutions in Slovenia. Altogether, they employed 10,400 people, of whom 4,900 were researchers and 3,900 expert or technical staff.

In the past ten years, the number of researchers has almost doubled: the number of Ph.D. graduates increased from 1,100 to 1,565, while the number of M.Sc.'s rose from 650 to 1,029. The "Young Researchers" (i.e. postgraduate students) program has greatly helped towards revitalizing research. The average age of researchers has been brought down to 40, with one-fifth of them being younger than 29.

The table below shows the distribution of researchers according to educational level and sectors (in 1993):

Sector	Ph.D.	M.Sc.
Business enterprises	51	196
Government	482	395
Private non-profit organizations	10	12
Higher education organizations	1022	426
Total	1,565	1,029

Financing Research and Development. Statistical estimates indicate that US$ 185 million (1,4% of GDP) was spent on research and development in Slovenia in 1993. More than half of this comes from public expenditure, mainly the state budget. In the last three years, R&D expenditure by business organizations has stagnated, a result of the current economic transition. This transition has led to the financial decline and increased insolvency of firms and companies. These cannot be replaced by the growing number of mainly small businesses. The shortfall was addressed by increased public-sector spending: its share of GDP nearly doubled from the mid-seventies to 0,86% in 1993.

Income of R&D organizations spent on R&D activities in 1993 (in million US$):

Sector	Total	Basic res.	App. res.	Exp. dev.
Business ent.	83,9	4,7	32,6	46,6
Government	58,4	16,1	21,5	20,8
Private non-p.	1,3	0,2	0,6	0,5
Higher edu.	40,9	24,2	8,7	8
Total	184,5	45,2	63,4	75,9

The policy of the Slovene Government is to increase the percentage intended for R&D in its budget. The Science and Technology Council of the Republic of Slovenia is preparing the draft of a national research program (NRP). The government will harmonize the NRP with its general development policy, and submit it first to the parliamentary Committee for Science, Technology and Development and after that to the parliament. The parliament approves the NRP each year, thus setting the basis for deciding the level of public support for R&D.

The Ministry of Science and Technology is mainly a government institution responsible for controlling expenditure of the R&D budget, in compliance with the NRP and the criteria provided by the Law on Research Activities. The Ministry finances research or co- finances development projects through public bidding, partially finances infrastructure research institutions (national institutes), while it directly finances management and top-level science.

The focal points of R&D policy in Slovenia are:

- maintaining the high level and quality of research activities,
- stimulating collaboration between research and industrial institutions,
- (co)financing and tax assistance for companies engaged in technical development and other applied research projects,
- research training and professional development of leading experts,
- close involvement in international research and development projects,
- establishing and operating facilities for the transfer of technology and experience.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of Slovenia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia
Tel.:+386 61 1773 900, Fax.:+386 61 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA
AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of the original figures on separate sheets and the text on an IBM PC DOS floppy disk or by e-mail – both in ASCII and the Informatica LaTeX format. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 61111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than two years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

Referees:

David Abramson, Catriel Beeri, Azer Bestavros, Jacek Blazewicz, Laszlo Boeszoermenyi, Ivan Bratko, Jerzy Brzezinski, Marian Bubak, Wojciech Buszkowski, Ryszard Choras, David Cliff, Travis Craig, Tadeusz Czachorski, Luc De Raedt, Georg Dorfner, Maciej Drozdowski, Hesham El-Rewini, Pierre Flener, Terrence Forgarty, Hugo de Garis, Eugeniusz Gatnar, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Inman Harvey, Elke Hochmueller, Rod Howell, Ryszard Jakubowski, Eric Johnson, Li-Shan Kang, Roland Kaschek, Jan Kniat, Gilad Koren, Henryk Krawczyk, Ben Kroese, Zbyszko Krolikowski, Benjamin Kuipers, Phil Laplante, Bud Lawson, Ulrike Leopold-Wildburger, Joseph Y-T. Leung, Raymond Lister, Doug Locke, Andrzej Marciniak, Vladimir Marik, Jacek Martinek, Florian Matthes, Dieter Merkl, Zbigniew Michalewicz, Roland Mittermeir, Tadeusz Morzy, Daniel Mossé, Jerzy Nogieć, Stefano Nolfi, Tadeusz Pankowski, Warren Persons, Gustav Pomberger, Gary Preckshot, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Ewaryst Rafajlowicz, Wolf Rauch, Peter Rechenberg, Felix Redmill, Bo Sanden, Guenter Schmidt, William Spears, Przemysław Stpiczyński, Maciej Stroinski, Tomasz Szmuc, Jurij Tasič, Ken Tindell, A Min Tjoa, Wieslaw Traczyk, Marek Tudruj, Andrzej Urbanski, Janusz Zalewski, Yanchun Zhang, Gerhard Widmer

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or Board of Referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board and Board of Referees are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Volume 19 Number 3 September 1995 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Contents: