Volume 18 Number 1 April 1994 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Profile: Robert Trappl
Guest Editor: Miroslav Kubat
Diagnosis of Technical Systems
Closure of Database Relation
IJCAI'93: Critical

The Slovene Society Informatika, Ljubljana, Slovenia

Informatica 18 (1994) Number 18, pp. 1–105

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: ftp://ftp.arnes.si/magazines/informatica

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 1994 (Volume 18) is
- DEM 50 (US$ 34) for institutions,
- DEM 25 (US$ 17) for individuals, and
- DEM 10 (US$ 4) for students

plus the mail charge DEM 10 (US$ 4).

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, Department for Computer Science, Jožef Stefan Institute: Tel (+386) 61 1259 199, 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)
Authomatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences (Janez Peklenik)

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

PROFILES

For the first time in its Profiles history, *Informatica* takes the opportunity to introduce one of its editors who is not only a distinguished researcher, computer scientist and professional AI-activity organizer, but also a technological innovator and the leader of one of the largest and the most important industrial projects in the history of AI. In the SQUAD project at NEC Corporation, over 20,000 software engineers are participating in the implementation of knowledge engineering for software quality control.

It is certainly important to stress at least one of the major innovative fields of *Hiroaki Kitano,* the *massively parallel artificial intelligence.* This field is a consequence of his scientific and research work concerning his speech-to-speech translation system, implemented for a 'live' Japanese-to-English translation on several massively parallel machines like CM-2, IXM-2, and SNAP-1.

It is not for the first time that the readers of *Informatica* can learn about some innovative *Kitano*'s achievements. A short overview of his professional activity was already reported in *Informatica,* **18** (1994), pp. 97–99 (M. Gams, *Report: IJCAI'93—A Critical Review*). So let us cite some characteristic sequences from there.

The Computers and Thought Award for outstanding young scientists (below 35 years) was given to Hiroaki Kitano.

... According to the reviewers, Kitano won the prize for novel translation work on a massively parallel computer. His approach builds on memory as the foundation of intelligence. Each of computer's 32,000 processors contains a few sentences and determines the best match for an input sentence in natural language. Translation is based on extensive use of memory. Statements are parsed into basic parts, translated, and sensibly combined to the translated language. All reasoning is by analogy. The program achieves around 75-percent accuracy by one reports and around 90-percent in others.

... He categorises complete, correct and consistent problem domains as toy domains. Real-world domains are denoted as incomplete, incorrect, inconsistent, and are in addition characterised by human bias, tractability and economics.

... a great part of AI research is devoted to toy problems which are unscalable not only to real-world problems but also to intelligent systems.

... AI as well as computer science and engineering in general are much more dependent on the development of computer hardware that generally recognised. Future computer projects can be characterised by massive computing, massive parallelism, massive data.

The readers of *Informatica* may agree how the preceding dictum illuminates complementarily not only the work but also the philosophical and engineering orientation of Hiroaki Kitano. It gives significant hints not only to researchers but also innovators in computer and software engineering. The adjective *massive* in respect to processors, memory and data comes close to something called the *informational* and regards an in componential manner *massive* system (machine) in any respect, that is, in the sense of an informational machine which supports systemically the informing of its hardware and software components (entities).

Hiroaki Kitano

Hiroaki Kitano is a researcher at Sony Computer Science Laboratory, Tokyo, Japan. He holds a Ph.D. in computer science from Kyoto University. Since 1988, he was a visiting scientist at Center for Machine Translation, Carnegie Mellon University. In 1993, he received the Computers and Thought Award from IJCAI for his accomplishments in speech-to-speech translation and massively parallel AI research. He is currently serving as a member of several important committees, such as the advisory committee for IJCAI, the international committee on evolutionary computation, the program committee for AAAI-94, etc.

His scientific contribution spans broad range of AI research. In 1989, Kitano developed one of the first speech-to-speech translation system, a system which translates spoken Japanese into English producing vocal outputs. The system, Dm-Dialog system, was also the first system, so far the only one, which demonstrates the possibility of simultaneous interpretation by machines. Kitano, based on his experiences as a professional interpreter, proposed an almost concurrent parsing and generation model so that, in some occasions, translation can be generated even before the completion of an input sentence. The Dm-Dialog system employs massively parallel marker-passing to carry out computation, and based on

memory-based translation approach.

Due to highly parallel nature of the computing scheme in Dm-Dialog, his system was implemented on several massively parallel machines, such as CM-2 connection machine, IXM-2 associative memory processor, and SNAP-1 semantic network array processor. These implementations demonstrate very high performance translation, translation in milliseconds order, which made a decisive case for massively parallel approach for spoken language translation. In fact, ATR Interpreting Telephony Research Laboratory, a Japanese semi-government research institute, made a decision to pursue a massively parallel memory-based approach to develop its speech-to-speech translation system.

The impact of Kitano's work was not limited to natural language community, it was the first application of massive parallelism to AI for highly complex tasks. Kitano also challenged to expand horizon of high performance computing using the state-of-the-art semi-conductor technology. In 1992, he designed a special purpose wafer-scale integration (WSI), called WSI-MBR. It is dedicated device for memory-based reasoning (MBR). The architecture exploits the redundunt nature of MBR to overcome inevitable fabrication defects of WSI. With its special analog-digital hybrid design, the performance of a WSI-MBR on an eight-inch wafer is expected to be equivalent to 70 Tera-Flops in digital computers. It enables over 2 million parallelisms in one wafer. The low cost, small size, and high-performance of WSI-MBR have significant potential to re-shape the future of computing and industry.

The success of these projects led him to promote a new field in AI—massively parallel artificial intelligence. He chaired a panel on this subject in IJCAI-91, held a tutorial in IJCAI-93, and recently edited a book *Massively Parallel Artificial Intelligence* (Kitano and Hendler, Eds.) published from AAAI Press/The MIT Press.

In 1990, he published a seminal paper on the combination of genetic algorithms (GA) and neural networks (NN). His paper, appeared in Complex System journal, introduced a developmental stage in combining GA and NN. This is the first work to integrate evolution, development, and learning. This work quickly gets attention in GA and ALife community, and a number of research projects was initiated and inspired by his method.

Kitano has a substantial contribution to case-based reasoning and corporate information system field, too. While at NEC Corporation, he leads the SQUAD project, which is a project to develop a large scale corporate-wide case-based system on the software quality control domain. The SQUAD system has over 25,000 cases in various areas of software quality control, and the size of the case-base increases at the speed of 3,000 cases a year. In one of the largest *knowledge engineering* projects undertaken in the AI history, over 20,000 software engineers are involved in reporting cases throughout the company.

Aside from his scientific work, Kitano actively promoted important new areas in AI, such as massively parallel AI, entertainment, and grand challenge AI applications. He chaired several important workshops such as the First International Workshop on Parallel Processing for AI (PPAI-91) held in conjunction with IJCAI-91 at Sydney, Australia, and the Second International Workshop on Parallel Processing for AI (PPAI-93), IJCAI-93, at Chambery, France. Back in Japan, he established SIG-PPAI, a new SIG under Japanese Society for Artificial Intelligence (JSAI). SIG-PPAI is a dedicated forum to discuss parallel processing for AI.

In 1991, he organized and chaired the workshop on Grand Challenge AI Applications, at Tokyo, Japan, which triggered several key research projects in Japan each of which addresses grand challenge applications. In 1993, he organized a panel "Grand Challenge AI Applications" at IJCAI-93 to discuss international aspects on grand challenge. His recent emphasis in promoting a new area is entertainment. At AAAI-94, he co-chaired the workshop on AI, Alife, and Entertainment. He is currently planning several other workshops to help researchers to explore a new area of AI research.

Kitano serves as an associate editor in several of the major journals, such as Journal of Evolutionary Computation, and Applied AI, and is a member of the editorial board for Journal of AI Research, Journal of Theoretical and Experimental AI (JETAI), Artificial Life journal, and *Informatica.*

Edited by *A.P. Železnikar*

Evolution of Methods in Object Schema

Z. Tari and X. Li
School of Information Systems
Queensland University of Technology
GPO Box 2324, Brisbane, Australia
zahirt@icis.qut.edu.au, xueli@fit.qut.edu.au

Keywords: Object-oriented databases, schema evolution, methods, proof correctness

Edited by: Xindong Wu
Received: May 12, 1994 **Revised:** September 20, 1994 **Accepted:** October 6, 1994

This paper deals with the problem of method restructuring in object-oriented databases. We propose a framework that allows a change of methods while maintaining database consistency. A set of minimal changing operations which affect all parts of methods is provided. The semantics of methods evolution relates to two levels of granularity of a schema. The first level concerns the evolution of methods in the context of class inheritance hierarchy, whereas the second level relates to behavioural evolution in which the chain of calling relationships between methods is considered.

1 Introduction

The purpose of this paper is to show how behavior of an object-oriented (O-O) schema can be updated by keeping the state of the database consistent. This problem is generally involved in a more global problem, namely *schema evolution*. This problem relates to the ability of updating information within a schema. Schema evolution is an important facility because most of the design of database applications calls for frequent changes to catch up to changes in reality.

Schema evolution has been primarily performed on the relational model. With the emergence of richer models (e.g., O-O models), schema evolution is addressed in a more general way, i.e. not only on the structural perspective as it was done in the relational model. In this paper we consider the problem of schema evolution in O-O systems. This problem concerns the ability to safely alter a schema both structurally and behaviourally.

- *Structural alterations* may be performed at different levels of granularity within a schema: class level, relationship level, instance level. Several works, e.g. [4,13,14,16], have investigated these different levels of structural modifications. For each level a set of alteration operations and semantic rules to ensure consistency are provided. Many products have been enhanced by the above facilities (e.g. Gemstone, O_2, Orion). A detailed study of schema evolution in such systems can be found in [15,8].
- *Behavioural alterations* concern the evolution of the behaviour of objects. The operations relate to the possibility of modifying both the signature and the code of the methods, and ensuring the correctness of these operations. Some approaches have been provided (e.g., [1,9,11,17,18]) in which some structural changes, such as updating relationships between objects, may affect the behavior of objects. In other words, they are considered as behavior updates.

The literature has widely addressed the first above aspect. However, regarding the second above aspect, i.e. the modifications of methods, to our knowledge no system fully supports such an aspect. Indeed, the research in this area may be considered as a "sample view" of the problem of method changing, and it does not cover every part of a method.

This paper provides our solution to the problem of method evolution in O-O schema. The problem of behavioural consistency is also addressed.

The behaviour of an O-O schema is considered here as a pattern of activities coherent to the schema. Method evolution refers to updating the methods using operations which include *deletion*, *insertion*, and *modification*. These, after their use, may generate inconsistencies which involve

(1) run-time type errors,

(2) side-effects,

(3) unchanged methods made redundant or meaningless, and

(4) unexpected results given by methods.

In this paper, solutions to the problems (1)-(4) cited above are provided. To do so, a fundamental issue is the way in which methods are represented. The semantics of methods is represented declaratively by using appropriate data structures, such as

- a graph for representing method overriding and the inheritance relationships between methods; and
- a graph for describing overloading semantics and the method-calling relationships.

Because the information about method behaviours is represented in graphs, the behavioural inconsistency problems can be checked out by searching the states represented in graphs. By making the information that constrains the method behaviours available to a method-changing handling mechanism, we address the strategies used to deal with behavioural inconsistency problems.

Briefly, the approach we propose has three aspects:

- method representation framework,
- the method evolution operations, and
- the checking strategies.

The first aspect is addressed by proposing appropriate graph data structures that represent all the information about methods. A set of operations are provided for the second aspect, and this covers all the possible modifications of a method schema. The last aspect is addressed by checking the inconsistent behaviours on different states of the graphs. We also provide consistency checking algorithms that take these modified graphs as input and output validation of the graphs. Program verification techniques (e.g., Hoare rules [10]) are used to provide a proof of correctness for the modified methods. The consistency checking is performed both at the inheritance level and at the method-calling level. The algorithms to check these different levels of consistency are proposed. The remainder of the paper is organised as follows. The next section discusses the related work. Section 3 describes the basic concepts related to methods in O-O schema. Section 4 gives an example. Section 5 fixes the notations and presents a formalisation of the semantics of methods. In section 6, the syntax of a method evolution language is described. Section 7 presents the strategies for behaviour consistency checking and we conclude with future work in section 8.

2 Related Work

The evolution of methods in O-O databases is a complex problem. Generally, methods are described in a procedural language in which the effects of update operations are difficult to understand. A small amount of literature has been dedicated to this problem, and generally addresses the declarative part of methods, i.e. the update of the signatures. This section surveys previous work in the area of method evolution and puts our work in context.

- In [9,18], the authors mainly approach structural evolution. Behavioural evolution is focused on the update of signatures. Behavioural consistency is restricted to run-time type errors and side-effect problems. The former are detected during structural consistency checking which ensures that the name and signature of methods are compatible along the method-inheritance chain. Data Flow technique (VDF) is used in [9] to detect run-time errors in terms of three type safe sufficient conditions. The type safe conditions guarantee
  - the inheritance relationships among data types referred by a method;
  - the declaration of method data types; and

- the matches of actual and formal arguments.

- Waller's approach [17] is graph-oriented for checking method consistency. An abstract run is introduced to represent the unfolding of the method computation from input to output. The signature of a method is "unfolded" as initial and final states of a directed graph which gives a consistency proof. When a method is changed, another consistency proof will be generated. The approach is based on incremental consistency checking which analyses the differences between two consistency proofs.

- Method Execution Tree (MET) [11] gives an intuitive way to represent the execution of method calls. The MET of a method execution is represented by the root of the tree **(m,t)[o]**, where **m** is a method name, **t** is the type that **m** is encompassed in, and **o** is an object instance of **t** used as the actual argument to invoke **m**. When **m** is executed it calls other methods and passes objects to them. Therefore, a node in the tree is a method call and an edge in the tree represents a calling relationship. The ultimate goal to use a MET is to find four kinds of problems: (1) unavoidable nontermination, (2) the method is reachable from another, (3) a method has bounded execution, and (4) method execution is aborted. The problems (1) and (3) can be detected through the configuration of its MET. Also for each method, the problems (2) and (4) are dealt with by examining a report of the method execution.

The above mentioned approaches suffer from the lack of semantics associated with methods. Especially, there are no discussions about the method correctness after method updates. However, among these approaches Waller's approach seems to be very elaborate. It has following drawbacks:

- When an abstract run is unfolding a method, some cycles may still appear in the graph which are not distinctly different from that of the recursion of methods. For example, *people – people* may represent a family tree or an *employee – manager* relationship. The unfolding algorithm runs into the problem it has tried to avoid by making the assumption that methods are recursion-free. Moreover, if the abstract runs to unfold recursive methods are unavoidable, this approach runs into a more serious problem because of the complexity of a recursive method before it is unfolded.

- Although this approach is computationally sound and complete, it still cannot deal with low level behavioural inconsistency, i.e. with instances. For example, the output of a method, say for instance an integer, may be fed into another method deriving unexpected negative values of employee salaries. The problem is rooted in the representation of the graph: the vertices are represented as (t_i, c_i, i) where t_i is a collection of method output objects, c_i is a class name, and i is the sequence number in the steps of abstract run. The checking is based on the isomorphism between the corresponding vertices and the minimal bridge between two graphs of the consistency proofs. Virtually there is still nothing more than the preventing of side-effects and run-time type errors in methods.

In this paper we provide solutions for the above problems by

- proposing a framework for representing the semantics of methods as graphs. This framework bears the information about signatures, pre- post-conditions, and relationships of methods;

- proposing the basic operations for method evolution; and

- proposing a consistency checking strategy. The checking algorithms are developed based on the two kinds of graphs and can be implemented to automatically validate the changed methods.

3 Basic Concepts

This section clarifies the basic concepts of the object paradigm such us: *overriding*, *overloading*, and *late binding*. These characteristics are fundamental to method evolution in O-O databases.

Other aspects of the object paradigm such us *static* and *dynamic typing*, and *covariance* and *contravariance* subtype rules complement the previous features. All of these aspects related to the object paradigm will be described in this section in order to provide a clear understanding of such concepts in the context of method evolution framework in O-O databases. The reader is assumed to be familiar with the O-O terminology [3].

Only a brief overview of the basic O-O concepts is given here. A *class* is a central concept in O-O systems. It is an abstraction of some information of the real world. Instances may be generated from classes. They are called *objects*. A class has attributes and a set of methods that are available to all objects of the class. Attributes and implementations of methods are hidden, and methods are used only through their signatures. The signature of a method specifies a set of input attributes and a result attribute. A method is invoked by sending a message to the object to which the method is attached. A message actually is a call to a method with actual arguments assigned to match with the method signature. In this paper we use the terms of "calling a method" or "sending a message" interchangeably. An O-O schema is a set of classes connected together with various types of relationships. The most common (used) relationships are: *inheritance* (isa) and *aggregation* (partof).

In this paper we adopt a "simplified" Booch notation [5] for describing an O-O application (see Figure 1). In [5], classes are specified as cloud dashed icons. An icon class contains the name, the attributes and the interfaces of the class. Additional relationships such as *using*, *instantiation* and *metaclass* relationships are not considered in this paper. They are generally introduced during the design, and then transformed to an *isa* or *partof* relationship during the implementation. The static part of an O-O schema, i.e. the attributes of classes, is widely addressed in the literature. However the dynamic part, i.e. the methods of classes, is generally not deeply studied. Indeed, method plays an important role in O-O systems, especially for method evolution.

This section clarifies concepts related to methods in O-O systems to prepare for further work in the area of method evolution.

Overriding, Overloading and Late Binding. Method overriding is inherent in class inheritance semantics. A method defined in the superclass can be redefined by its subclass(es). By using the same name, a number of methods may override along with an inheritance path. Method overloading reflects the meaning of the polymorphism. Different methods with the same name may be defined within the same lexical scope (e.g., a class). Method overriding and overloading give an effective way of a natural mapping between a problem space and a method space. Late binding is a way that guarantees the overriding and the overloading are properly handled [3]. During compile-time, the binding of method names can only be done in a context-free format and may not be able to link the ones that are meant to be executed. However, at run-time, a certain method to be bound is dependent on the context that

- a method of a subclass should always override the one of the superclass;
- amongst the methods with the same name in the same lexical scope, the one which matches the signature of the caller should be bound.

In other words, the late binding is a process in which the system determines the method code to be run and performs the type checking at run-time. Late binding is also known as dynamic binding or run-time binding.

Static and Dynamic Typing. Methods in O-O systems are categorised as either static typed [2] or dynamic typed. Sometimes static typing is also known as strong typing [12]. A static typed method is characterised by the fact that all types of objects referred to by the method can be determined at compile-time. Situations include:

- a call for an overloaded method such that the types of signatures need to be discriminated;
- a call for an inherited method such that the types of objects involved need to be decided;
- any call-by-reference situations in which a method or an object is to be referred to via a pointer.

In the presence of late binding, some of the static-typed languages require that all candidates

must uniformly match with that of the dispatcher (i.e., a caller of a method or a variable referring to a method). Dynamic typing in programming languages is characterised by the fact that the types referred to in a program are determined at compile-time whenever that is possible. Otherwise, they are determined at run-time. This means that during late binding, the match of types of objects is performed within the context of method execution. Whether a static-typed or a dynamic-typed programming language is used, the type safety problem for overloaded and inherited methods have some principles, namely, the covariance subtype rule [9] and the contravariance subtype rule [7,6]. Following are the definitions of such typing rules.

Covariance Subtype Rule. When a method is redefined or substituted by another method, in order to maintain the type safe method calling, the covariance subtype rule states that a method m_1 with a signature $X \longrightarrow Y$, denoted as $m_1 : (X \longrightarrow Y)$, is a subtype of a method m_2 with a signature $X' \longrightarrow Y'$ if X is a subtype of X' and Y is a subtype of Y'. Thus if m_2 is replaced by m_1 with the satisfaction of the covariance subtype rule, it is still type safe. However, when late binding is taking place, the covariance subtype rule per se cannot prevent run-time type errors. In this case, the late binding may take an object $x' \in X'$ instead of $x \in X$ to match with m_1. This is allowable by the covariance rule since X is a subtype of X' and X inherits all objects from X'. However x' may not have some attribute that is peculiar to x required by m_1. In this case, a run-time type error within m_1 would be detected.

In brief, the covariance subtype rule provides a sufficient condition to specialise a method into its subtype and problems may occur when an object from a supertype is passed into a subtype.

Example 1 *Let us consider the following example to clarify the covariance subtyping. Assume that* $X = Student$, $Y = Lecturer$, $X' = Person$ *and* $Y' = Staff$. *If* $m_2 : (Person \longrightarrow Staff)$ *is replaced by* $m_1 : (Student \longrightarrow Lecturer)$, *then all calls previously to* m_2 *now become calls to* m_1. *Since Student is a subtype of Person and Lecturer is a subtype of Staff, this replacement is acceptable for the reason that a student inherits all the attributes of the class Person. This is the same for a lecturer who is a person. Consequently, any enquires previously to a person and a staff will then be substituted by to student and lecturer. This is the nature of subtyping. However with late binding, there might be a case that an actual argument* p *which is of type Person is bound to the formal argument* s *which is of type Student. Since* p *cannot provide some specific information required by* s *(say, credit-points of a student), a run-time type error is reported. It can be seen that this is not a problem when* m_2 *is not replaced by its subtype* m_1.

Contravariance Subtype Rule. The contravariance subtype rule has been discussed by Cardelli in [7]. Given two methods m_1 and m_2, where $m_1 : (X \longrightarrow Y)$ and $m_2 : (X' \longrightarrow Y')$, m_1 isa m_2 means that X is a supertype of X' and Y is a subtype of Y'. By contravariance subtype rule, type safety is provided in late binding. In this case, m_2 may be substituted by m_1. Then, during late binding, if an object $x' \in X'$ instead of $x \in X$ is taken to match the m_1 signature $X \longrightarrow Y$, since X is a supertype of X', any object taken from X' must have all properties that $x \in X$ has, so no run-time type error could possibly happen. In brief, a method $m_2 : (X' \longrightarrow Y')$ can be generalised to $m_1 : (X \longrightarrow Y)$ when contravariance subtype rule is applied.

Example 2 *Let us consider an example where* $X = Person$ *and* $Y = Lecturer$, $X' = Student$ *and* $Y' = Staff$. *If* m_2 *is replaced by* m_1, *then all the calls previously to* m_2 *are now calls to* m_1. *Since Person is a supertype of Student and Lecturer is a subtype of Staff, this replacement is acceptable, in the sense that* m_1 *requires less information as its input type of Person and returns more information as its output type of Lecturer. Apparently, any previous calls to* m_2 *will be acceptable to* m_1 *because the signature of* m_2 *actually specifies more information than is required by the signature of* m_1. *During late binding, if either an actual argument* s *with the type Student or* p' *with the type Person is bound to the formal argument* p *with the type Person, there is no run-time type error because* s *actually provides redundant information for* p *and* p' *is compatible to* p.

Covariance vs. Contravariance. The covariance subtype and the contravariance subtype rules are

mutually exclusive. That is, if a system adopts the covariance subtyping, it will give an error whenever the redefined domain (left) part of the method signature violates this rule of the covariance subtyping. A similar situation happens to a system that adopts the contravariance subtype rule. When the redefined domain (left) part of the method signature is not a supertype of the domain it is redefined from, an error is issued. Many existing systems use only one of these rules: Eiffel for example uses the covariance rule. However in general none of the existing systems support method evolution. In fact, the covariance and the contravariance subtype rules are applied under the following circumstances:

- When a method is altered, the relationship or consistency with other methods needs to be reconfirmed.
- When a method is called in a context sensitive situation such as pointer dereferencing, overriding, or overloading, in which late binding is invoked, the method signature needs to be matched with the actual arguments.

We are concerned with a combination of these two rules in the context of the method migration. The method migration can be performed in two directions: *specialisation* and *generalisation* (section 7). The combined rule is an exclusive *or*ed of covariance and contravariance subtype rules. So, either one of the rules is allowed to be applied on a method but not both. The distinction between the use of these two rules is based on the semantics of method evolution which is detailed in section 7. By applying either the covariance rule or the contravariance rule, methods are specialised or generalised. The type safety then relies on consistency checking in the presence of late binding. This is also detailed in section 7 and a run-time exception handler for this idea is to be discussed in future work.

4 University Example

This section describes an example of an O-O application relating to a university environment (see Figure 1). This example will be used to illustrate the method alteration within an O-O schema as well as consistency checking. Booch [5] notation is used for showing the class diagram. Dashed icons represent the classes, such as *Subject* and *Course*. Every class has three types of information: the name of the class, the set of attributes (properties), and the corresponding methods that are used by the objects derived from that class. Classes are connected with different types of relationships - *aggregation* and *inheritance* relationships. The former are represented graphically as a line with a plain boulet at the beginning of the relationships. For instance, the class *Course* has *course_name*, *time*, plus a reference to *Subject* as attributes. This reference expresses the aggregation relationship that contains the class *Subject*. The other relationship, graphically expressed as a directed arrow, expresses the inheritance relationship between classes. For instance, there is an inheritance relationship between the classes *FullTimeL* and *Lecturer* which expresses the fact that a full time lecturer is a lecturer.

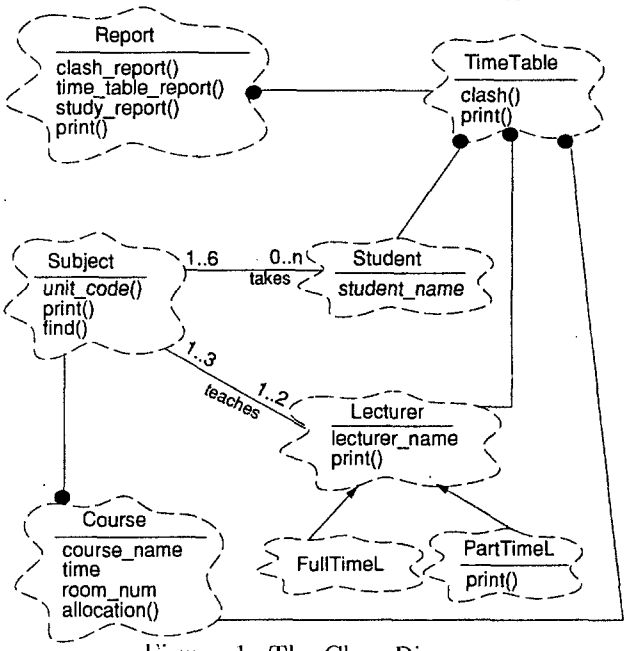


Figure 1: The Class Diagram

The above class diagram contains information that is required for design purposes. However, for the evolution of the method, other types of knowledge, such as the relationships between methods, should be declaratively specified. These kinds of relationships are called *calling relationships*, and these model the calling dependencies between methods. Calling relationships are generally parts of the implementation part of methods. In our approach we will provide additional data structures that allow the declarative specifications of such

calling relationships.

Let us consider the example of Figure 1 to show the advantages of the use of calling relationships for method evolution purposes. For instance, the method *allocation*() of the class *Course*, denoted as *Course.allocation*(), refers in its implementation to the method *Subject.find*(). This calling relationship between *Course.allocation*() and *Subject.find*() means that *Course.allocation*() may have to be modified when *Subject.find*() is changed. This relationship will be declaratively represented in appropriate data structures and then used as a basic "piece" of information during the evolution of methods.

5 Method Schema

Definitions and notations are proposed in this section to prepare the formalisation of methods. Basic definitions of graphs, class inheritance hierarchy, and method schema are given. Notations for the relationships between methods are described.

A *graph* G is defined as a tuple $< N, E >$ where N is a set of nodes and E is a set of edges. Every edge describes a relation between two nodes. A *directed graph* is a graph in which every edge is an asymmetrical relation. A *directed acyclic graph* G=$< N, E >$ is a directed graph without cycles.

The class inheritance hierarchy of an O-O schema is a directed acyclic graph $< C, E >$, where C is the set of classes and E is set of *isa* relationships between classes. Given a class c and a method m, the method m in c is identified by c.m which is called the method identifier.

A method schema describes all information related to methods of a schema. It represents the behaviour part of a schema and it is represented as a tuple in the form of $< C, M, ID, I >$ in which

- C is the set of all class names,
- M is the set of all method names,
- ID is the set of all method identifiers,
- I is the inheritance hierarchy defined over C.

The class inheritance hierarchy of an O-O schema is a directed acyclic graph <C,E>, where C is the set of classes and E is the set of isa relationships between classes. Polymorphism allows a class' method to be redefined in its subclasses. Thus given a class c and a method m, we identify the method m in c by c.m which is called a method identifier. An attribute set A is defined as a set of functions, A = F: B⟶T, where

- B is a set of attribute names.
- T is a set of all types.
- F is a set of functions that map between attribute names and types.
- A is the set of all attributes, denoted by A = {a|a = b:t, where b∈B, t∈T}.

Refinements are needed for the above notations, such as the functions that map attributes to classes and map attributes to method signatures etc. These are omitted because they are not in our focus. Square brackets [] are used to delimit the notations. Given classes c, c', and c'', and methods m, m', we denote by

[$c \uparrow c'$]: c is a subclass of c'.

[$c.m \downarrow c'.m'$]: the method m of the class c calls the method m' of the class c'. Or we may say that $c'.m'$ is expected by $c.m$.

[$c.m \longrightarrow c'.m$]: the method m in the class c is redefined from the class c', where $c \uparrow c'$ and $\nexists c''$ such that m is defined in c'', $c.m \longrightarrow c''.m$ and $c' \uparrow c''$.

[$a \preceq a'$]: a is a subtype of a', where $a = b : t$; $a' = b' : t'$; $a, a' \in A$; $b, b' \in B$; $t, t' \in T$; and t is a subtype of t'.

[$m \preceq_{cv} m'$]: method m is a subtype of method m' satisfying the covariance subtype rule.

[$m \preceq_{ct} m'$]: method m is a subtype of method m' satisfying the contravariance subtype rule.

[$m@m'$]: m is bound where m' is expected. This implies that m' is replaced by m.

[$m \circ m'$]: the message sent to m is received by m' due to the late binding. This means that the actual arguments of m are to be matched with the formal arguments of m' because m' is bound.

A method consists of two parts: the **definition part** and the **implementation part**. The former is in the form of $c.m : (S \longrightarrow t)$ where c is a class

name; m is a method of the class c; S is a list of input attributes of the method; and t is the resultant attribute of the method. The function $S \longrightarrow t$ is the signature of the method. Note that the arrow "$\longrightarrow$" has an overloaded meaning with that of the method redefinition notation in section 5. However, in the context of our discussion, there should be no ambiguity. When methods are overloaded, the same name is declared for different methods in the same lexical scope. However they are distinguished on the basis of the number and types of their arguments. When it is necessary, a system may need to refer to the signature of a method to distinguish between overloaded methods. Figure 2 describes a graphical description of a method. The implementation part is the body of the method and this is represented in the form of c.m(P,Q,R,U), where

- P is a set of attributes local to m in c and may be used as actual arguments to call other methods.
- Q is a set of constraints on the method input attributes (i.e., formal arguments). They are predicates specifying the pre-conditions before the method is executed. The variables of the predicates are taken from the signature S.
- R is a set of predicates specifying the post-conditions after the method is executed.
- U is a set of methods directly called by c.m. Elements of U can be expressed by method identifiers with the signatures which are substituted by the attributes from P or S.

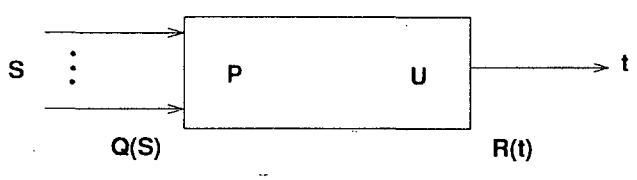


Figure 2: Graphical Representation of a Method

In an O-O schema, a class is defined within an inheritance hierarchy. A method as a property of a class may be inherited from that of its superclasses. By polymorphism, a method may also be redefined in subclasses with the same method name(s) as that of its superclass(es). Every method is involved in two orthogonal hierarchical structures: the *method definition directed acyclic graph* (DAG) and the *method dependency graph* (MDG). The former is a graph which models the inheritance relationships between methods, while the latter models the calling relationships. These graphs are defined as follows:

Definition 1 *(Method Definition)*
Given a method schema $< C, M, ID, I >$ *and a method* m *of* M*, the method definition directed acyclic graph of* m *is a graph* $DAG(m)=< C', E' >$*, where*

- $C' = \{c' | c' \in C$, *and* m *is declared in* $c'\}$
- $E' = \{e' | e' : c'.m \longrightarrow c".m$ *where* $c' \in C', c" \in C', m \in M, c'.m \in ID, c".m \in ID\}$.

When context is clear, we may mention DAG for DAG(m).

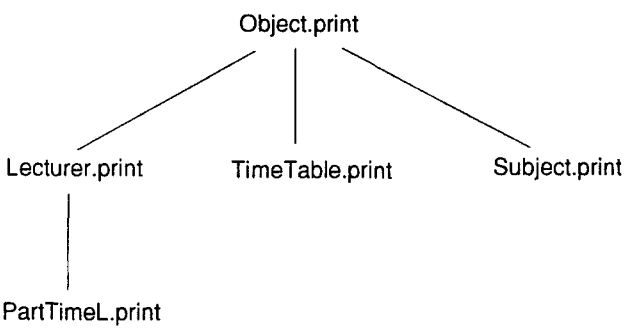


Figure 3: The Method Definition DAG of *print*

Note that a node with only incoming arrows of a DAG indicates the common definition. All other kinds of nodes of the graph indicate the redefinition of methods. Figure 3 describes the DAG of print method which has the following structure: DAG(print)=$< C', E' >$, where

- $C' = \{Object$, $Lecturer$, $TimeTable$, $Subject$,$Report\}$
- $E' = \{Lecturer.print \longrightarrow Object.print$, $Subject.print \longrightarrow Object.print$, $Report.print \longrightarrow TimeTable.print$,$TimeTable.print \longrightarrow Object.print\}$

Definition 2 *(Method Dependency Graph)*
Given a method schema $< C, M, ID, I >$*, a class* c *of* C *and a method* m *of* M*, the method dependency graph of c.m method represents the methods of* M *which are linked to c.m by the calling relationship. This graph is in the form of* $MDG(c.m)=< ID', E' >$*, where*

- $ID' = \{c'.m' | c'.m' \in ID$, *and* $c.m \downarrow c'.m'$ *or* $\exists\ c".m" \in ID'$ *such that* $c".m" \downarrow c'.m'\}$
- $E' = \{e | e : c'.m' \downarrow c".m", c' \in C, c" \in C, c'.m' \in ID', c".m" \in ID'\}$.

When context is clear, we may mention MDG for MDG(c.m).

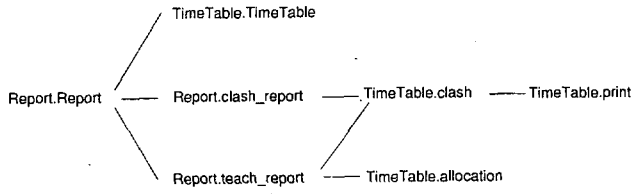


Figure 4: Method Dependency Graph of *Report.Report*

Figure 4 shows the method dependency of *Report.Report*. This is presented in the following form: MDG(Report.report) = $< ID', E' >$, where

- ID' = {*Report.Report*, *TimeTable.TimeTable*, *Report.clash_report*, *Report.teach_report*, *TimeTable.clash*, *Teach.allocation*, *TimeTable.print*}
- E' = {*Report.Report* $\downarrow$ *TimeTable.TimeTable*, *Report.Report* $\downarrow$ *Report.clash_report*, *Report.clash_report* $\downarrow$ *TimeTable.clash*, *TimeTable.clash* $\downarrow$ *TimeTable.print*, *Report.tech_report* $\downarrow$ *TimeTable.clash*, *Report.teach_report* $\downarrow$ *Teach.allocation*}.

With all the concepts introduced in this section, the semantics of a method can now be represented as a single concept: namely **method intension**. It is an integration of all the information related to a method. A method intension contains the identifier, the signature, the implementation, and the two graphs based on method definition and method dependency.

Definition 3 *(Method Intension)*
Given a method m of a class c, the semantics of c.m is defined by the tuple INT(c.m) = $< c.m : S \longrightarrow t, c.m(P,Q,R,U), DAG(m), MDG(c.m) >$ *where,*

- *c.m is the method identifier;*
- $S \longrightarrow t$ *is the signature of c.m;*
- *c.m(P,Q,R,U) is the method implementation of c.m;*
- *DAG(m) is the method definition DAG defined over c.m; and*
- *MDG(c.m) is the method dependency graph of c.m.*

The next section introduces the syntax of method-changing operations using the definitions and notations presented in this section as well as those concepts described in previous sections.

6 Syntax of the Method Evolution Language

The evolution of a method may be performed by different operations on a schema. These operations are: the deletion of a method, the insertion of a method, and the modification of a method. Our approach considers only two basic method-changing operations of deletion and insertion. The method-changing operation should be coded as a transaction to combine elementary operations into a single method-changing operation. For instance, the modification of a method is described as a transaction of deletion followed by an insertion. The moving operation is regarded as a composition of deletion and insertion operations as well. Consistency checking is done after the transaction. Within a transaction, a virtually deleted method can still be accessed. This section discusses the syntax of the method-changing operations and emphasises what can be changed rather than how it can be changed.

Method-changing operations refer only to locally defined methods rather than those references inherited via the class inheritance hierarchies. In other words, the method-changing operations can only be performed on those existing methods which can be found in DAG(m), m∈M. Otherwise, an insertion operation of method m will create a new DAG(m). For instance, according to the schema of Figure 1, the class *FullTimeL* inherits the method *print*() from the class *Lecturer* but does not modify it since the method *print* is not locally declared within the class *FullTimeL*. This aspect will be detailed later.

Bearing in mind the fact that a method can be changed by designing a transaction consisting

of only two basic method-changing operations, a method can be changed by the following operations:

A. Deletion

1. Deletion of a class: the class is deleted and all its methods are implicitly deleted as well.
2. Deletion of a method: the method is deleted explicitly from a class.

B. Insertion

1. Insertion of a method with a new class.
2. Insertion of a method within an existing class.

C. Modification

1. Modification of the name or the signature of a method.
2. Modification of the implementation of a method.
3. Modification of both the definition and the implementation of a method.

The proposed set of operations may be reduced to a minimal set. For instance, a method can be deleted explicitly from a class or deleted implicitly due to the deletion of a class. The implicit deletion is sequenced within a transaction into two distinct operations: the deletion of all methods defined within the class, and the deletion of the class. Similarly, the insertion of a method may occur with two distinct operations: the insertion of a new class followed by the insertion of the method. Since the structural evolution of a schema is not addressed in this paper, the insertion and the deletion of a class will not be discussed. Consequently, the set of method-changing operations is reduced to the situations A-2, B-2, C-1 and C-2. The syntax of the method evolution language is the following.

```
method_changing_operation
   ::= <deletion>|<insertion>|<modification>
deletion
   ::= delete_method (<method_identifier>)
method_identifier
   ::= <c>.<m>
c  ::= <class name where m is defined>
m  ::= <name of the method>
insertion
   ::= insert_method (<method_definition>,
       <method_implementation>)
method_definition
   ::= <method_identifier>
       [:<method_signature>]
method_signature
   ::=  <input_type> -> <output_type>
method_implementation
   ::= <method_identifier> <method_body>
method_body
   ::= ([<P>,<Q>,<R>,<U>])
P  ::=  {<local_variables>}
Q  ::=  {<pre-conditions>}
R  ::=  {<post-conditions>}
U  ::=  {<method_identifiers>}
modification
   ::=  <modify_definition>|
        <modify_implementation>|
        <modify_method>
<modify_definition>
    ::=  modify_def ([<method_definition>],
         <method_definition>)
<modify_implementation>
    ::= modify_imp ([<method_implementation>],
        <method_implementation>)
<modify_method>
    ::= modify_mtd ([<method>],<method>)
<method>
    ::= (<method_definition>[,<method_body>])
```

The <modify_method> may be seen as a generic operation to <modify_def> and <modify_imp>. The <modify_method> can also be used as a generic method for moving and copying operations. Using the above syntax for method evolution, a method $c.m$ may be updated as follows:

Insertion

- syntax: $insert\_method(c.m : (S \longrightarrow t), c'.m())$
- semantic: This operation inserts m in c. The body of m is copied from the method with the same name in c'.
- example: $insert\_method(Lecturer.find : (c : char \longrightarrow find : char), Subject.find())$

Deletion

- syntax: $delete\_method(c.m)$
- semantic: This operation deletes the method m from the class c.
- example: $delete\_method(Course.allocation)$

Modification of the name

- syntax: $modify\_def(c.m, c.m')$
- semantic: This operation changes the name m into m'. This is interpreted as a transaction which deletes m and is followed by the insertion of m'. Since the modification operation is a transaction, the deleted m remains accessible from m'. The new method m' has the same signature and the same body as m.
- example:
  modify_def(Subject.find, Subject.search)

Modification of the signature

- syntax: $modify\_def(, c.m : (S' \longrightarrow t'))$
- semantic: This operation is regarded as a transaction which firstly deletes $c.m$. Then, a method is inserted with the same name and with the specified signature. When the context is clear, a comma in the command indicates that the method before the comma has the same identifier as the method after the comma.
- example:
  modify_def(, Report.Report:(S⟶t))
  The details of S and t ought to be given.

Modification of the body

- syntax:
  modify_imp(c.m, c.m(P',Q',R',U'))
- semantic: This operation modifies the body of $c.m$.
- example:
  $modify\_imp(Student.print,$
  $Student.print(P', Q', R', U'))$
  The details of P', Q', R', U' ought to be given in practice.

The proposed method-changing operations are arbitrary and may possibly result in compile-time or run time errors. These errors can be detected or handled at various stages by a recompilation trigger mechanism or by a run-time exception handler. The focus of this paper will be on the development of algorithms that are applicable to these stages, in order to produce methods that are type safe and behaviourally consistent.

Informally speaking, run-time errors are those situations that can disrupt the execution of methods. Some run-time errors are machine-implementation dependent. In this case, the run-time errors report abnormal situations of the program execution environment such as arithmetic overflows, memory, disk, communication, and file management failures. However, due to overriding, overloading, and later binding, many run-time errors are caused by method updates. These include subtype mismatches during late binding, violations of domain constraints, or some abnormal arithmetic errors. The major problem to be dealt with is type mismatch errors during run-time. The pre-detection of the run-time type errors is not a trivial problem.

Additionally, methods should be defined as side-effect free. In fact, the specifications of method pre/post conditions give a means to ensure that the expected behaviours are performed at the implementation stage. When a method is not side-effect free, a global variable, or an object in the database, may be changed by the method without a passing of method arguments. In our approach, it is important to the evolution handling mechanism to maintain side-effect free methods, especially when a method is subject to intensive calls or massive data updates in a database.

7 Behavioural Consistency

This section develops a mechanism for handling method-changing operations and consistency checking. For that, the concept of **method scope** is introduced. This is defined as a set of integrated graphs and a set of method intensions. The method scope represents all the information about the methods contained in an O-O schema. Algorithms for building the method scope are given and strategies for method consistency checking based on the method scope are described.

7.1 The Semantic of Method Changing

The consistency of method behaviour can be seen as the possibility of calling the method and obtaining the expected results. Inconsistent situa-

tions are run-time type errors, side-effects, redundant methods, and unexpected behaviours, such as (1) - (4) described in section 1. In the following subsections, we discuss various aspects of the semantics of method changing. In subsection 7.1.1, method migration, as a special case of method changing, is presented. The type safety in this case is provided by the migration rules. Subsection 7.1.2 develops the behavioural consistency.

7.1.1 Method Migration

When a method is changed using the operators discussed in section 6, it can happen in many different ways. One common situation is that methods may be migrated between super and subclasses. This is called migration by either specialisation or generalisation. The operation of the method migration may be constructed as a transaction of a deletion followed by an insertion. With the method migration, the type-safety problem concerns that: (1) the signature of a method may be changed; (2) a message may have an unexpected receiver i.e., a method from its super/subclass is bound. Another issue regarding the behaviour of migrated methods, such as the consistency of modified pre/post conditions, will be covered in the following two subsections.

Two rules, covariance subtype rule and contravariance subtype rule, play an important role in the checking of type safety. Based on these two rules, we shall examine two specific late binding situations.

1. **Method generalisation**: this means that a method with the supertype of its input attributes is used where its subtype is expected. For example, a method m_1 : (*Person* $\longrightarrow$ *Lecturer*) may be used where a method m_2 : (*Student* $\longrightarrow$ *Person*) is expected. Here *Student* is generalised to *Person* and *Student* is surely acceptable to *Person* because at run-time, type *Student* can always be safely generalised to type *Person*. However, the result type *Person* is changed to the type Lecturer. This substitution satisfies the contravariance subtype rule. Fig.5 illustrates this idea. The dashed arrows indicate the subtype relations and the curved solid arrow indicates that the method m_2 is generalised to the method m_1.

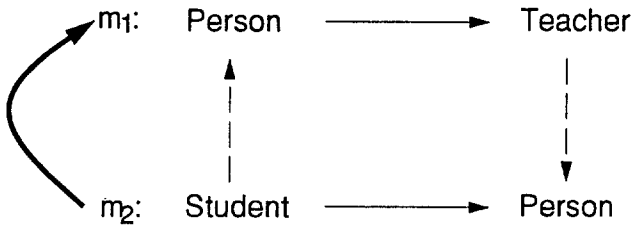


Figure 5: Method Generalisation

2. **Method specialisation**: this means that a method with the subtype of its input attributes is used where a supertype is expected. For example, a method m_3 : (*Student* $\longrightarrow$ *Student*) may be used where a method m_4 : (*Person* $\longrightarrow$ *Person*) is expected. In this case, *Person* is specialised to *Student*. Surely, at run time, only messages sent to the object with m_3 can be bound with type safety. When the late binding mechanism tries to bind a message sent to the object with m_4, a run-time type error (*Person* does not match with *Student*) is detected. This is illustrated by Figure 6. The dashed arrows indicate the subtype relations and the curved solid arrow indicates that the method m_4 is specialised to the method m_3.

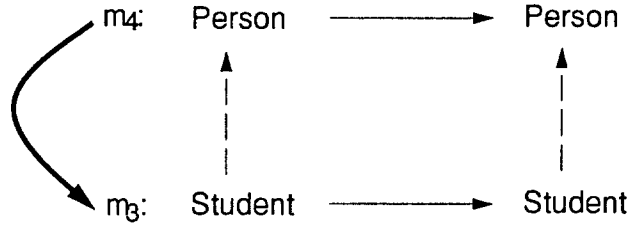


Figure 6: Method Specialisation

Updating a method either by generalisation or specialisation offers a general framework for method migration as well as for type safety. The following definition completes the previous discussions by integrating the specialisation and the generalisation rules.

Definition 4 *(Migration Rules)*
In summary, the migration via generalisation or specialisation is said to be type safe if

- *a method is generalised, or*
- *a specialised method is not to be bound to the method of its supertype.*

7.1.2 Behavioural Consistency

In addition to the type safety problem, the other issue is behavioural consistency which is a fundamental aspect of method evolution. This relates to the capability to produce correct behaviour applications in the sense that they reflect the assumptions of the real world. However the properties of behavioural consistency are difficult to identify and views on consistency have been given by many other works. The type safety problem can be regarded as a sub-issue of consistency. We now give a definition of behaviour consistency by taking into account all of the modifications of every part of a method.

Definition 5 *(Behavioural Consistency)*
An O-O database schema is behaviourally consistent if and only if it satisfies the following conditions:

- *The behaviour of a method* m *in a class* c *is prescribed by* c.m:(S⟶t) *and* c.m(P,Q,R,U).
- *The relationships among methods are described in* DAGs *and* MDGs.
- *There are no such consequences caused directly by the method-changing operations, as*
  - *run-time type errors*
  - *side-effects*
  - *redundant methods, i.e. methods that have the same signature and the same implementation*
  - *unexpected behaviour, i.e. pre/post conditions cannot be satisfied.*

In a given method schema $H = < C, M, ID, I >$, a method c.m∈ID with c.m:(S⟶t) and c.m(P,Q,R,U) as the definition and the implementation parts, can be updated using the proposed method-changing operations. The principles of behavioural consistency checking are that:

1. Any arbitrary method changing is allowed.
2. The consistency checking is a decidable problem if the method intensions {INT(c.m)| c.m∈ID} are given. This is mainly because of the following:
   - All INTs have limited sizes.
   - All effects of method-changing operations are reflected in INTs.
   - The effects of method-changing operations have finite states.
   - The checking algorithms are designed based on the searches and comparisons of INTs' states.

   Thus the method consistency checking problem can be abstracted as a problem space that has a limited size, with finite states, and a set of search algorithms.
3. When a method c.m is changed, $INT(c.m)$ becomes $INT'(c.m')$ which may be empty if $c.m$ is deleted.
4. The checking criteria are used to check the following situations:
   - the signature of $c.m$ is type-consistent within $MDG(c.m)$.
   - the signature of $c.m$ is consistent with regards to the redefined methods within $DAG(m)$.
   - there are no two methods $c'.m'$ and $c".m"$ such that they have identical signatures and pre/post conditions.
   - there is no such method intension $INT(c".m")$ such that the pre/post condition $P"$ and $Q"$ of $c".m"$ are false in any case of calls within the graphs.
   - all the $c.m(P, Q, R, U)$ in $INT(c.m)$ are checked using $MDG(c.m)$. The principle of the checking is to adopt the Hoare rules [10] to check method correctness and method calling consistency, and to satisfy the matches between the formal and actual arguments (i.e., the substitution of signatures by the input or local attributes for the called methods). The migration rules are to be checked and the following rule is also used to check the signatures:

     if $c.m \downarrow c'.m' \in MDG(c.m)$
     then $\forall c'.m' \in U, U \subseteq ID,\ S' \cup \{t'\} \subseteq P \cup S \cup \{t\} \wedge R \supseteq Q'$
   - if $c.m$ is deleted, then $INT(c.m)$ should be deleted too. For any other $INT"(c".m")$, where $c.m$ appears either

in $DAG(m")$ or in $MDG(c".m")$, $c.m$ should be deleted from the $DAG(m")$ and the $MDG(c".m")$. Then, the consequent deletions are checked in terms of the above mentioned criteria.

Given a class c and a method m of that class, a scenario of the method-changing and the behaviour consistency checking is that

- a method-changing operation is defined on $c.m.$;
- the consistency checking procedure uses $INT(c.m)$ and other related $INTs$;
- a method-changing operation is rejected if either errors are found or a warning message is issued with the acceptance of the operation;
- an audit report on the operation is given.

In section 7.2 we develop appropriate data structures for handling method evolution. This is called *method scope*, and algorithms for consistency checking are provided in section 7.3.

7.2 Method Scope

In order to check inconsistency, method intensions which carry the semantics of methods including the changes on the methods are used as the basis of checking. The checking procedure may refer to many different method intensions for deciding consistencies amongst methods. Indeed, a global view on methods is needed to optimise the checking strategies. For that reason, the concept of method scope as the integration of all method intensions is introduced. Since there may exist several method intensions such that they all share the same DAG, DAGs are integrated as a global DAG(M) in the method scope. On the other hand, MDGs are integrated as MDG(ID) because an MDG may be a subgraph of some others. MDG(ID) is not necessarily a connected graph since there may be some methods which do not call each other at all. The connectivity of MDG(ID) is not an important issue since it is not necessary that a method must call another one within a schema.

Definition 6 *(Method Scope)*
The method scope of a method schema <C,M,ID, I> is a tuple which consists of integrated graphs and method intensions. It is defined as SCOPE = <C,M,ID,DAG(M),MDG(ID),INT> where,

- $DAG(M) = \bigcup_{m \in M} DAG(m)$
- $MDG(ID) = \bigcup_{c.m \in ID} MDG(c.m)$
- $INT = \{INT(c.m) | c.m \in ID, c \in C, m \in M\}$

We may treat the SCOPE as a generic class. The algorithm we proposed (*algorithm*1) builds up the instances of SCOPE gradually with the development of a method schema. Initially, the SCOPE is empty. Then, when a method is added to a schema, an INT(c.m) is created in the SCOPE for $c.m$. When a method is deleted, the INT(c.m) will be deleted from the INT and the SCOPE is modified accordingly. The operations on the SCOPE should be in transactions and be coupled with consistency checking. When a check fails, the transaction should roll back. Particularly, when

- m is added in c, then m is used to search for the $DAG(m)$. If there is no $DAG(m)$ found, then a new $DAG(m)$ is created. Otherwise, the $c.m$ will be added into a $DAG(m)$ according to the position of c in its inheritance hierarchy. Consequently, $DAG(M)$ is also modified.
- m is added in c, since every method is also associated with an MDG, $MDG(c.m)$ will be created and $MDG(ID)$ consequently updated. This process of building $MDG(ID)$ follows a bottom-up pattern, which means that every method must be in existence before it can be called.
- $c.m$ is dropped from a schema, the corresponding $INT(c.m)$ including $DAG(m)$ should also be modified or deleted.

The maintenance of the method scope is coupled with method-changing operations. The above algorithm is incorporated with the consistency checking algorithms 2, 3, 4, and 5. Considering the complexity of the algorithm, it is in $O(n^2)$ where n is the size of the problem space. By evaluating the size of DAG(M) and MDG(ID), n can be decided as the product of the number of classes multiplied by the average number of methods and then multiplied by the average number of

attributes of a method. Since the algorithms discussed in this section are all basically search algorithms based on directed graphs, the complexity analysis for the each of the following algorithms is exactly same as $O(n^2)$.

7.3 Semantic Consistency Checking

Unexpected behaviours of methods are the activities performed by methods which are not specified by the method declarations. The update of one method may cause changes in the behaviour of other methods. It is crucial to the checking of behavioural consistency that each method is guarded by its signature and its pre/post conditions. However, since a method may call other methods via a class inheritance hierarchy or just from other classes, the relationships amongst methods need to be constructed. In order to achieve that, the following information about a method is needed:

- the signature,
- the pre/post conditions,
- the local attributes that may be passed as actual arguments to the called methods in the MDG, and
- the identifiers of methods called.

To ensure behavioural consistency, a **two-level checking strategy** is proposed. The first level of checking concerns the prevention of run-time type errors and side effects. This is performed mainly on DAGs. The second level of checking relates to the unexpected and redundant behaviours of methods. This is performed mainly on MDGs. These levels are complete in the sense that they involve all possible situations of the consistency. Here we describe these two levels in terms of the method-changing operations.

7.3.1 Method Insertion

When a method m is inserted in a class c, the two levels of checking will be carried out. For the first level, INT(c.m) is created and used to check the consistency of the names and signatures of m. We informally describe the algorithm for this level, i.e. *algorithm 2*. This algorithm checks the following conditions:

% Algorithm 2 (Level-1 Insertion Consistency Checking on DAGs) %

1. c exists in the schema; In the case of overriding, methods are distinguished by the method identifiers with different class names. In the case of overloading, methods are distinguished by the method identifier plus the signatures. This implies that an INT(c.m) may represent a group of methods which are overloaded.
2. There is no c'.m$\longrightarrow$c.m which exists in DAG(m) such that c'$\uparrow$c$\in$I. This is to ensure that there is no method redefined from c.m before c.m is inserted. It can be seen that the acyclicity of DAG(m) is also ensured by this condition.
3. The signature of c.m:$(S \longrightarrow t)$ is compatible with those methods in DAG(m). The compatibility of signatures is defined as follows:

   $\forall$c.m$\longrightarrow$c'.m $\in$ DAG(m), such that c$\uparrow$c' and c.m:$(S \longrightarrow t)$, c'.m:$(S' \longrightarrow t')$

   If $S = s_1 \times s_2 \times s_3 \times ... \times s_j$ and $S' = s'_1 \times s'_2 \times s'_3 \times ... \times s'_k$

   then

   (i) $s_1 \preceq s'_1, s_2 \preceq s'_2, s_3 \preceq s'_3, ..., s_j \preceq s'_k, t \preceq t'$ for j=k,

   (ii) $s_1 \preceq s'_1, s_2 \preceq s'_2, s_3 \preceq s'_3, ..., s_j \preceq s'_j, t \preceq t'$ for $j < k$,

   (iii) $s_1 \preceq s'_1, s_2 \preceq s'_2, s_3 \preceq s'_3, ..., s_k \preceq s'_k, t \preceq t'$ for $j > k$.

   This definition actually defines the common part of signatures to be checked. The proposed definition is composed of two rules, namely, the covariance subtype rule and the contravariance subtype rule. We check these two rules as one case because we are doing two-level checking and any violation on subtyping or mismatch on signatures will be checked later. Particularly, for the covariance subtype rule, which is regarded as situations (i) and (ii) above (i.e., $S \preceq S'$), the match with $s'_{j+1}, ..., s'_k$ will be checked later at the second level where the MDG(c.m) is checked. For the contravariance subtype rule which is regarded as situations (i) and (iii) above (i.e., $S' \preceq S$), the match with $s_{k+1}, ..., s_j$ will be

checked later at the second level where the MDG(c.m) is checked. The advantages here are that it allows arbitrary changes to be applied on the signatures of a redefined method. Especially in the case of multiple inheritance, a method may be redefined from that of more than one superclass. In this case, the signature of the redefined method may only match with the common part of that of its superclasses and the non-common part is left to be checked later within the method-calling chains. This means that the violation of either the covariance subtype rule or the contravariance subtype rule in the non-common part of the signatures is not a problem until the method is to be called. The bottom line is to support the overriding, overloading, and later binding semantics. A method can be called by many different methods in many different situations (e.g., a method may be used to replace one which is to be deleted in the DAG, i.e., method migration).

4. There is no redundant method in SCOPE. For a given INT($c.m$), any INT($c'.m'$) in SCOPE is checked in order to find:
   - identical structures of DAG(m) with DAG(m'), or MDG(m) with MDG(m');
   - identical signatures of $c.m : (S \longrightarrow t)$ with that of $c'.m' : (S' \longrightarrow t')$, i.e., $S = S'$ and $t = t'$, and identical pre/post conditions $P = P'$, $Q = Q'$.

   The insertion of a redundant method will be warned.

For the second level of consistency checking, *algorithm 3* is developed to check the implementation part of the method INT(c.m). These are the different steps of the algorithm:

% Algorithm 3 (Level-2 Insertion Consistency Checking on MDGs) %

1. For every c'.m' involved in MDG(c.m), INT(c'.m') is defined.
2. For any c".m"↓c'.m' ∈ MDG(c.m), with the method intension INT(c".m") and INT(c'.m') respectively, ensure that S'∪{t'}∪ $P"$ ∪ $S"$ ∪ {$t"$} and check if $R" \supseteq Q'$. Then, method migration rules (section 7.1) are checked. After that, Hoare inductive assertion techniques [10] should be applied here to prove the partial correctness of the method intensions in INT. This is to ensure that all post-conditions in a method-calling chain are true, derived from the signatures and preconditions of calling methods. The violation of the covariance subtype rule and the contravariance subtype rule is dealt with here.

7.3.2 Method Deletion

When the deletion of a method m from class c is requested, as for the insertion of a method, the two-level checking is carried out. At the first level of checking, we develop an algorithm (*algorithm 4*) that checks the following aspects:

% Algorithm 4 (Level-1 Deletion Consistency Checking on DAGs) %

1. c.m should be deleted from DAG(m);
2. either drops or reconnects the rest of nodes in the graph. In particular, all the successors of c.m may need to be reconnected with the predecessor(s) of c.m;
3. checks the compatibility of signatures between the reconnected nodes.

At the second level, we develop an algorithm (*algorithm 5*) that modifies the MDG according to the methods which call c.m or are called by c.m. Since MDG(c.m) disappears with the deletion of INT(c.m), MDG(ID) needs to be modified. Algorithm 5 ensures that all references to $c.m$ in MDG(ID) should be modified. This occurs in two steps: (1) the modification of the caller methods of c.m, and (2) the modification of the called methods of c.m.

The following details the different steps of *algorithm 5* in which the modifications of the caller methods and the called methods are described.

% Algorithm 5 (Level-2 deletion Consistency Checking on MDGs) %

A) The Modification of the Callers

The modification of the caller methods of c.m regards the set of $\{INT(c'.m') | c'.m' \downarrow c.m, c.m \in$

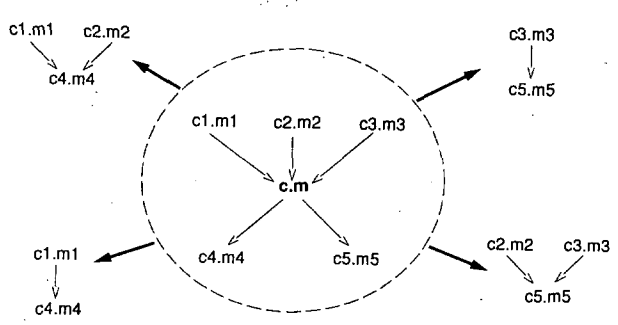


Figure 7: Method Deletion

$ID, c'.m' \in ID\}$. There are two cases for modifying caller methods. In the first case, the c.m may be replaced by a method in the DAG(m). If this is not possible, then the alternative is to try the MDG(c.m) to determine a method to replace c.m. We give details as follows.

1. If c.m is not a root node of MDG(ID), all methods previously calling c.m now have to call the successor of c.m in the DAG(m) according to c in the inheritance path. This implies that the signatures of all the callers of c.m have to be checked against that of the successor of c.m in DAG(m). Assume for example that *Temporary* and *Temporary_Lecturer* are classes where *Temporary_Lecturer* is a subclass of *Lecturer* and *Temporary*. In this case if *Temporary_Lecturer.print* is deleted, then *Lecturer.allocation* may have to call *Lecturer.print* or *Temporary.print* according to the checking of the signature compatibility and the resolution of the multiple inheritance conflict.

2. After the deletion of c.m, if no replacement of c.m can be found from DAG(m) for the callers of c.m, the method intensions of all the callers of c.m will have two choices: (a) the reference (call) to c.m is replaced by a successor of c.m in MDG(c.m), or (b) the reference to c.m is deleted from the caller's method intension. This decision is made through a user's interaction or by a predefined constraint. Otherwise, the deletion should be rejected. Assume a method c.m is to be deleted and there is no substitution that can be found from DAG(m), then a situation may be left for the user to make a decision (see Figure 7).

Figure 7 can be read as that: if c.m is deleted, then the modification on the MDG is arbitrary, decided by a user or by some predefined constraints.

B) Modification of the Called

Now, let us consider the methods that are called by c.m, i.e. c.m is a root node or has successors in the MDG(ID). The modification on the called method of c.m regards the set of $\{INT(c'.m')|c.m \downarrow c'.m', c.m \in ID, c'.m' \in ID\}$. There are three ways to maintain the method scope: cascade, nullified, and restricted.

1. **Cascade.** If there is no other caller to the successors of c.m, then all the method intensions of c.m successors will be deleted together with INT(c.m) unless there are some predefined constraints preventing this (e.g., if c.m$\downarrow$c'.m' and c$\neq$c' (c.m$\in$ID, c'.m'$\in$ID, c$\in$C, c'$\in$C) then c'.m' cannot be deleted). In this case, a recursive checking for all successors will be carried out.

2. **Nullified.** Only INT(c.m) is deleted and the set $\{INT(c'.m')|c'.m' \downarrow c.m, c.m \in ID, c'.m' \in ID\}$ is modified with $c'.m' \downarrow c.m$ being deleted. This leaves the set $\{INT(c'.m')|c.m \downarrow c'.m', c.m \in ID, c'.m' \in ID\}$ untouched.

3. **Restricted.** The deletion of c.m will be rejected if there is any caller of c.m. In this case, the effect on the method scope is minimum. Again, the decision of restricted deletion is made through an interactive session with a user or it is predefined as a constraint.

7.3.3 Method Modification

Method modification is a transaction of deleting and inserting operations. Thus the checking of the behaviour consistency can be different from that of insertion or deletion only operations.

Within a transaction, a method is virtually deleted and then inserted with changed parts. The consistency checking is therefore made in two

steps. In the first step, the checking is made on deletion: the reaction of the checking will be suppressed. In the second step, the checking on insertion is made and the reaction of the checking is also suppressed. When the transaction is completed, all suppressed checking reactions then will be synthesised. A method-changing transaction may roll back if the following conditions are not satisfied:

- DAGs are still acyclical.
- There is no redundant method.
- To prevent the run-time errors and side-effects, signatures of methods in method definition DAGs are checked for compatibility.
- MDGs involved in the transaction are examined in terms of the linkages between pre/post conditions and there is no *false* returned.

8 Conclusion

In this paper we addressed the development of a framework for the evolution methods in object-oriented databases. This involves

- the definition of the set of update operations;
- the development of appropriate structures to represent the semantic of methods; and
- an approach and algorithms for consistency checking.

Data structures, such as DAG and MDG, that model the semantics of inheritance and calling relationships are proposed. These structures are used to check the consistency of a schema after method updates.

However our approach lacks a better incorporation with the transaction model. In our approach, if an update of a method occurs and causes some inconsistent database states, then we automatically roll back the transaction. We think that a better facility, such as *compensation transactions*, will improve our approach and put it in a real development environment.

Acknowledgment

The authors gratefully acknowledge the comments provided by the anonymous referees which helped improving the readability of the paper.

References

[1] Abiteboul, S., Kanellakis, P., and Waller, E.: *Method Schemas.* Proc. of the ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, Nashville, pp. 16-27, 1990.

[2] Agrawal, R., DeMichiel, L.G., and Lindasay, B.G.: *Static Type checking of Multi-Methods.* Proc. of the Int. Conf. on Object-Oriented Programming Systems, Languages and Applications, pp. 113-128, 1991.

[3] Atkinson M., Bancilhon F., et al: *The Object-Oriented Database System Manifesto.* Proc. of the Int. Conf. on Deductive and Object-Oriented Databases, Kyoto, pp. 40-56, 1989.

[4] Banerjee J.W., Kim W., Kim H-J. and Korth H.F.: *Semantics and Implementation of Schema Evolution in Object-Oriented Databases.* Proc. of the Int. Conf. on Management of Data, San Francisco, pp. 311-322, 1987.

[5] Booch, G.: *Object-Oriented Analysis and Design With Applicants.* 2nd Edition, Addison-Wesley, 1994.

[6] Canniing P.S., Cook W.R., Hill W.L., and Olthoff W.G. *Interfaces for Strongly-Typed Object-Oriented Programming*, Proc. Object-Oriented Programming: Systems, Languages and Applications, pp. 457-467, 1989.

[7] Cardelli, L.: *A Semantics of Multiple Inheritance.* Information and Computation, Vol. 76, Academic Press, pp. 138-164, 1988.

[8] Clamen, S.: *Schema Evolution Support Summary.* Electronic news (group OBJECTS) in computer network, posted by clamen+cs.CMU.EDU, 1992.

[9] Coen-Porisini, A., Lavazza, L. and Zicari, R.: *The ESSE Project: An Overview.* Proc. of the Far East Workshop on Future Database Systems, Kyoto, Vol. 3, pp. 28-37, 1992.

[10] Hoare, C.A.R.: *An Axiomatic Basis for Computer Programming.* Communications of the ACM, Vol. 12, pp. 576-583, Oct. 1969.

[11] Hull, R., Katsumi, T., and Yoshikawa, M.: *Behavior Analysis of Object- Oriented Databases: Method Structure, Execution Trees, and Reachability.* Proc. of the FODO, Lecture Notes in Computer Science, No. 367, Springer-Verlag, pp. 3 72-388, 1989.

[12] Meyer B., *Ensuring Strong Typing in an O-O Language.* Proc. of the Int. Conf. on Object-Oriented Programming Systems, Languages and Applications, 1992.

[13] Monk, S. and Sommerville, I.: *Schema Evolution in ooDBs Using Class Versioning.* SIGMOD Record, 22(3), pp. 16-22, 1993.

[14] Penney, D.J. and Stein, J.: *Class Modification in the Gemstone Object-Oriented DBMS.* Proc. of the Int. Conf. on Object-Oriented Programming Systems, Languages and Applications, Florida, pp. 111-117, 1987.

[15] Roddick, J.F.: *Schema Evolution in Database Systems - An Annotated Bibliography.* SIGMOD Record, 21(4), 1992.

[16] Skarra A.H. and Zdonik S.B.: *Type Evolution in an Object-Oriented Database.* In Research Directions in Object-Oriented Programming, B. Shriver and P. Wegner (eds), MIT Press, 1987.

[17] Waller E.: *Schema Updates and Consistency.* Proc. of the Int. Conf. Deductive and Object-Oriented Databases, 1991.

[18] Zicari, R.: *A Framework for Schema Updates in an Object-Oriented Database System.* Chapter 7, in *Building an Object-Oriented Database System: the Story of O2*, Bancilhon F., Delobel C., Kenellakis P., eds), Morgan Kaufmann, 1991.

INFORMATIONAL BEING-OF

Anton P. Železnikar
Volaričeva ulica 8, 61111 Ljubljana, Slovenia
a.p.zeleznikar@ijs.si

Keywords: axioms, Being-of, functional composition and decomposition, function, includedness, informational, informational frame and gestalt, metaphysical gestalts, metaphysicalism, nested functional forms; serial, parallel, circular and metaphysical functionality

Edited by: V. Fomichov
Received: April 5, 1994 **Revised:** August 5, 1994 **Accepted:** September 5, 1994

Informational Being-of is another fundamental informational concept of functionality in comparison with the informational includedness studied in [9]. It has its formal-theoretical informational structure which is recursive, circular and spontaneous. Informational Being-of can be studied in many aspects from which we chose basic axioms concerning informational functionality, informational interpretations of formula $\varphi \models_{of} \alpha$, *and phenomena of serial, parallel, circular informational functionality. Some advanced problems of decomposition (destruction) and composition (construction) concerning informational functionality are treated. At the end, informational functionality of metaphysical cycles impacted by an exterior entity is studied and the so-called metaphysical gestalts concerning the informational Being-of are introduced. Informational gestalts reveal several problems of informational formula structuring, functional interdependence and the like.*

1 Introduction

Informational Being-of is the original term coined in this paper[1]. In the common speech we say that something *is of* something or *is* something's something, for example, also in the context, to *be* a property (a definite something) *of* something, an information *of* information, in general, etc. Further meaning can be deduced from that which is comprehended as formal functionality (being a function of something), where the function of the function's argument is coming into presence.

Informational Being-of concerns the so-called functionality of informational entities, that is explicit and implicit formulas of the kind $\varphi(\alpha)$ and $\varphi \models_{of} \alpha$, respectively, being informational operands within the informational theory [4]. Informational function is the basic and one of the most powerful informational concepts which enables the active informational role of an entity upon another (passive or active) entity. In this respect, formula $\varphi(\alpha)$ has to be informationally determined in an informationally recursive and arising manner, as a regularly informing operand. Expression $\varphi(\alpha)$ symbolizes a system of informational formulas in which several operands can depend on argument α, for instance in the filled metaphysical shell belonging to an entity. Definition 1 is the basic concept determining the informational function as a fundamental item of the informational theory. In the last consequence (in a concrete situation), formulas depending on α are explained (informationally interpreted) by formulas, in which α appears explicitly as an operand and not in the functional form (operand-operator-parenthesis formula).

Informational Being-of is in several informational-theoretical aspects a parallel and complementary construction to informational Being-in [9]. It can in several respects be more complex than informational Being-in. Both give to the informational theory the power of extremely complex functionality where active, passive, and active-passive entities can be distinguished in a

transparent way. The question is, for example, what kind of functionality does the informational includedness (which is a synonym for informational Being-in) perform. Thus, informational includedness can also be studied from this point of view, that is, to be understood as a particular case of the informational Being-of. We will show how such a view is righteous and has its roots in the basic philosophy of an arising informational theory.

2 An Initial Philosophy of Informational Being-of

To be something *of* something pertains to the meaning of the *of* as described, for example, in [10], where it is written, among other meanings, that from its original sense, *of* was used in the expression of the notions of removal, separation, privation, derivation, origin, starting-point, spring of action, cause, agent, instrument, material, and other senses, which involve the notion of talking, coming, arising, or resulting *from*. *Of* means from, away from; also down of, up of, and off of when following an adverb, with which it is sometimes closely connected. *Of* indicates a point of time from which something begins or proceeds, the emergence out of which something is formed. *Of* is used in certain phrases, which particularize the meaning, as within of, wide of, back of, backwards of, etc. It expresses a property, possession or appurtenance.

As an informational operator, the Of is connected with verbs (operational compositions, see [7]), e.g., to recover, deliver, empty, free, rid of, etc. The Of introduces that (φ as a function) which is removed (deduced, inferred) from something (α as a functional argument). There are functional relations (informational transitions) between the maker (argument α and the impacting environment), its making (informing) and the made (function φ as $\varphi(\alpha)$). The Of expresses racial or local origin, descent, etc. after the verbs arise, be, come, descend, spring, be born, bred, propagated, and the like. In informational language, we already use informational arising of, being of something and, further, coming into existence of and from, etc. The Of connects notions of origin (cause, maker, generator) and consequence (the made, result), where φ is a consequence of α (and, possibly, other entities) in $\varphi(\alpha)$.

Metaphysical sense of the Of concerns oneself, something informing by one's own impetus or motion, that is, spontaneously, without instigation or aid of another or together with another entity (this is the so-called metaphysical environment of an informational entity). The Of indicates the cause, reason (in reasoning), or ground (in understanding) of an intelligently acting, occurring, informing, sensing entity, etc. It points to the informational agent or doer (e.g. the subjective genitive is a form $\varphi(\alpha)$ with the direct speech equivalent α's φ, that is φ of α).

The Of has a significant role in interpretation when a transformation is expressed from a former (origin) situation into a new interpretation (cause-consequence). There are numerous phrases in which the Of is figuring as a functional transition between entities, where it indicates the subject-matter of thought, attitude, or action. Thus, it figures in the sense: concerning, about, with regard to, in reference to, etc.

In regard to an informing entity α, α's informing $\mathcal{I}_\alpha$ is nothing else than a function of α, that is, $\mathcal{I}(\alpha)$. In this sense we use $\mathcal{I}_\alpha$ to enable a direct expression of the $\mathcal{I}_\alpha$'s functionality, for example, $\mathcal{I}_\alpha(\beta)$, in which $\mathcal{I}_\alpha$ is a function of β. A direct functional expression in this case would be $\mathcal{I}(\alpha)(\beta)$. Thus, we must remind that the parenthetical sequence ')(' in a formula is nothing other than a functional connection (a kind of functional product, marked by ')*(', for example). Instead of $\mathcal{I}_\alpha(\beta)$, one could also take the so-called linear functional expression $\mathcal{I}(\alpha,\beta)$, where informing $\mathcal{I}$ is a function of both α and β. In this manner, a functional informational entity φ can depend on several informational arguments, e.g. $\alpha_1,\alpha_2,\cdots,\alpha_n$, that is $\varphi(\alpha_1,\alpha_2,\cdots,\alpha_n)$.

The notion of mathematical function belongs to the most essential constructs in mathematics. If a function gets its argument of the appropriate type then, by means of its own functionality (e.g. algorithm, procedure, program), it produces the 'regular' (legitimate, well-defined) result. This view (and technique) may be understood to be the most naïve one, that is the most simple, reductionistic, and idealistic. Informational functions (Being-of's) will be determined within the broadest informational realm, including the naïve (logical, calculational) formal constructions.

3 Basic Axioms and Definition of Informational Being-of

In this section we have to study axiomatic properties of informational externalism, internalism, metaphysicalism, and phenomenalism, pertaining to the informational Being-of. We will also use the term *informational function* of something instead of informational Being-of.

Definition 1 [Functional Notations] *We introduce the following informational implications which concern the informational Being-of:*

$$\begin{array}{lll} \varphi(\alpha) & \Longrightarrow & (\varphi \models_{\text{of}} \alpha; \alpha \models \varphi) \quad \text{or} \\ \varphi_\alpha & \Longrightarrow & (\varphi \models_{\text{of}} \alpha; \alpha \models \varphi) \quad \text{or} \\ (\varphi)(\alpha) & \Longrightarrow & (\varphi \models_{\text{of}} \alpha; \alpha \models \varphi) \end{array}$$

where φ is the functional informational entity and α is the functional argument (variable). Informational function $\varphi(\alpha)$ will be recursively defined by Definition 2. Informational operator $\models_{\text{of}}$ is a functionally particularized operator $\models$ with the meaning "informs to be dependent on" or "informs to be a function of". □

Implication formulas in the last definition are read as follows:

- $\varphi(\alpha)$: φ as a function of α implies that φ informs to be a function of α or that φ informs to be an entity arising by the impact of α.
- φ_α: φ subscript α implies that φ informs to be a function of α or that φ informs to be an entity arising by the impact of α.
- $(\varphi)(\alpha)$: φ as a complex entity depends on α as a complex entity implies that φ informs to be a function of α or that φ informs to be an entity arising by the impact of α.

Definition 1 is informationally recursive in the sense that the implicative property of functionality can be nested (derived sequentially, serialized) to an arbitrary depth.

Axiom 1 [Functional Externalism] *A function of the form $\varphi(\alpha)$, as determined by Definition 1, informs externalistically in a regular way [7], that is,*

$$\varphi(\alpha) \Longrightarrow (\varphi(\alpha) \models)$$

where the right side of operator $\Longrightarrow$ can be deconstructed in a parallel manner,

$$(\varphi(\alpha) \models) \rightleftharpoons \begin{pmatrix} \varphi \models_{\text{of}}; \\ \alpha \models; \\ (\varphi \models_{\text{of}} \alpha) \subset; \\ (\alpha \models \varphi) \subset_{\text{of}} \end{pmatrix}$$

Function $\varphi(\alpha)$ informs by all its components, φ, α, $\varphi \models_{\text{of}} \alpha$ and $\alpha \models \varphi$. □

Functional externalism says:

- that in a part of externalism, $\varphi \models_{\text{of}}$, entity φ can become a function of any (other) argument, e.g. $\varphi \models_{\text{of}} \beta$ (externalistic functional openness);
- that functional argument α informs, that is $\alpha \models$, in a general manner (externalistic argumentative openness);
- that the functional transition $\varphi \models_{\text{of}} \alpha$ informs includably in a general informational way (externalistic includable openness of functional transition); and
- that the argumentative transition $\alpha \models \varphi$ informs in an of-includable (particularly includable) way (externalistic of-includable openness of argumentative transition).

As shown in [7], the next basic axioms are in fact axiomatic consequences of Axiom 1. Let us see these axioms!

Axiom 2 [Functional Internalism] *A function of the form $\varphi(\alpha)$, as determined by Definition 1, informs internalistically in a regular way [7], that is,*

$$\varphi(\alpha) \Longrightarrow (\models \varphi(\alpha))$$

where the right side of operator $\Longrightarrow$ can be deconstructed in a parallel manner,

$$(\models \varphi(\alpha)) \rightleftharpoons \begin{pmatrix} \models_{\text{of}} \alpha; \\ \models \varphi; \\ \subset (\varphi \models_{\text{of}} \alpha); \\ \subset_{\text{of}} (\alpha \models \varphi) \end{pmatrix}$$

All components of function $\varphi(\alpha)$, that is, α, φ, $\varphi \models_{\text{of}} \alpha$ and $\alpha \models \varphi$, are specifically informed (operators $\models_{\text{of}}$, $\models$, $\subset$, and $\subset_{\text{of}}$). □

We can observe a phenomenal informational symmetry between functional externalism and functional internalism. Functional internalism says:

- that in a part of internalism, $\models_{of}\ \alpha$, entity α can become an argument of any (other) function, e.g. $\psi \models_{of} \alpha$ (internalistic argumentative openness);
- that functional entity φ is informed, that is $\models \varphi$, in a general manner (internalistic functional openness);
- that functional transition $\varphi \models_{of} \alpha$ is includably informed in a general informational way (internalistic includable openness of functional transition); and
- that argumentative transition $\alpha \models \varphi$ is of-includably (particularly includable) informed (internalistic of-includable openness of argumentative transition).

Axiom 3 [Functional Metaphysicalism] *A function of the form $\varphi(\alpha)$, as determined by Definition 1, informs metaphysically in a regular way [7], that is,*

$$\varphi(\alpha) \Longrightarrow (\varphi(\alpha) \models \varphi(\alpha))$$

where the right side of operator $\Longrightarrow$ can be deconstructed in a parallel manner,

$$(\varphi(\alpha) \models \varphi(\alpha)) \rightleftharpoons \begin{pmatrix} \varphi \models_{of} \varphi; \\ \alpha \models \alpha; \\ (\varphi \models_{of} \alpha) \subset (\varphi \models_{of} \alpha); \\ (\alpha \models \varphi) \subset_{of} (\alpha \models \varphi) \end{pmatrix}$$

Function $\varphi(\alpha)$ informs metaphysically by all its components, φ, α, $\varphi \models_{of} \alpha$ and $\alpha \models \varphi$. □

Functional metaphysicalism is a significant property of informational Being-of, since it enables the metaphysical, that is, self-productive informational arising of the function. Functional metaphysicalism says:

- that function φ can become a function of itself, that is $\varphi \models_{of} \varphi$ (metaphysical functional closeness or circularity);
- that functional argument α informs metaphysically, that is $\alpha \models \alpha$, in a general manner (metaphysical argumentative closeness or circularity);
- that the functional transition $\varphi \models_{of} \alpha$ informs metaphysical-includably in a general informational way (metaphysical-includable closeness or circularity of functional transition); and
- that the argumentative transition $\alpha \models \varphi$ informs metaphysically in an of-includable (particularly includable) way (externalistic of-includable closeness or circularity of argumentative transition).

Axiom 4 [Functional Phenomenalism] *A function of the form $\varphi(\alpha)$, as determined by Definition 1, informs phenomenalistically in a regular way [7], that is,*

$$\varphi(\alpha) \Longrightarrow \begin{pmatrix} \varphi(\alpha) \models; \\ \models \varphi(\alpha) \end{pmatrix}$$

where the right side of operator $\Longrightarrow$ can be deconstructed in a parallel manner,

$$\begin{pmatrix} \varphi(\alpha) \models; \\ \models \varphi(\alpha) \end{pmatrix} \rightleftharpoons \begin{pmatrix} \varphi \models_{of}; \\ \models_{of} \alpha; \\ \models \varphi; \\ \alpha \models; \\ (\varphi \models_{of} \alpha) \subset; \\ \subset (\varphi \models_{of} \alpha); \\ (\alpha \models \varphi) \subset_{of}; \\ \subset_{of} (\alpha \models \varphi) \end{pmatrix}$$

Function $\varphi(\alpha)$ informs phenomenalistically by all its components, φ, α, $\varphi \models_{of} \alpha$ and $\alpha \models \varphi$. □

By functional phenomenalism, the parallelism of functional externalism and functional internalism (including functional metaphysicalism in an implicit manner) is explicitly introduced into the functional game. Functional phenomenalism says:

- that functional phenomenalism informs and is informed in a functional externalistic and functional internalistic form which means:
- that in a part of externalism, $\varphi \models_{of}$, entity φ can become a function of any (other) argument, e.g. $\varphi \models_{of} \beta$ (externalistic functional openness);
- that in a part of internalism, $\models_{of}\ \alpha$, entity α can become an argument of any (other) function, e.g. $\psi \models_{of} \alpha$ (internalistic argumentative openness);

- that functional argument α informs, that is $\alpha \models$, in a general manner (externalistic argumentative openness);
- that functional entity φ is informed, that is $\models \varphi$, in a general manner (internalistic functional openness);
- that the functional transition $\varphi \models_{\text{of}} \alpha$ informs includably in a general informational way (externalistic includable openness of functional transition);
- that functional transition $\varphi \models_{\text{of}} \alpha$ is includably informed in a general informational way (internalistic includable openness of functional transition);
- that the argumentative transition $\alpha \models \varphi$ informs in an of-includable (particularly includable) way (externalistic of-includable openness of argumentative transition); and
- that argumentative transition $\alpha \models \varphi$ is of-includably (particularly includable) informed (internalistic of-includable openness of argumentative transition).

Definition 2 [Informational Function] *Let entity φ be an informational function of entity α, that is, $\varphi(\alpha)$. This expression reads:* φ is a function of α. *Let the following parallel system of informational function (Being-of) be defined recursively:*

$$\varphi(\alpha) \rightleftharpoons \begin{pmatrix} \varphi \models_{\text{of}} \alpha; \\ \alpha \models \varphi; \\ (\varphi \models_{\text{of}} \alpha) \subset \varphi; \\ (\alpha \models \varphi) \subset_{\text{of}} \varphi \end{pmatrix}$$

where, for the first informational includedness of the formula, according to [9], there is

$$((\varphi \models_{\text{of}} \alpha) \subset \varphi) \rightleftharpoons \begin{pmatrix} \varphi \models (\varphi \models_{\text{of}} \alpha); \\ (\varphi \models_{\text{of}} \alpha) \models \varphi; \\ (\varphi \models (\varphi \models_{\text{of}} \alpha)) \subset \varphi; \\ ((\varphi \models_{\text{of}} \alpha) \models \varphi) \subset \varphi \end{pmatrix}$$

and, for the second informational includedness, according to [9],

$$((\alpha \models \varphi) \subset_{\text{of}} \varphi) \rightleftharpoons \begin{pmatrix} \varphi \models_{\text{of}} (\alpha \models \varphi); \\ (\alpha \models \varphi) \models_{\text{of}} \varphi; \\ (\varphi \models_{\text{of}} (\alpha \models \varphi)) \subset_{\text{of}} \varphi; \\ ((\alpha \models \varphi) \models_{\text{of}} \varphi) \subset_{\text{of}} \varphi \end{pmatrix}$$

□

This definition recursively determines the parallel informational mechanisms of the informational Being-of, irrespective of the functional-nesting depth.

Consequence 1 [Nested Informational Functional Form $\varphi(\alpha(\beta))$] *By means of Definition 2 for an informational function, it is possible to deduce formula systems for arbitrarily deeply nested informational functional forms. For $\varphi(\alpha(\beta))$, with the nesting depth $d_{\text{nest}} = 2$, there is*

$$\varphi(\alpha(\beta)) \rightleftharpoons \begin{pmatrix} \varphi \models_{\text{of}} \alpha(\beta); \\ \alpha(\beta) \models \varphi; \\ (\varphi \models_{\text{of}} \alpha(\beta)) \subset \varphi; \\ (\alpha(\beta) \models \varphi) \subset_{\text{of}} \varphi \end{pmatrix};$$

$$((\varphi \models_{\text{of}} \alpha(\beta)) \subset \varphi) \rightleftharpoons \begin{pmatrix} \varphi \models (\varphi \models_{\text{of}} \alpha(\beta)); \\ (\varphi \models_{\text{of}} \alpha(\beta)) \models \varphi; \\ (\varphi \models (\varphi \models_{\text{of}} \alpha(\beta))) \subset \varphi; \\ ((\varphi \models_{\text{of}} \alpha(\beta)) \models \varphi) \subset \varphi \end{pmatrix};$$

$$((\alpha(\beta) \models \varphi) \subset_{\text{of}} \varphi) \rightleftharpoons \begin{pmatrix} \varphi \models_{\text{of}} (\alpha(\beta) \models \varphi); \\ (\alpha(\beta) \models \varphi) \models_{\text{of}} \varphi; \\ (\varphi \models_{\text{of}} (\alpha(\beta) \models \varphi)) \subset_{\text{of}} \varphi; \\ ((\alpha(\beta) \models \varphi) \models_{\text{of}} \varphi) \subset_{\text{of}} \varphi \end{pmatrix}$$

and within these formulas, for the nested functional component $\alpha(\beta)$, there is,

$$\alpha(\beta) \rightleftharpoons \begin{pmatrix} \alpha \models_{\text{of}} \beta; \\ \beta \models \alpha; \\ (\alpha \models_{\text{of}} \beta) \subset \alpha; \\ (\beta \models \alpha) \subset_{\text{of}} \alpha \end{pmatrix}$$

where

$$((\alpha \models_{\text{of}} \beta) \subset \alpha) \rightleftharpoons \begin{pmatrix} \alpha \models (\alpha \models_{\text{of}} \beta); \\ (\alpha \models_{\text{of}} \beta) \models \alpha; \\ (\alpha \models (\alpha \models_{\text{of}} \beta)) \subset \alpha; \\ ((\alpha \models_{\text{of}} \beta) \models \alpha) \subset \alpha \end{pmatrix}$$

and

$$((\beta \models \alpha) \subset_{\text{of}} \alpha) \rightleftharpoons \begin{pmatrix} \alpha \models_{\text{of}} (\beta \models \alpha); \\ (\beta \models \alpha) \models_{\text{of}} \alpha; \\ (\alpha \models_{\text{of}} (\beta \models \alpha)) \subset_{\text{of}} \alpha; \\ ((\beta \models \alpha) \models_{\text{of}} \alpha) \subset_{\text{of}} \alpha \end{pmatrix}$$

Formulas with operators $\subset$ and $\subset_{\text{of}}$ can then be derived according to Definition 1 in [9]. □

4 Informational (Verbal) Interpretations of Formula $\varphi \models_{\text{of}} \alpha$

In this section we have to clarify the reading, meaning, and possibilities of formula $\varphi \models_{\text{of}} \alpha$. We have already listed the possible meanings of the word 'of'. Different 'equivalents' to operator $\models_{\text{of}}$ are possible, where the orientation of the equivalent operator may be reversed in respect to operands φ and α.

Formula $\varphi \models_{\text{of}} \alpha$ is a rudimentary informational formula, where operator $\models_{\text{of}}$ connects the left operand φ with the right operand α. This operator is nothing else than a particularization of the informational operational metaphor $\models$ which represents an operational joker (a place for the possible particular operational possibility). So, let us introduce a precise, manifold, and parallel structured definition of the case $\varphi \models_{\text{of}} \alpha$.

Definition 3 [Reading Formula $\varphi \models_{\text{of}} \alpha$] *Informational formula of the form*

$$\varphi \models_{\text{of}} \alpha$$

is read in the following possible manners:

1. *Operand entity φ informs to be informationally dependent on operand entity α.*
2. *Operand φ is an informational function of operand α.*
3. *Operand α is informed that operand φ informationally depends on α itself $[(\alpha \models \varphi) \models \alpha]$ (and, as a consequence of informational openness of formulas, on other operand entities).*
4. *Operand α is informed that operand φ is an informational function of α (itself) $[\varphi(\alpha) \models_{\text{of}} \alpha]$.*
5. *Operand α is informed that operand φ is caused by α itself (and possibly by other operands).*
6. *Operand α dependently (functionally) informs operand φ $[\alpha \models_{\text{depend}} \varphi]$.*
7. *Operand α is a constructor (co-constructor) of operand φ.*
8. *$\varphi \models_{\text{of}} \alpha$ is an informational functional principle, which causes some other consequent informational formulas to come into existence. E.g., $(\varphi \models_{\text{of}} \alpha) \Longrightarrow (\alpha \models \varphi)$. And so forth.*

These cases do not exhaust other possible interpretations of reading of formula $\varphi \models_{\text{of}} \alpha$. □

Additional interpretations of formula $\varphi \models_{\text{of}} \alpha$ come to the surface when considering meanings, which pertain to the meaning of the word 'of'.

Consequence 2 [A Possible Parallel Informational Interpretation of Formula $\varphi \models_{\text{of}} \alpha$] *Considering the language concepts pertaining to the word 'of' [10], there is,*

$$(\varphi \models_{\text{of}} \alpha) \Longrightarrow \left(\begin{array}{l} \varphi \models_{\text{be\_a\_function\_of}} \alpha; \\ \varphi \models_{\text{be\_dependent\_on}} \alpha; \\ \varphi \models_{\text{be\_a\_derivation\_of}} \alpha; \\ \varphi \models_{\text{be\_a\_consequence\_of}} \alpha; \\ \varphi \models_{\text{be\_an\_instrument\_of}} \alpha; \\ \varphi \models_{\text{come\_from}} \alpha; \\ \varphi \models_{\text{arise\_from}} \alpha; \\ \varphi \models_{\text{result\_from}} \alpha; \\ \varphi \models_{\text{be\_removed\_from}} \alpha; \\ \varphi \models_{\text{from}} \alpha; \\ \varphi \models_{\text{within\_of}} \alpha; \\ \varphi \models_{\text{be\_delivered\_by}} \alpha; \\ \varphi \models_{\text{be\_generated\_by}} \alpha; \\ \alpha \models_{\text{cause}} \varphi; \\ \alpha \models_{\text{deliver}} \varphi; \\ \alpha \models_{\text{generate}} \varphi \end{array} \right)$$

Operator $\Longrightarrow$ marks that on its right side only some of the known parallel alternatives concerning its left side are listed. □

Formula $\varphi \models_{\text{of}} \alpha$ is understood to mean the listed possibilities in a parallel manner, and also other possibilities which may arise in an informational situation.

5 A Notion of the Informational Frame

Notion of an informational frame is in no connection with the frame in psychology. Here, a frame is simply another word for a formation of elements (operands, operators, and/or parentheses) which appear in informational formulas. Informational frame is an arbitrary serried (compact) section

of a well-formed informational formula. We are forced to introduce this strange (irregular) structure, called frame, to master some problems of various possibilities of informational formula arising.

In this section we have to define the notion of an informational frame in a formal manner. We begin with the statement that each informational formula, which is a well-formed structure of operands, operators, and parentheses, is a frame. Such a frame is viewed as a well-structured and well-organized informational whole. But, if we are breaking-down a formula introspectively into its arbitrary structured components, we do in no way discard the original formula as a whole. The breaking-down has the role of additional interpretation possibilities of the original formula and, as we will see, an identification of frames within a frame in the sense of simple inclusion.

We distinguish several kinds of informational frames: operand or formula frames are called *harmonious* frames. On the other hand, operator frames or any other non-well-formed arrays of informational components (operands, operators, parentheses) are called *disharmonious* frames.

Definition 4 [Harmonious Informational Frames] Harmonious informational frames *are enframed, well-formed informational formulas or well-formed parts of formulas (subformulas), built-up according to the informational formula syntax. Thus,*

$$\boxed{\alpha}, \boxed{(\alpha)}, \boxed{\alpha \models}, \boxed{(\alpha \models)}, \boxed{\models \alpha}, \boxed{\alpha \models \beta},$$
$$(\boxed{(\alpha \models \beta) \models \gamma}) \models \delta) \models \varepsilon$$

etc. are examples of harmonious informational frames. □

Harmonious frames arise together with the arising of informational formulas.

Definition 5 [Disharmonious Informational Frames] Disharmonious informational frames *are enframed, syntactically non-well-formed parts of informational formulas. Thus,*

$$\boxed{(}\alpha), (\alpha\boxed{)}, \boxed{(\alpha}), (\boxed{\alpha)}, \boxed{(\alpha \models}), (\alpha \boxed{\models)},$$
$$\alpha\boxed{\models}\beta, \boxed{((\alpha} \models \beta\boxed{) \models \gamma) \models}\delta$$

etc. are examples of disharmonious informational frames. □

Disharmonious frames arise together with the arising of informational formulas.

Definition 6 [Embedded Harmonious and Disharmonious Informational Frames] *Harmonious and disharmonious informational frames can be embedded in other informational frames to any possible depth and form. For example,*

$$\boxed{(\boxed{\alpha})}, \boxed{(\boxed{(}\alpha)}, \boxed{\alpha\boxed{\models}}, \boxed{\boxed{\alpha}\models\boxed{\beta}},$$
$$\boxed{(\boxed{\alpha}\models\boxed{\beta})}, \boxed{(\boxed{\alpha}\models\boxed{\beta)}\models\gamma}$$

etc. are examples of embedded harmonious and disharmonious informational frames. □

We see how an informational formula can be systematically enframed by frames, so that the result is a complete enframing and frames "connection".

Definition 7 [Well-enframed Formulas] *An informational formula or a frame in or of a formula is well-enframed or frame-formed, if all formula components concerning it are enframed in the following way:*

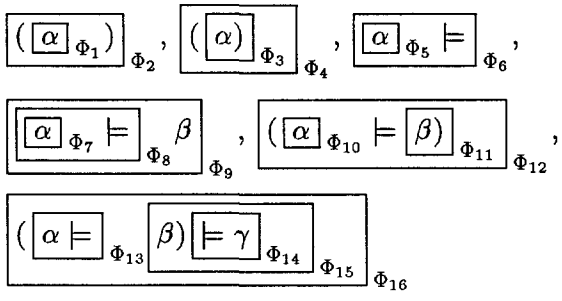
1. *A well-formed formula is enframed, e.g.* $\boxed{\alpha}$.
2. *Two adjacent frames, harmonious and disharmonious, or disharmonious and harmonious, or disharmonious and disharmonious can be concatenated, e.g.* $\boxed{(\alpha}\boxed{)}$.
3. *Within a formula frame, there are concatenated frames, e.g.* $\boxed{(\alpha}\boxed{\models}\boxed{\beta)}$.
4. *If a formula frame is completely filled with concatenated frames, harmonious and disharmonious ones, it is called the well-enframed formula.*

The procedure of enframing of formula parts starts from the formula as a whole. □

Definition 8 [Parenthesis Frames] *Two parenthesis frames are distinguished: the left parenthesis frame,* $\Phi^n_($, *where* $n = 0, 1, 2, \cdots$, *marking a sequence of n left parentheses, e.g.* $\Phi^4_(\rightleftharpoons \boxed{((((}$ *and the right parenthesis frame,* $\Phi^n_)$, *where* $n = 0, 1, 2, \cdots$, *marking a sequence of n right parentheses, e.g.* $\Phi^5_) \rightleftharpoons \boxed{)))))}$.

Parenthesis frames $\Phi_($ and $\Phi_)$ mark a frame of an adequate number of the left and the right parentheses, respectively. □

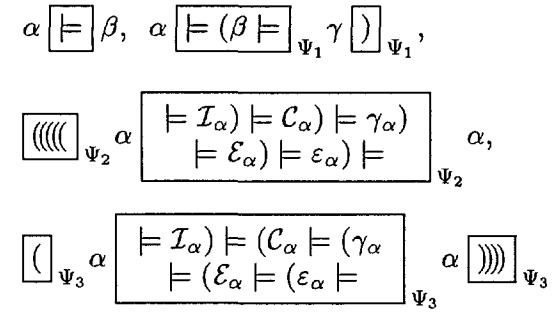
Definition 9 [Subscript Embedded Harmonious and Disharmonious Informational Frames] *Embedded harmonious and disharmonious informational frames can be subscribed with the aim to distinguish them for the purpose of their textual description and further informational interpretation. A subscribed frame is marked by the subscribed Φ, which is the marker for the enframed informational frame. For example,*



etc. are examples of embedded harmonious and disharmonious informational frames. □

Subscribing informational frames, we can discuss them concretely. For instance, frames Φ_1, Φ_5, Φ_7, and Φ_{10} mark equivalent harmonious frames, which are well-formed formulas marked by operand α. Examples of disharmonious informational frames are Φ_3, Φ_{11}, and Φ_{15}, representing non-well-formed formulas (non-well-formed parts of well-formed formulas).

Definition 10 [Disharmonious Informational Frames Concerning Informational Operators] *Operator frames are not arbitrarily disharmonious; they must satisfy the condition to be sequences of operands, operators, and parentheses set between two operands. Within this rough determination, they can be split in two parts and united through the unique frame subscript Ψ_n. Particular examples of operator frames are:*



etc. □

In the first example, operator frame is operator $\models$, that is, $\boxed{\models}$. In the second case, operator frame Ψ_1 is split in two parts, that is, in $\boxed{\models(\beta\models}$ and $\boxed{)}$. In the third example, we have a metaphysical operator frame Ψ_2 between operands α and α, that is, the enframed part $\boxed{\models\mathcal{I}_\alpha)\models\mathcal{C}_\alpha)\models\gamma_\alpha)\models\mathcal{E}_\alpha)\models\varepsilon_\alpha)\models}$ and, before this enframed part (before α), the split part of Ψ_2, that is, $\boxed{(((((}$ which equals $\Phi_(^5$. These two frames (the left and the right frame Ψ_2) constitute an informational operator between two operands α, that is $\Psi_2\alpha\Psi_2\alpha$, which may result as a decomposition (destruction) of the initial metaphysical situation $\alpha\models\alpha$. In the fourth example, we have a metaphysical operator frame Ψ_3 between operands α and α, that is, the enframed part $\boxed{\models\mathcal{I}_\alpha)\models(\mathcal{C}_\alpha\models(\gamma_\alpha\models(\mathcal{E}_\alpha\models(\varepsilon_\alpha\models}$ and, before this enframed part (before α), the split part of Ψ_3, that is, $\boxed{(}$ which equals $\Phi_(^1$. After the middle enframed part, there is the right split part of Ψ_3, that is $\boxed{))))}$, which equals $\Psi_)^4$. These three frames (the left, middle and the right frame Ψ_3) constitute an informational operator between two operands α, that is $\Psi_3\alpha\Psi_3\alpha\Psi_3$, which may result as a decomposition (destruction) of the initial metaphysical situation $\alpha\models\alpha$.

The concept of informational frame becomes very helpful in studying of possibilities of the so-called informational gestalts pertaining to serial and metaphysical functionality.

6 Serial Informational Functionality

Serial informational functionality offers several possibilities of its understanding and to this understanding adequate notation. At the beginning, we consider the most conventional form of functionality, which has its roots in the mathematical tradition.

Consequence 3 [Implicative Serial Functional Forms] *According to Definition 2, for the functionally nested expressions the following informational implications are evident:*

$$\varphi(\alpha(\beta)) \implies \begin{pmatrix} \varphi \models_{\text{of}} (\alpha \models_{\text{of}} \beta); \\ (\beta \models \alpha) \models \varphi \end{pmatrix};$$

$$\varphi(\alpha(\beta(\gamma))) \implies \begin{pmatrix} \varphi \models_{\text{of}} (\alpha \models_{\text{of}} \\ \quad (\beta \models_{\text{of}} \gamma)); \\ (((\gamma \models \beta) \models \alpha) \models \varphi \end{pmatrix};$$

$$\vdots$$

$$\varphi(\alpha(\beta(\cdots\psi(\omega)\cdots))) \implies$$

$$\begin{pmatrix} \varphi \models_{\text{of}} (\alpha \models_{\text{of}} \\ \quad (\beta \models_{\text{of}} (\cdots(\psi \models_{\text{of}} \\ \quad \omega)\cdots)); \\ (((\cdots(\omega \models \psi) \models \cdots\beta) \models \\ \quad \alpha) \models \varphi \end{pmatrix}$$

These cases of informational functionality we call natural *serial functional forms.* □

Consequence 4 [Natural Serial Functional Forms] *Let* $2 \cdot (n-1)$ *serial forms of entities* $\alpha_1, \alpha_2, \cdots, \alpha_n$, *that is,*

$$\alpha_1 \models^*_{\text{of}} (\alpha_2 \models_{\text{of}} (\alpha_3 \models_{\text{of}} (\cdots(\alpha_{n-1} \models_{\text{of}} \alpha_n)\cdots)));$$
$$(((\cdots(\alpha_n \models \alpha_{n-1})\cdots) \models \alpha_3) \models \alpha_2) \models^* \alpha_1;$$

$$(\alpha_1 \models_{\text{of}} \alpha_2) \models^*_{\text{of}} (\alpha_3 \models_{\text{of}} (\cdots(\alpha_{n-1} \models_{\text{of}} \alpha_n)\cdots));$$
$$((\cdots(\alpha_n \models \alpha_{n-1})\cdots) \models \alpha_3) \models^* (\alpha_2 \models \alpha_1);$$

$$((\alpha_1 \models_{\text{of}} \alpha_2) \models_{\text{of}} \alpha_3) \models^*_{\text{of}} (\cdots(\alpha_{n-1} \models_{\text{of}} \alpha_n)\cdots);$$
$$(\cdots(\alpha_n \models \alpha_{n-1})\cdots) \models^* (\alpha_3 \models (\alpha_2 \models \alpha_1));$$

$$\vdots$$

$$((\cdots((\alpha_1 \models_{\text{of}} \alpha_2) \models_{\text{of}} \alpha_3)\cdots) \models_{\text{of}} \alpha_{n-1}) \models^*_{\text{of}} \alpha_n;$$
$$\alpha_n \models^* (\alpha_{n-1} \models (\cdots(\alpha_3 \models (\alpha_2 \models \alpha_1))\cdots))$$

be given. According to Definition 2, these serial forms can evidently be implied by the corresponding functional forms

$$\alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots)));$$
$$\alpha_1(\alpha_2)(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots));$$
$$\alpha_1(\alpha_2(\alpha_3))(\cdots(\alpha_{n-1}(\alpha_n))\cdots);$$
$$\cdots$$
$$\alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1})\cdots)))(\alpha_n)$$

respectively. Operators $\models^*_{\text{of}}$ *and* $\models^*$ *mark the main operators of particular sequences (the distinguishing operational points between the main informer and observer entities).* □

To make the main function distinguishable from the argument function, we can introduce the separation marker * between them, that is, between the consequent parentheses ')' and '('. Thus, the last sequence of functional markers becomes

$$\boxed{(\alpha_1)}\,{}^*(\alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots)));$$
$$\boxed{\alpha_1(\alpha_2)}\,{}^*(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots));$$
$$\boxed{\alpha_1(\alpha_2(\alpha_3))}\,{}^*(\cdots(\alpha_{n-1}(\alpha_n))\cdots);$$
$$\vdots$$
$$\boxed{\alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1})\cdots)))}\,{}^*(\alpha_n)$$

In the first case, we put α_1 between parentheses. Each line of the last functional array can be further decomposed, keeping the sequence of operands $\alpha_1, \alpha_2, \cdots, \alpha_n$ preserved and only mutating the parenthesis pairs. This decomposition procedure delivers new functional forms exhausting at the end all possibilities of the upper $n-1$ functions (lines). Thus, for instance, the first function decomposes into

$$\boxed{(\alpha_1)}\,{}^*(\,\boxed{(\alpha_2)}\,{}^*(\alpha_3(\alpha_4(\cdots(\alpha_{n-1}(\alpha_n))\cdots)));$$
$$\boxed{(\alpha_1)}\,{}^*(\,\boxed{\alpha_2(\alpha_3)}\,{}^*(\alpha_4(\cdots(\alpha_{n-1}(\alpha_n))\cdots)));$$
$$\boxed{(\alpha_1)}\,{}^*(\,\boxed{\alpha_2(\alpha_3(\alpha_4))}\,{}^*(\alpha_5(\cdots(\alpha_{n-1}(\alpha_n))\cdots)));$$
$$\vdots$$
$$\boxed{(\alpha_1)}\,{}^*(\,\boxed{\alpha_2(\alpha_3(\cdots(\alpha_{n-2}(\alpha_{n-1}))\cdots))}\,{}^*(\alpha_n))$$

with two stars in each line, etc., recursively, then with three stars, etc. In each line, two frames show the function-of-function situation in an evident manner.

7 Parallel Informational Functionality

Parallel informational functionality can be conceptualized in different ways. We will deal only with some of the most evident cases.

Definition 11 [A Form of Parallel Well-connected Informational Functionality] *Let* $\alpha_1, \alpha_2, \cdots, \alpha_n$ *be operand entities. An informational system* $\alpha^{\|}(\alpha_1, \alpha_2, \cdots, \alpha_n)$ *is called a well-connected, functionally parallel system, if*

$$\alpha^{\|}(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{n-2}, \alpha_{n-1}, \alpha_n) \rightleftharpoons \begin{pmatrix} \alpha_1(\alpha_2); \\ \alpha_2(\alpha_3); \\ \vdots \\ \alpha_{n-2}(\alpha_{n-1}); \\ \alpha_{n-1}(\alpha_n) \end{pmatrix}$$

This is, in fact, a serially connected parallel system. □

Which could be a senseful (adequate) consequence of the introduced parallel system $\alpha^{\|}(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{n-2}, \alpha_{n-1}, \alpha_n)$?

Consequence 5 [A Form of Substitution of Parallel Informational Functions] *Let $\alpha_1, \alpha_2, \cdots, \alpha_n$ be operand entities belonging to the system in Definition 11. By the operation of substitutional implication, there is, evidently,*

$$\begin{pmatrix} \alpha_1(\alpha_2); \\ \alpha_2(\alpha_3); \\ \vdots \\ \alpha_{n-2}(\alpha_{n-1}); \\ \alpha_{n-1}(\alpha_n) \end{pmatrix} \Longrightarrow_{\text{substitute}}$$

$$\begin{pmatrix} \alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-2}(\alpha_{n-1}(\alpha_n)))\cdots))); \\ \alpha_2(\alpha_3(\cdots(\alpha_{n-2}(\alpha_{n-1}(\alpha_n)))\cdots)); \\ \vdots \\ \alpha_{n-2}(\alpha_{n-1}(\alpha_n)) \end{pmatrix}$$

Certainly, also 'shorter' functional formulas are possible. □

A parallel array of shorter formulas instead of the first formula on the right side of operator $\Longrightarrow_{\text{substitute}}$ would be

$$\begin{array}{l} \alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-2}(\alpha_{n-1}))\cdots))); \\ \alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-2})\cdots))); \\ \vdots \\ \alpha_1(\alpha_2(\alpha_3)) \end{array}$$

Consequence 6 [A Parallel Functional Dependence] *A function α can simultaneously (in parallel) depend on more than only one operand. This parallelism of dependence on several operands can be expressed as*

$$\begin{pmatrix} \alpha(\alpha_1); \\ \alpha(\alpha_2); \\ \vdots \\ \alpha(\alpha_n) \end{pmatrix} \Longrightarrow \alpha(\alpha_1, \alpha_2, \cdots, \alpha_n)$$

that is,

$$\alpha(\alpha_1, \alpha_2, \cdots, \alpha_n) \Longrightarrow \begin{pmatrix} \alpha \models_{\text{of}} \alpha_1, \alpha_2, \cdots, \alpha_n; \\ \alpha_1, \alpha_2, \cdots, \alpha_n \models \alpha \end{pmatrix}$$

which proves the adequacy of the introduced parallel functional expression. □

Informational parallelism and informational functionality are informationally dependent phenomena, which interfere with each other. Functions $\alpha^{\|}(\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{n-2}, \alpha_{n-1}, \alpha_n)$ and $\alpha(\alpha_1, \alpha_2, \cdots, \alpha_n)$ (Definition 11 and Consequence 6, respectively) are essentially different functional structures (functionalities).

Consequence 7 [Informational Parallelism and Functionality] *The beginning question is, what is the difference between the regular functional expression $\alpha(\alpha_1, \alpha_2, \cdots, \alpha_n)$ and expression $\alpha(\alpha_1; \alpha_2; \cdots; \alpha_n)$ where commas have been replaced by semicolons. The comma system $(\alpha_1, \alpha_2, \cdots, \alpha_n)$ is a system of separated entities $\alpha_1, \alpha_2, \cdots, \alpha_n$ which may or may not cooperate (inform among each other). The semicolon system $(\alpha_1; \alpha_2; \cdots; \alpha_n)$ is a characteristic parallel system in which semicolons are nothing else than parallel informational operators (e.g., $\|\models$). The meaning is*

$$(\alpha_1; \alpha_2; \cdots; \alpha_n) \rightleftharpoons \begin{pmatrix} \alpha_i \|\models \alpha_j; \\ i \neq j; \\ i, j = 1, 2, \cdots, n \end{pmatrix}$$

Operator $\|\models$ has the meaning "informs in parallel with". "In parallel" means simultaneously, dependently or independently, spontaneously, circularly, particularly, etc. For instance, for the meaning of the last formula there are the following three alternatives:

$$\begin{pmatrix} \alpha_i \Vdash \alpha_j; \\ i \neq j; \\ i,j = 1,2,\cdots,n \end{pmatrix} \rightleftharpoons$$

$$\left(\begin{pmatrix} \alpha_i \nVdash \alpha_j; \\ i \neq j; \\ i,j = 1,2,\cdots,n \end{pmatrix} \vee \begin{bmatrix} \text{parallel} \\ \text{independence} \end{bmatrix} \right.$$

$$\begin{pmatrix} (\alpha_i \models \alpha_j); \\ (\alpha_j \nVdash \alpha_i); \\ i \neq j; \\ i,j = 1,2,\cdots,n \end{pmatrix} \vee \begin{bmatrix} \text{partly} \\ \text{parallel} \\ \text{dependence/} \\ \text{independence} \end{bmatrix}$$

$$\left. \begin{pmatrix} \alpha_i \models \alpha_j; \\ i \neq j; \\ i,j = 1,2,\cdots,n \end{pmatrix} \begin{bmatrix} \text{a complete} \\ \text{parallel} \\ \text{dependence} \end{bmatrix} \right)$$

where operator $\vee$ *replaces the usual semicolon and means 'or' (informational 'or', that is an informational alternative).* □

Informational operator $\Vdash$ enables an explicit studying of informational parallelism, especially in a functional environment.

8 Circular Informational Functionality

Circular informational function as an informational function belongs to the phenomenon of circular serial phenomenality. An adequate functional parallelism would mean simply an occurrence of adequate functions in parallel, which build an cyclically structured system of simpler informational functions. In this section, we have to study a sufficiently general concept of circular informational function by means of informational frames.

Definition 2 guarantees some basic forms of informational functionality which can be developed (decomposed, deconstructed) to complex circularly functional schemes.

Consequence 8 [Some Basic Forms of Circularity Pertaining to Informational Function] *According to Definition 2, the following implications can be deduced:*

$$((\varphi \models (\varphi \models_{\text{of}} \alpha)) \subset \varphi) \Longrightarrow$$
$$((\boxed{\varphi \models \underbrace{(\varphi \models_{\text{of}} \alpha)}_{\text{functional transition}}}) \models \varphi);$$

$$(((\varphi \models_{\text{of}} \alpha) \models \varphi) \subset \varphi) \Longrightarrow$$
$$(\varphi \models (\boxed{\underbrace{(\varphi \models_{\text{of}} \alpha)}_{\text{functional transition}} \models \varphi}));$$

$$(((\varphi \models_{\text{of}} (\alpha \models \varphi)) \subset_{\text{of}} \varphi) \Longrightarrow$$
$$((\boxed{\varphi \models_{\text{of}} \underbrace{(\alpha \models \varphi)}_{\text{argumentative transition}}}) \models_{\text{of}} \varphi);$$

$$(((\alpha \models \varphi) \models_{\text{of}} \varphi) \subset_{\text{of}} \varphi) \Longrightarrow$$
$$(\varphi \models_{\text{of}} (\boxed{\underbrace{(\alpha \models \varphi)}_{\text{argumentative transition}} \models_{\text{of}} \varphi}))$$

The marked functional transition appears within the general informing φ*-cycles while the marked argumentative transition is a part of particularly informing (of-informing)* φ*-cycles.* □

Definition 12 [General Circular Informational Function] *A* general *circular informational function,* $\varphi^{\circlearrowleft}(\alpha)$*, is a functional informational system, that is,*

$$\varphi^{\circlearrowleft}(\alpha) \rightleftharpoons (\Phi_(\; \varphi(\alpha) \; \Phi \; \varphi(\alpha) \; \Phi_))$$

or, expressed in a functionally detailed form,

$$\varphi^{\circlearrowleft}(\alpha)^*(\alpha_1, \alpha_2, \cdots, \alpha_n) \rightleftharpoons$$
$$(\Phi_(\; \varphi(\alpha) \; \Phi(\alpha_1, \alpha_2, \cdots, \alpha_n) \; \varphi(\alpha) \; \Phi_))$$

where the interior *(circular, loop) operands* α_1, α_2, $\cdots$, α_n *may arbitrarily depend on the* exterior *operand* α*, that is,* $\alpha_1(\alpha)$, $\alpha_2(\alpha)$, $\cdots$, $\alpha_n(\alpha)$. *An* inverse general *circular informational function,* $\varphi^{\circlearrowleft}(\alpha)$*, is a functional informational system, that is,*

$$\varphi^{\circlearrowleft}(\alpha) \rightleftharpoons (\Phi_(\; \varphi(\alpha) \; \Phi^{-1} \; \varphi(\alpha) \; \Phi_))$$

or, in a functionally detailed form,

$$\varphi^{\circlearrowleft}(\alpha)^*(\alpha_1, \alpha_2, \cdots, \alpha_n) \rightleftharpoons$$
$$(\Phi_(\; \varphi(\alpha) \; \Phi^{-1}(\alpha_1, \alpha_2, \cdots, \alpha_n) \; \varphi(\alpha) \; \Phi_))$$

Informational frames $\Phi_($, $\Phi_)$, Φ, *and* Φ^{-1} *are defined in the following way:*

– $\Phi_($ *is a left-parenthesis frame, which can also be empty (an empty place, marked by* Λ*). Instead of it, we introduce a significant marker,* $\Subset$*. Thus,*

$$\Phi_(\rightleftharpoons \Subset;$$
$$\Subset \rightleftharpoons \begin{cases} \Lambda, \text{ if empty place;} \\ \underbrace{((\cdots(}_{\ell\text{-times}}, \text{ if } \ell > 0 \end{cases}$$

– $\Phi_)$ *is a right-parenthesis frame, which can also be empty (an empty place, marked by* Λ*). Instead of it, we introduce a significant marker,* $\Supset$*. Thus,*

$$\Phi_) \rightleftharpoons \Supset;$$
$$\Supset \rightleftharpoons \begin{cases} \Lambda, \text{ if empty place;} \\ \underbrace{))\cdots)}_{\ell\text{-times}}, \text{ if } \ell > 0 \end{cases}$$

– Φ *or* $\Phi(\alpha_1, \alpha_2, \cdots, \alpha_n)$ *is a general frame, which is a disharmonious frame of a formula, called the right-frame. Instead of it, we introduce a significant marker,* $\Rrightarrow$*. Thus,*

$$\Phi \rightleftharpoons \Rrightarrow \quad \text{or}$$
$$\Phi(\alpha_1, \alpha_2, \cdots, \alpha_n) \rightleftharpoons \Rrightarrow(\alpha_1, \alpha_2, \cdots, \alpha_n)$$

– Φ^{-1} *or* $\Phi^{-1}(\alpha_1, \alpha_2, \cdots, \alpha_n)$ *is a general inverted frame, which is a disharmonious frame of a formula; it is called the left-frame. Instead of it, we introduce a significant marker,* $\Lleftarrow$*. Thus,*

$$\Phi^{-1} \rightleftharpoons \Lleftarrow \quad \text{or}$$
$$\Phi^{-1}(\alpha_1, \alpha_2, \cdots, \alpha_n) \rightleftharpoons \Lleftarrow(\alpha_1, \alpha_2, \cdots, \alpha_n)$$

General informational function is said to be right-circular whereas inverse general informational function is said to be left-circular. □

Definition 13 [Particular Circular Informational Function] *A* particular *circular informational function,* $\varphi^{\circlearrowright}_{\text{part}}(\alpha)$*, is a functional informational system, that is,*

$$\varphi^{\circlearrowright}_{\text{part}}(\alpha) \rightleftharpoons (\Phi_(\; \varphi(\alpha) \; \Phi_{\text{part}} \; \varphi(\alpha) \; \Phi_)) \quad \text{or}$$

$$\varphi^{\circlearrowright}_{\text{part}}(\alpha)^*(\alpha_1(\alpha), \alpha_2(\alpha), \cdots, \alpha_n(\alpha)) \rightleftharpoons (\Phi_(\; \varphi(\alpha) \; \Phi_{\text{part}}(\alpha_1(\alpha), \alpha_2(\alpha), \cdots, \alpha_n(\alpha)) \; \varphi(\alpha) \; \Phi_))$$

An inverse particular *circular informational function,* $\varphi^{\circlearrowleft}_{\text{part}}(\alpha)$*, is a functional informational system, that is,*

$$\varphi^{\circlearrowleft}_{\text{part}}(\alpha) \rightleftharpoons (\Phi_(\; \varphi(\alpha) \; \Phi^{-1}_{\text{part}} \; \varphi(\alpha) \; \Phi_)) \quad \text{or}$$

$$\varphi^{\circlearrowleft}_{\text{part}}(\alpha)^*(\alpha_1(\alpha), \alpha_2(\alpha), \cdots, \alpha_n(\alpha)) \rightleftharpoons (\Phi_(\; \varphi(\alpha) \; \Phi^{-1}_{\text{part}}(\alpha_1(\alpha), \alpha_2(\alpha), \cdots, \alpha_n(\alpha)) \; \varphi(\alpha) \; \Phi_))$$

Subscript 'part' marks a particular case of frame Φ_{part} *or* Φ^{-1}_{part} *(e.g., numerical index, semantic designator, particular symbol, etc.) in which operands* $\alpha_1(\alpha)$, $\alpha_2(\alpha)$, $\cdots$, $\alpha_n(\alpha)$ *occur in the sequence written.* □

Consequence 9 [Circular Informational Shell and Function] *According to the previous definition, a right-circular and left-circular functional shells* $\varphi^{\circlearrowright}$ *and* $\varphi^{\circlearrowleft}$ *are*

$$\varphi^{\circlearrowright} \rightleftharpoons \Subset\varphi \Rrightarrow \varphi\Supset;$$
$$\varphi^{\circlearrowleft} \rightleftharpoons \Subset\varphi \Lleftarrow \varphi\Supset$$

respectively. For a function $\varphi(\alpha)$*, the circular forms are*

$$\varphi^{\circlearrowright}(\alpha) \rightleftharpoons \Subset\varphi(\alpha) \Rrightarrow_{\varphi(\alpha)} \varphi(\alpha)\Supset;$$
$$\varphi^{\circlearrowleft}(\alpha) \rightleftharpoons \Subset\varphi(\alpha) \Lleftarrow_{\varphi(\alpha)} \varphi(\alpha)\Supset$$

where $\Rrightarrow_{\varphi(\alpha)}$ *and* $\Lleftarrow_{\varphi(\alpha)}$ *are concrete informational frames depending on operand* α*, for example, general or basic metaphysical frames of entity* α*.* □

Definition 14 [Inverse Informational Frame $\Lleftarrow$ in Regard to Informational Frame $\Rrightarrow$] *An inverse informational frame* $\Lleftarrow(\Phi^{-1})$ *to the informational frame* $\Rrightarrow(\Phi)$ *is obtained by the following procedure:*

– *In right-frame* $\Rrightarrow(\Phi)$*, all informational operators* $\models$ *(left-to-right operators) are replaced by the alternative informational operators* $=\!|$*.*

– *The left parentheses become the meaning of the right ones and vice versa.*

– *If in this manner modified frame is read from the right side to the left side, the resulting frame is the inverse informational frame,* $\Lleftarrow(\Phi^{-1})$*.*

– *Now, the modified frame can be typographically inverted, so that the right end comes to the left beginning and the left beginning to the right end. The result is the inverted graph with informational operators* $\models$.

Example 1 [A Frame and Its Inversion] Considering the frames, marked by Φ_2 in Definition 10 and the formula as a whole, there is,

$$\boxed{}_{\Psi_2}\alpha\boxed{\begin{array}{c}\models\mathcal{I}_\alpha)\models\mathcal{C}_\alpha)\models\gamma_\alpha)\\ \models\mathcal{E}_\alpha)\models\varepsilon_\alpha)\models\end{array}}_{\Psi_2}\alpha;$$

$$\alpha\boxed{\begin{array}{l}\models(\varepsilon_\alpha\models(\mathcal{E}_\alpha\models\\ \quad(\gamma_\alpha\models(\mathcal{C}_\alpha\models(\mathcal{I}_\alpha\models\end{array}}_{\Psi_2^{-1}}\alpha\boxed{}_{\Psi_2^{-1}}$$

The second formula is obtained from the first formula by the replacement of operators $\models$ by operators $=\!|$, reading the formula from the left to the right, reading the right frame from the bottom to the top, and understanding left-parenthesis frame as the right-parentheses one. That is,

$$\boxed{}_{\Psi_2^{-1}}\alpha\boxed{\begin{array}{c}=\!|\,\mathcal{I}_\alpha)=\!|\,\mathcal{C}_\alpha)=\!|\,\gamma_\alpha)\\ =\!|\,\mathcal{E}_\alpha)=\!|\,\varepsilon_\alpha)=\!|\end{array}}_{\Psi_2^{-1}}\alpha$$

The reader can observe the inverted formulas in the first and the second case. But, the shortened forms of the original and inverted formula would be

$$\Subset\alpha\Rrightarrow_\alpha\alpha;$$
$$\alpha\Lleftarrow_\alpha\alpha\Supset$$

It is to stress that $\Subset$ in the first formula can be automatically identified from frame $\Rrightarrow_\alpha$, where the right parentheses are counted. The same can be considered for $\Supset$ in the second formula. □

9 Decomposition (Deconstruction) of Informational Functionality

Decomposition or deconstruction[2] of an informational situation and attitude is nothing else than a process of interpretation in which serial, parallel, circular or otherwise mixed ways of deconstruction can come into existence. A functional expression as a beginning situation (concept, idea) must be deconstructed in concrete details, by which both functional and argument components become informationally determined. Deconstruction means that different functional (and other) markers come to the surface, where they play significant roles in further (especially metaphysical) way of decomposition. To different functional markers, different functional systems (concepts) can be associated.

A complex functional form can always be deconstructed in more primitive forms which constitute the complex formula. This is a very natural way of parallel decomposition which reveals the structure of a formula, that is, its informationally distinguished components.

Consequence 10 [A Parallel Functional Decomposition of $\alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots)))$**]** *Let us introduce the implicative decomposition (operator* $\Longrightarrow_{\text{decompose}}$ *which reads "informs decomposingly") of the nested functional form* $\alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots)))$ *in the following way:*

$$\alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots))) \Longrightarrow_{\text{decompose}}$$
$$\left(\begin{array}{l}\alpha_1;\alpha_2;\alpha_3;\cdots;\alpha_{n-1};\alpha_n;\\ \alpha_{n-1}(\alpha_n);\\ \vdots\\ \alpha_3(\cdots(\alpha_{n-1}(\alpha_n)\cdots);\\ \alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n)\cdots));\\ \alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots)))\end{array}\right)$$

According to Definition 2, this decomposition causes another decomposition, that is,

$$\alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots))) \Longrightarrow_{\text{decompose}}$$
$$\left(\begin{array}{l}\alpha_{n-1}\models_{\text{of}}\alpha_n;\\ \alpha_n\models\alpha_{n-1};\\ \vdots\\ \alpha_3\models_{\text{of}}(\cdots(\alpha_{n-1}\models_{\text{of}}\alpha_n)\cdots);\\ (\cdots(\alpha_n\models\alpha_{n-1})\cdots)\models\alpha_3;\\ \alpha_2\models_{\text{of}}(\alpha_3\models_{\text{of}}(\cdots(\alpha_{n-1}\models_{\text{of}}\alpha_n)\cdots));\\ ((\cdots(\alpha_n\models\alpha_{n-1})\cdots)\models\alpha_3)\models\alpha_2;\\ \alpha_1\models_{\text{of}}(\alpha_2\models_{\text{of}}(\alpha_3\models_{\text{of}}(\cdots(\alpha_{n-1}\models_{\text{of}}\\ \qquad\alpha_n)\cdots)));\\ (((\cdots(\alpha_n\models\alpha_{n-1})\cdots)\models\alpha_3)\models\alpha_2)\models\alpha_1\end{array}\right)$$

[2] Deconstruction (we use the general term 'decomposition') means, for instance, a strategy of critical analysis (Jacques Derrida, 1930) directed towards exposing unquestioned metaphysical assumptions and internal contradictions in philosophical and literary language [10].

The first and the second informational system concerning decomposition in this consequence reveal together the informational complexity being hidden in the consequently serially embedded functional form $\alpha_1(\alpha_2(\alpha_3(\cdots(\alpha_{n-1}(\alpha_n))\cdots)))$. □

Consequence 11 [A Decomposition of Linear Informational Function] *Let an ordered set of informational items* α_i, *where* $i = 1, 2, \cdots, n$, *be denoted by*

$$\mathcal{A}_n^{\prec} = \{\alpha_1 \prec \alpha_2 \prec \cdots \prec \alpha_n\}$$

where operator symbol $\prec$ *has the role of an ordering comma. Let hold the following:*

ordered indexing:
$(\alpha_i \prec \alpha_j) \Longrightarrow (i < j)$;
transitivity of ordering:
$(\alpha_i \prec \alpha_j; \alpha_j \prec \alpha_k) \Longrightarrow (\alpha_i \prec \alpha_k)$;
subscript-entity difference:
$(i \neq j) \Longrightarrow (\alpha_i \neq \alpha_j)$

Then, for a function $\varphi(\alpha_1, \alpha_2, \cdots, \alpha_i)$, *where* $i \geq 1$ *and* $i \leq n$, *the following implication is determined:*

$$\begin{pmatrix} \varphi(\alpha_1, \alpha_2, \cdots, \alpha_i); \\ i \geq 1;\ i \leq n \end{pmatrix} \Longrightarrow_{\text{decompose}}$$

$$\left(\begin{array}{l} (\varphi(\xi);\ \xi \in \mathcal{A}_i^{\prec}); \\ \left(\begin{array}{l} \varphi(\xi_1, \xi_2); \\ \xi_1 \prec \xi_2; \\ \xi_1, \xi_2 \in \mathcal{A}_i^{\prec} \end{array} \right); \\ \vdots \\ \left(\begin{array}{l} \varphi(\eta_1, \eta_2, \cdots, \eta_i); \\ \eta_j \prec \eta_k; \\ j, k \in \{1, 2, \cdots, i\}; \\ \eta_1, \eta_2, \cdots, \eta_i \in \mathcal{A}_i^{\prec} \end{array} \right) \end{array} \right)$$

This scheme of decomposition delivers all possible linear functions of lengths $\ell = 1$ *to* $\ell = i$, *according to the ordered set* $\mathcal{A}_i^{\prec}$ *of operands* $\alpha_1, \alpha_2, \cdots, \alpha_i$. □

Proof. This kind of informational decomposition is customary in cases of metaphysical interpretation of phenomena, that is, in linear-decomposition scenarios belonging to metaphysically circular schemes and also elsewhere. The proof proceeds from the informational fact which, at least in the framework of a language, says the following: if a function depends on several informational entities, then it depends also on each of its arguments. Recursively, if a function depends on i arguments, then it can depend on all possible combinations, within an ordered set of arguments, say $\mathcal{A}_i^{\prec}$, on $i-1$ arguments. Such a relativity of decomposition is a consequence of an interpretational freedom, that is, possibility in an occurring situation (a part of the unforeseeable informational arising). □

10 Composition (Construction) of Informational Functionality

Composition or construction[3] can be understood as a reverse process to decomposition (deconstruction). If decomposition proceeds into details of a roughly determined informational situation by a process of interpretation, composition builds systems from the existing informational lumps (subsystems) and connects them informationally. Then, the result of this form of construction becomes a new entity carrying a new (characteristic) interpretation.

We can understand how decomposition and composition condition each other and, in some situations, it becomes impossible to distinguish which way represents the reason and which the consequence. There exists an informational game concerning both of them (deconstruction and construction) when entities (operands, formulas, formula systems) arise, emerge, or come into existence.

Interpretation (together with induction, evolution of entities, etc.) as a complex informational mechanism can include several known and unknown procedures e.g., substitution, insertion of a new or additional (parallel or serial) 'interpretation', introduction of circularity in regard to functions or functional arguments, spontaneity as a supplement of an unforeseeable entity to the existing informational situation, etc. Such views of decomposition and composition of

[3]Construction (we use the general term 'composition') means, for instance, a strategy of critical informational synthesis directed towards integrating unquestioned metaphysical assumptions and internal contradictions in any informational language.

informational systems concern their understanding. Several reasons for decomposition and composition interferences can exist in the form of informational-system-interior and informational-system-exterior entities. We must not forget that any informational system has the system concerning environment and it depends not only upon its own metaphysicalism, but also on system-disturbing external entities.

Consequence 12 [A Case of Parallel-serial Functional Composition] *From a well-connected parallel functional form*

$$\alpha_{||}(\alpha_1, \alpha_2, \cdots, \alpha_n)$$

in Definition 11, the following implicative composition from primitive parallel functions into sequential serial functions seems to be reasonable:

$$\begin{pmatrix} \alpha_1(\alpha_2); \\ \alpha_2(\alpha_3); \\ \vdots \\ \alpha_{n-2}(\alpha_{n-1}); \\ \alpha_{n-1}(\alpha_n) \end{pmatrix} \Longrightarrow_{\text{compose}}$$

$$\begin{pmatrix} \alpha_1(\alpha_2(\alpha_3)); \\ \alpha_1(\alpha_2(\alpha_3(\alpha_4))); \\ \vdots \\ \alpha_1(\alpha_2(\cdots(\alpha_{n-2}(\alpha_{n-1}))\cdots); \\ \alpha_1(\alpha_2(\cdots(\alpha_{n-2}(\alpha_{n-1}(\alpha_n)))\cdots) \end{pmatrix}$$

As we can see, the last composition was implemented by means of substitution. □

11 Informational Functionality of Metaphysical Cycles Impacted by an Exterior Entity α

In this section we turn our attention to the functionality which concerns metaphysical phenomena as functions of observing an external set of entities. An intelligent entity is, for example, metaphysical in observing its environment. The metaphysical is a regular property of any informational entity, regardless of its structure and organization. It has something in common with the entity existence. Existing means to be metaphysical in the sense to preserve (memorize, maintain, support) a certain structure and organization of the entity's intentionality, its informational functioning in the world. In this manner, the metaphysical of an entity is a standard property for which one can put the question: in which way is it standard?

In some previous papers [6, 7], one of the possible standards was proposed. This standard roots in a logical consideration which is closely connected with the nature of an informational entity. Such an entity is subjected to informational arising, which in a trivial case approaches to the state of an absolute stability of the entity's structure and organization. Otherwise the entity is arising together with its vanishing, which is only a particular case of the arising phenomenality.

As the reader may state, we distinguish three substantial phases (processes) of an entity's informational arising. This arising is not only a change, in the sense of modification, but also the coming of new information into existence. Changed and emerged informational pieces (lumps) have to be informationally connected to the existing body of the informing entity. We say, that the arisen items have to be informationally embedded and that through the process of embedding, in fact, informational entity has emerged to a different state in comparison to the previous one. This process of three subsequent phases is circularly (hermeneutically, viciously, investigational) closed, so the process of arising is reaching a satisfactory state by cycling, from informing, counterinforming, and embedding—and again in this way to a possible satisfaction.

What is the functionality of the metaphysical phenomenon belonging to an informing entity, which is informationally impacted by an exterior entity or set of entities? The impactedness may mean nothing else than the observing and vice versa. An entity ι is impacted by an outside entity α in the framework of ι's metaphysicalism. Roughly, $\alpha \models \iota$, where ι has to be metaphysically decomposed (deconstructed) in a serial circular way, to satisfy the possibilities of informational adequateness (equilibrium, satisfaction, semantics, etc.).

Let us take only one possible form of metaphysical cycle, which belongs to entity ι observing entity α. As we shall see later, one such form is sufficient for generating all possible metaphysical forms, that is, the so-called metaphysical gestalt belonging to ι observing α. So, let us set an ini-

tial form of possible standard metaphysical structures, in which components of informing, counterinforming and informational embedding appear in an cyclic serial form.

Definition 15 [A Standard Metaphysical Form and Its Functionalism] *Let the meaning of informational operands (entities) be the following:*

1. *Operand ι is an entity, which has to be cyclically decomposed as a metaphysical structure of informing, counterinforming, and informational embedding when observing α. This dependence can roughly be denoted by the functional form $\iota(\alpha)$. Thus, in a metaphysical situation,*

$$(\alpha \models \iota) \Longrightarrow \iota(\alpha)$$

2. *Operand α marks an exterior entity or a set of entities (impacting environment) in regard to ι. It functions as an independent informational variable of function ι. Thus,*

$$\alpha \subset \varepsilon_{\text{environment}}(\iota)$$

Environment $\varepsilon_{\text{environment}}(\iota)$ is the environment which can impact ι and is the only one which can be sensed (observed) by ι. For ι, other environment does not exist.

3. *Operand ι informs and is informed means that there exists the so-called informing component of ι being marked by $\mathcal{I}_\iota$. It is to understand that $\mathcal{I}_\iota$ means a function $\mathcal{I}(\iota)$ simultaneously. Being informationally involved in ι, a consequence of functionality $\iota(\alpha)$ is*

$$\iota(\alpha) \Longrightarrow \begin{pmatrix} \mathcal{I}_\iota(\alpha); \\ \mathcal{I}(\iota(\alpha)); \\ \mathcal{I}(\alpha, \iota) \end{pmatrix}$$

The first form depends solely on α. The second case is a nested functional dependence of rank 2. The third function linearly depends on both α and ι, where

$$\mathcal{I}(\alpha, \iota) \Longrightarrow_{\text{decompose}} \begin{pmatrix} \mathcal{I}(\alpha); \mathcal{I}(\iota); \\ \mathcal{I}(\alpha, \iota) \end{pmatrix}$$

4. *Operand ι informs and is informed means that there does not only exist the informing component $\mathcal{I}_\iota$, but also the counterinforming component $\mathcal{C}_\iota$. It is to understand that $\mathcal{C}_\iota$ means a function $\mathcal{C}(\mathcal{I}(\iota))$ simultaneously. Being informationally involved in ι and $\mathcal{I}_\iota$, a consequence of functionalities $\iota(\alpha)$ and $\mathcal{I}_\iota(\alpha)$ is*

$$(\iota(\alpha) \models \mathcal{I}_\iota(\alpha)) \Longrightarrow \begin{pmatrix} \mathcal{C}_\iota(\alpha); \\ \mathcal{C}(\mathcal{I}(\iota(\alpha))); \\ \mathcal{C}(\alpha, \iota, \mathcal{I})_{meta} \end{pmatrix}$$

The first case is a function, depending on α only. The second form is a nested functional case of rank 3. The third form is a linear function depending on three arguments, which can be decomposed according to Consequence 11, where

$$\mathcal{C}(\alpha, \iota, \mathcal{I}) \Longrightarrow_{\text{decompose}} \begin{pmatrix} \mathcal{C}(\alpha); \mathcal{C}(\iota); \mathcal{C}(\mathcal{I}); \\ \mathcal{C}(\alpha, \iota); \mathcal{C}(\alpha, \mathcal{I}); \\ \mathcal{C}(\iota, \mathcal{I}); \\ \mathcal{C}(\alpha, \iota, \mathcal{I}) \end{pmatrix}$$

5. *Operand ι informs and is informed means that there does not only exist the informing component $\mathcal{I}_\iota$ and the counterinforming component $\mathcal{C}_\iota$, but also the counterinformational component γ_ι. It is to understand that γ_ι means a function $\gamma(\mathcal{C}(\mathcal{I}(\iota)))$ simultaneously. Being informationally involved in ι, $\mathcal{I}_\iota$ and $\mathcal{C}_\iota$, a consequence of functionalities $\iota(\alpha)$, $\mathcal{I}_\iota(\alpha)$, and $\mathcal{C}_\iota(\alpha)$ is*

$$(((\iota(\alpha) \models \mathcal{I}_\iota(\alpha)) \models \mathcal{C}_\iota(\alpha)) \models \gamma_\iota(\alpha)) \Longrightarrow$$

$$\begin{pmatrix} \gamma_\iota(\alpha); \\ \gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha)))); \\ \gamma(\alpha, \iota, \mathcal{I}, \mathcal{C}) \end{pmatrix}$$

The first formula is a function of the exterior entity α. The second form is a nested functionality of rank 4. The third form is a linear functional case of four arguments for which a decomposition according to Consequence 11 can be realized, that is,

$$\gamma(\alpha, \iota, \mathcal{I}, \mathcal{C}) \Longrightarrow_{\text{decompose}}$$

$$\begin{pmatrix} \gamma(\alpha); \gamma(\iota); \gamma(\mathcal{I}); \gamma(\mathcal{C}); \\ \gamma(\alpha, \iota); \gamma(\alpha, \mathcal{I}); \gamma(\alpha, \mathcal{C}); \gamma(\iota, \mathcal{I}); \\ \quad \gamma(\iota, \mathcal{C}); \gamma(\mathcal{I}, \mathcal{C}); \\ \gamma(\alpha, \iota, \mathcal{I}); \gamma(\alpha, \mathcal{I}, \mathcal{C}); \gamma(\iota, \mathcal{I}, \mathcal{C}); \\ \gamma(\alpha, \iota, \mathcal{I}, \mathcal{C}) \end{pmatrix}$$

6. *Operand ι informs and is informed means that there does not only exist the informing component $\mathcal{I}_\iota$, the counterinforming component $\mathcal{C}_\iota$, and the counterinformational component γ_ι, but also the embedding component $\mathcal{E}_\iota$. It is to understand that $\mathcal{E}_\iota$ means a function $\mathcal{E}_\iota(\gamma(\mathcal{C}(\mathcal{I}(\iota))))$ simultaneously. Being informationally involved in ι, $\mathcal{I}_\iota$, $\mathcal{C}_\iota$, and γ_ι, a consequence of functionalities $\iota(\alpha)$, $\mathcal{I}_\iota(\alpha)$, $\mathcal{C}_\iota(\alpha)$ and γ_ι is*

$$\left(\begin{array}{l}(((\iota(\alpha) \models \mathcal{I}_\iota(\alpha)) \models \mathcal{C}_\iota(\alpha)) \models \gamma_\iota(\alpha)) \\ \quad \models \mathcal{E}_\iota(\alpha)\end{array}\right) \Longrightarrow$$

$$\left(\begin{array}{l}\mathcal{E}_\iota(\alpha); \\ \mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))); \\ \mathcal{E}(\alpha,\iota,\mathcal{I},\mathcal{C},\gamma)\end{array}\right)$$

The first formula is a function of the exterior entity α. The second form is a nested functionality of rank 5. The third form is a linear functional case of five arguments for which a decomposition according to Consequence 11 can be realized in the form

$$\mathcal{E}(\alpha,\iota,\mathcal{I},\mathcal{C},\gamma) \Longrightarrow_{\text{decompose}}$$

$$\left(\begin{array}{l}\mathcal{E}(\alpha);\mathcal{E}(\iota);\mathcal{E}(\mathcal{I});\mathcal{E}(\mathcal{C});\mathcal{E}(\gamma); \\ \mathcal{E}(\alpha,\iota);\mathcal{E}(\alpha,\mathcal{I});\mathcal{E}(\alpha,\mathcal{C});\mathcal{E}(\alpha,\gamma); \\ \quad \mathcal{E}(\iota,\mathcal{I});\mathcal{E}(\iota,\mathcal{C});\mathcal{E}(\iota,\gamma);\mathcal{E}(\mathcal{I},\mathcal{C}); \\ \quad \mathcal{E}(\mathcal{I},\gamma);\mathcal{E}(\mathcal{C},\gamma); \\ \mathcal{E}(\alpha,\iota,\mathcal{I});\mathcal{E}(\alpha,\iota,\mathcal{C});\mathcal{E}(\alpha,\iota,\gamma); \\ \quad \mathcal{E}(\alpha,\mathcal{I},\mathcal{C});\mathcal{E}(\alpha,\mathcal{I},\gamma);\mathcal{E}(\alpha,\mathcal{C},\gamma); \\ \quad \mathcal{E}(\iota,\mathcal{I},\mathcal{C});\mathcal{E}(\iota,\mathcal{I},\gamma);\mathcal{E}(\iota,\mathcal{C},\gamma); \\ \mathcal{E}(\alpha,\iota,\mathcal{I},\mathcal{C});\mathcal{E}(\alpha,\iota,\mathcal{I},\gamma);\mathcal{E}(\alpha,\iota,\mathcal{C},\gamma); \\ \quad \mathcal{E}(\iota,\mathcal{I},\mathcal{C},\gamma); \\ \mathcal{E}(\alpha,\iota,\mathcal{I},\mathcal{C},\gamma)\end{array}\right)$$

7. *Operand ι informs and is informed means that there does not only exist the informing component $\mathcal{I}_\iota$, the counterinforming component $\mathcal{C}_\iota$, the counterinformational component γ_ι, and the informing embedding component $\mathcal{E}_\iota$, but also the informational embedding component ε_ι. It is to understand that ε_ι means a function $\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota)))))$ simultaneously. Being informationally involved in ι, $\mathcal{I}_\iota$, $\mathcal{C}_\iota$, γ_ι, and ε_ι, a consequence of functionalities $\iota(\alpha)$, $\mathcal{I}_\iota(\alpha)$, $\mathcal{C}_\iota(\alpha)$, $gamma_\iota(\alpha)$, and $\mathcal{E}_\iota(\alpha)$ is*

$$\left(\begin{array}{l}((((\iota(\alpha) \models \mathcal{I}_\iota(\alpha)) \models \mathcal{C}_\iota(\alpha)) \models \gamma_\iota(\alpha)) \\ \quad \models \mathcal{E}_\iota(\alpha)) \models \varepsilon_\iota(\alpha)\end{array}\right) \Longrightarrow$$

$$\left(\begin{array}{l}\varepsilon_\iota(\alpha); \\ \varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha)))))); \\ \varepsilon(\alpha,\iota,\mathcal{I},\mathcal{C},\gamma,\mathcal{E})\end{array}\right)$$

The first formula is a function of the exterior entity α. The second form is a nested functionality of rank 6. The third form is a linear functional case of six arguments for which a decomposition according to Consequence 11 can be realized in the form

$$\varepsilon(\alpha,\iota,\mathcal{I},\mathcal{C},\gamma,\mathcal{E}) \Longrightarrow_{\text{decompose}}$$

$$\left(\begin{array}{l}\varepsilon(\alpha);\varepsilon(\iota);\varepsilon(\mathcal{I});\varepsilon(\mathcal{C});\varepsilon(\gamma);\varepsilon(\mathcal{E}); \\ \varepsilon(\alpha,\iota);\varepsilon(\alpha,\mathcal{I});\varepsilon(\alpha,\mathcal{C});\varepsilon(\alpha,\gamma); \\ \quad \varepsilon(\alpha,\mathcal{E});\varepsilon(\iota,\mathcal{I});\varepsilon(\iota,\mathcal{C});\varepsilon(\iota,\gamma); \\ \quad \varepsilon(\iota,\mathcal{E});\varepsilon(\mathcal{I},\mathcal{C});\varepsilon(\mathcal{I},\gamma);\varepsilon(\mathcal{I},\mathcal{E}); \\ \quad \varepsilon(\mathcal{C},\gamma);\varepsilon(\mathcal{C},\mathcal{E});\varepsilon(\gamma,\mathcal{E}); \\ \varepsilon(\alpha,\iota,\mathcal{I});\varepsilon(\alpha,\iota,\mathcal{C});\varepsilon(\alpha,\iota,\gamma); \\ \quad \varepsilon(\alpha,\iota,\mathcal{E});\varepsilon(\alpha,\mathcal{I},\mathcal{C});\varepsilon(\alpha,\mathcal{I},\gamma); \\ \quad \varepsilon(\alpha,\mathcal{I},\mathcal{E});\varepsilon(\alpha,\mathcal{C},\gamma);\varepsilon(\alpha,\mathcal{C},\mathcal{E}); \\ \quad \varepsilon(\alpha,\gamma,\mathcal{E});\varepsilon(\iota,\mathcal{I},\mathcal{C});\varepsilon(\iota,\mathcal{I},\gamma); \\ \quad \varepsilon(\iota,\mathcal{I},\mathcal{E});\varepsilon(\iota,\mathcal{C},\gamma);\varepsilon(\iota,\mathcal{C},\mathcal{E}); \\ \quad \varepsilon(\iota,\gamma,\mathcal{E});\varepsilon(\mathcal{I},\mathcal{C},\gamma);\varepsilon(\mathcal{I},\mathcal{C},\mathcal{E}); \\ \quad \varepsilon(\mathcal{I},\gamma,\mathcal{E});\varepsilon(\mathcal{C},\gamma,\mathcal{E}); \\ \varepsilon(\alpha,\iota,\mathcal{I},\mathcal{C});\varepsilon(\alpha,\iota,\mathcal{I},\gamma);\varepsilon(\alpha,\iota,\mathcal{I},\mathcal{E}); \\ \quad \varepsilon(\alpha,\iota,\mathcal{C},\gamma);\varepsilon(\alpha,\iota,\mathcal{C},\mathcal{E});\varepsilon(\alpha,\iota,\gamma,\mathcal{E}); \\ \quad \varepsilon(\iota,\mathcal{I},\mathcal{C},\gamma);\varepsilon(\iota,\mathcal{I},\mathcal{C},\mathcal{E});\varepsilon(\iota,\mathcal{I},\gamma,\mathcal{E}); \\ \quad \varepsilon(\mathcal{I},\mathcal{C},\gamma,\mathcal{E}); \\ \varepsilon(\alpha,\iota,\mathcal{I},\mathcal{C},\gamma,\mathcal{E})\end{array}\right)$$

8. *Function $\iota(\alpha)$ informs and is informed means that there does not only exist the informing component $\mathcal{I}_\iota$, the counterinforming component $\mathcal{C}_\iota$, the counterinformational component γ_ι, the informing embedding component $\mathcal{E}_\iota$, and the informational embedding component ε_ι, but also that function $\iota(\alpha)$ is, through these components, circularly and specifically closed into itself (informational metaphysicalism). It is to understand that $\iota(\alpha)$ means a function $\iota(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha)))))))$ simultaneously. Being informationally involved in ι, $\mathcal{I}_\iota$, $\mathcal{C}_\iota$, γ_ι, and ε_ι, a consequence of functionalities $\iota(\alpha)$, $\mathcal{I}_\iota(\alpha)$, $\mathcal{C}_\iota(\alpha)$, $gamma_\iota(\alpha)$, $\mathcal{E}_\iota(\alpha)$, and $\varepsilon_\iota(\alpha)$ is*

$$\left(\begin{array}{l}((((\iota(\alpha) \models \mathcal{I}_\iota(\alpha)) \models \mathcal{C}_\iota(\alpha)) \models \gamma_\iota(\alpha)) \\ \quad \models \mathcal{E}_\iota(\alpha)) \models \varepsilon_\iota(\alpha)) \models \iota(\alpha)\end{array}\right) \Longrightarrow$$

$$\left(\begin{array}{l}\iota^{\circlearrowleft}_{\mathrm{meta}}(\alpha); \\ \iota(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha)))))); \\ \iota(\alpha, \iota, \mathcal{I}, \mathcal{C}, \gamma, \mathcal{E}, \varepsilon)\end{array}\right)$$

where

$$\iota^{\circlearrowleft}_{\mathrm{meta}}(\alpha) \Longrightarrow$$
$$\iota^{\circlearrowleft}(\alpha)^*(\mathcal{I}_\iota(\alpha), \mathcal{C}_\iota(\alpha), \gamma_\iota(\alpha), \mathcal{E}_\iota(\alpha), \varepsilon_\iota(\alpha));$$
$$\iota(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))))) \Longrightarrow$$
$$\iota^{\circlearrowleft}(\alpha)^*(\varepsilon_\iota(\alpha)(\mathcal{E}_\iota(\alpha)(\gamma_\iota(\alpha)(\mathcal{C}_\iota(\alpha)(\mathcal{I}_\iota(\alpha)))))));$$

and the linear circular decomposition is

$$\iota(\alpha, \iota, \mathcal{I}, \mathcal{C}, \gamma, \mathcal{E}, \varepsilon) \Longrightarrow_{\mathrm{decompose}}$$
$$\left(\begin{array}{l}\iota^{\circlearrowleft}(\alpha); \iota^{\circlearrowleft}(\mathcal{I}); \iota^{\circlearrowleft}(\mathcal{C}); \iota^{\circlearrowleft}(\gamma); \iota^{\circlearrowleft}(\mathcal{E}); \\ \quad \iota^{\circlearrowleft}(\varepsilon); \\ \iota^{\circlearrowleft}(\alpha, \mathcal{I}); \iota^{\circlearrowleft}(\alpha, \mathcal{C}); \iota^{\circlearrowleft}(\alpha, \gamma); \\ \quad \iota^{\circlearrowleft}(\alpha, \mathcal{E}); \iota^{\circlearrowleft}(\alpha, \varepsilon); \iota^{\circlearrowleft}(\mathcal{I}, \mathcal{C}); \\ \quad \iota^{\circlearrowleft}(\mathcal{I}, \gamma); \iota^{\circlearrowleft}(\mathcal{I}, \mathcal{E}); \iota^{\circlearrowleft}(\mathcal{I}, \varepsilon); \\ \quad \iota^{\circlearrowleft}(\mathcal{C}, \gamma); \iota^{\circlearrowleft}(\mathcal{C}, \mathcal{E}); \iota^{\circlearrowleft}(\mathcal{C}, \varepsilon); \\ \quad \iota^{\circlearrowleft}(\gamma, \mathcal{E}); \iota^{\circlearrowleft}(\gamma, \varepsilon); \iota^{\circlearrowleft}(\mathcal{E}, \varepsilon); \\ \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{C}); \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \gamma); \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{E}); \\ \quad {}^{\circlearrowleft}(\alpha, \mathcal{I}, \varepsilon); \iota^{\circlearrowleft}(\alpha, \mathcal{C}, \gamma); \iota^{\circlearrowleft}(\alpha, \mathcal{C}, \mathcal{E}); \\ \quad {}^{\circlearrowleft}(\alpha, \mathcal{C}, \varepsilon); \iota^{\circlearrowleft}(\alpha, \gamma, \mathcal{E}); \iota^{\circlearrowleft}(\alpha, \gamma, \varepsilon); \\ \quad \iota^{\circlearrowleft}(\alpha, \mathcal{E}, \varepsilon); \iota^{\circlearrowleft}(\mathcal{I}, \mathcal{C}, \gamma); \iota^{\circlearrowleft}(\mathcal{I}, \mathcal{C}, \mathcal{E}); \\ \quad {}^{\circlearrowleft}(\mathcal{I}, \mathcal{C}, \varepsilon); \iota^{\circlearrowleft}(\mathcal{I}, \gamma, \mathcal{E}); \iota^{\circlearrowleft}(\mathcal{I}, \gamma, \varepsilon); \\ \quad \iota^{\circlearrowleft}(\mathcal{I}, \mathcal{E}, \varepsilon); \iota^{\circlearrowleft}(\mathcal{C}, \gamma, \mathcal{E}); \iota^{\circlearrowleft}(\mathcal{C}, \gamma, \varepsilon); \\ \quad \iota^{\circlearrowleft}(\mathcal{C}, \mathcal{E}, \varepsilon); \iota^{\circlearrowleft}(\gamma, \mathcal{E}, \varepsilon); \\ \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{C}, \gamma); \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{C}, \mathcal{E}); \\ \quad \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{C}, \varepsilon); \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \gamma, \mathcal{E}); \\ \quad \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \gamma, \varepsilon); \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{E}, \varepsilon); \\ \quad \iota^{\circlearrowleft}(\mathcal{I}, \mathcal{C}, \gamma, \mathcal{E}); \iota^{\circlearrowleft}(\mathcal{I}, \mathcal{C}, \gamma, \varepsilon); \\ \quad \iota^{\circlearrowleft}(\mathcal{C}, \gamma, \mathcal{E}, \varepsilon); \\ \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{C}, \gamma, \mathcal{E}); \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{C}, \gamma, \varepsilon); \\ \iota^{\circlearrowleft}(\alpha, \mathcal{I}, \mathcal{C}, \gamma, \mathcal{E}, \varepsilon)\end{array}\right)$$

By items 1–8, the metaphysical scenario of an entity ι, being informed by an exterior entity α, is implicatively standardized. Thus, further metaphysical interpretations of discussed functions are possible. □

Consequence 13 [A Standard, Functional, and Circular Metaphysical Hierarchy of an Informational Entity] *As a consequence of Definition 15, the parallel functions*

$$\begin{array}{l}\iota(\alpha); \\ \mathcal{I}(\iota(\alpha)); \\ \mathcal{C}(\mathcal{I}(\iota(\alpha))); \\ \gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha)))); \\ \mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))); \\ \varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha)))))); \\ \iota(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))))))\end{array}$$

form a functional hierarchy in the following sense:

- *Entity $\iota(\alpha)$ as such is a function of exterior entity α.*
- *Informing $\mathcal{I}(\iota(\alpha))$ which depicts the intentional character of entity ι, depends on α and preserves the informational contents of ι.*
- *Counterinforming $\mathcal{C}(\mathcal{I}(\iota(\alpha)))$ arises as the informing within the informing $\mathcal{I}(\iota(\alpha))$ in a spontaneous manner, producing entity γ.*
- *Counterinformational entity $\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))$ is a free, informationally unconnected product of $\mathcal{C}(\mathcal{I}(\iota(\alpha)))$ and will become an object of the so-called embedding in the framework of entity $\iota(\alpha)$.*
- *Embedding $\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha)))))$ spontaneously observes the counterinformational (arisen, unconnected) entity $\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))$ with the aim to produce an adequate embedding (connecting) informational entity ε, by which γ will become an informational part of entity $\iota(\alpha)$.*
- *By the embedding produced embedding informational entity $\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))))$ is a free informational product of $\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))))$ by which the arisen counterinformational entity $\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))$ is appropriately embedded into $\iota(\alpha)$. Through this informational connection, to entity $\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))$ a sense or meaning within $\iota(\alpha)$ is granted.*
- *The described hierarchy of entities ι, $\mathcal{I}$, $\mathcal{C}$, γ, $\mathcal{E}$, and ε is called a standard functional and circular metaphysical hierarchy within entity ι. This hierarchy is circular in the functional sense of*

$$\iota\,\boxed{(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(}_{\Omega}\ \iota(\alpha)\ \boxed{))))))}_{\Omega}$$

where

$$\iota\ \Omega(\varepsilon,\mathcal{E},\gamma,\mathcal{C},\mathcal{I})\ \iota(\alpha)\ \Omega_{)};$$

$$\Omega(\varepsilon,\mathcal{E},\gamma,\mathcal{C},\mathcal{I}) \rightleftharpoons \boxed{(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(}\ ;$$

$$\Omega_{)} \rightleftharpoons \boxed{)))))}$$

and Ω is the general marker for the so-called functional frame (a framed functional part in contrary to the framed informing part, marked by Φ and Ψ).

Embedding masters all the components of the metaphysical cycle and counterinforming emerges from the present state of the circularly arising entity $\iota(\alpha)$. □

12 Some Metaphysical Gestalts Concerning the Informational Being-of

A *gestalt* is an informational whole belonging to a particular informational entity. In this sense, an informational entity is an informational part of the *whole* (arising system, unity, entirety), which is called the entity's gestalt. Gestalt is in no way a final result (like category) and arises together with the involved informational entity. A gestalt of an entity means that the appearance of the entity pulls to the gestalt belonging other entities into existence (e.g. possible and various forms of a metaphysical cycle). These entities can be understood as visible and invisible possibilities of informing of the original entity and they can be identified, for instance, solely by syntactic modifications of formulas in which the sequence of the occurring operands and operators remains unchanged and only parentheses in formulas are differently and adequately replaced in all possible ways. In other cases, to a gestalt of an informational formula, modified formulas can belong, in which the operand/operator sequence is preserved, but some of the operands and with them connected operators can be let out. Such cases can become senseful especially in the framework of metaphysically (circularly) structured informational systems.

On the other hand, the informational gestalt can as well concern the so-called semantic problems, in which the so-called interpretative decomposition and composition of a formula or formula system come into question. In the sequel, some metaphysical cases of gestalts will be presented. A metaphysical gestalt is particularly structured (e.g. standardized, characterized) and can be easily recognized even when its components are altered and structurally differently connected and when standard metaphysical components (informing, counterinforming, and embedding) occur in multiple variations. These variations follow an informational intention within a dynamic (changing, emerging) existence of an informing entity. A general theory of informational gestalt will be the subject of a separate study.

Definition 16 [Two Kinds of Metaphysically Informing Gestalts of an Informational Entity] *In comparison to an informational frame, the informing gestalt $\Gamma_{\models}$ is a higher informational structure which, to some extent, determines possible frames within an informational formula. In case $\alpha \models \iota$, the gestalt of the metaphysically informing entity ι observing α is*

$$\left(\begin{array}{l}\Gamma_{\models}(\iota^{\circlearrowleft}(\alpha)^{*}(\mathcal{I}_{\iota}(\alpha) \prec \mathcal{C}_{\iota}(\alpha) \prec \gamma_{\iota}(\alpha) \prec \\ \quad \mathcal{E}_{\iota}(\alpha) \prec \varepsilon_{\iota}(\alpha)))\end{array}\right) \rightleftharpoons$$

$$\left(\begin{array}{l}\left(\begin{array}{l}\Phi(\mathcal{I}_{\iota}(\alpha) \prec \mathcal{C}_{\iota}(\alpha) \prec \gamma_{\iota}(\alpha) \prec \\ \quad \mathcal{E}_{\iota}(\alpha) \prec \varepsilon_{\iota}(\alpha))\end{array}\right) \models_{\forall} \\ \quad \left(\begin{array}{l}\Subset \iota(\alpha) \\ \quad \left(\begin{array}{l}\Phi(\mathcal{I}_{\iota}(\alpha) \prec \mathcal{C}_{\iota}(\alpha) \prec \gamma_{\iota}(\alpha) \prec \\ \quad\ \mathcal{E}_{\iota}(\alpha) \prec \varepsilon_{\iota}(\alpha))\end{array}\right) \\ \quad \iota(\alpha) \Supset\end{array}\right)\end{array}\right)$$

Operator $\models_{\forall}$ reads 'inform(s) for all' and operator $\prec$ as 'precede(s)'. Expression $\mathcal{I}_{\iota}(\alpha) \prec \mathcal{C}_{\iota}(\alpha) \prec \gamma_{\iota}(\alpha) \prec \mathcal{E}_{\iota}(\alpha) \prec \varepsilon_{\iota}(\alpha)$ within the functional and set notation is used instead of a series $\mathcal{I}_{\iota}(\alpha) \prec \mathcal{C}_{\iota}(\alpha)$; $\mathcal{C}_{\iota}(\alpha) \prec \gamma_{\iota}(\alpha)$; $\gamma_{\iota}(\alpha) \prec \mathcal{E}_{\iota}(\alpha)$; $\mathcal{E}_{\iota}(\alpha) \prec \varepsilon_{\iota}(\alpha)$, where $\prec$ is understood to be a transitive operator, that is, $\mathcal{I}_{\iota}(\alpha) \prec \gamma_{\iota}(\alpha)$; $\mathcal{I}_{\iota}(\alpha) \prec \mathcal{E}_{\iota}(\alpha)$; $\mathcal{C}_{\iota}(\alpha) \prec \mathcal{E}_{\iota}(\alpha)$; etc.

If we introduce, according to Consequence 11, the linearly ordered metaphysical set of components,

$$\mathcal{M}_5^{\prec} = \{\mathcal{I}_{\iota}(\alpha) \prec \mathcal{C}_{\iota}(\alpha) \prec \gamma_{\iota}(\alpha) \prec \mathcal{E}_{\iota}(\alpha) \prec \varepsilon_{\iota}(\alpha)\}$$

then all possible frames including 1, 2, 3, 4, and 5 metaphysical components and all possible combinations of the parenthesis pairs '(' and ')' are

determined, in the sense of Consequence 11, by the scheme (scenario) of the implicative decomposition, which is

$$\left(\begin{array}{l}\Phi(\mathcal{I}_\iota(\alpha) \prec \mathcal{C}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \\ \quad \mathcal{E}_\iota(\alpha) \prec \varepsilon_\iota(\alpha))\end{array}\right) \Longrightarrow_{\text{decompose}}$$

$$\left(\begin{array}{l}(\Phi(\xi);\ \xi \in \mathcal{M}_5^{\prec});\\ \left(\begin{array}{l}\Phi(\xi_1,\xi_2);\\ \xi_1 \prec \xi_2;\\ \xi_1,\xi_2 \in \mathcal{M}_5^{\prec}\end{array}\right);\\ \left(\begin{array}{l}\Phi(\xi_1,\xi_2,\xi_3);\\ \xi_1 \prec \xi_2 \prec \xi_3;\\ \xi_1,\xi_2,\xi_3 \in \mathcal{M}_5^{\prec}\end{array}\right);\\ \left(\begin{array}{l}\Phi(\xi_1,\xi_2,\xi_3,\xi_4);\\ \xi_1 \prec \xi_2 \prec \xi_3 \prec \xi_4;\\ \xi_1,\xi_2,\xi_3,\xi_4 \in \mathcal{M}_5^{\prec}\end{array}\right);\\ \Phi(\mathcal{I}_\iota(\alpha) \prec \mathcal{C}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \varepsilon_\iota(\alpha))\end{array}\right)$$

The inverse gestalt of the metaphysically informing entity ι, which observes α, is

$$\left(\begin{array}{l}\Gamma_{\models}(\iota^{\circ}(\alpha) * (\mathcal{I}_\iota(\alpha) \prec \mathcal{C}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \\ \quad \mathcal{E}_\iota(\alpha) \prec \varepsilon_\iota(\alpha))\end{array}\right) \rightleftharpoons$$

$$\left(\begin{array}{l}\left(\begin{array}{l}\Phi(\varepsilon_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \\ \quad \mathcal{C}_\iota(\alpha) \prec \mathcal{I}_\iota(\alpha))\end{array}\right) \models_\forall \\ \quad \left(\begin{array}{l}\Subset \iota(\alpha)\\ \quad \left(\begin{array}{l}\Phi(\varepsilon_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \\ \quad \mathcal{C}_\iota(\alpha) \prec \mathcal{I}_\iota(\alpha))\end{array}\right)\\ \quad \iota(\alpha) \Supset\end{array}\right)\end{array}\right)$$

If we introduce, according to Consequence 11, the inverse, linearly ordered metaphysical set of components in comparison to $\mathcal{M}_5^{\prec}$, that is the inverse set

$$\mathcal{M}_5^{\prec^{-1}} = \{\varepsilon_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \mathcal{C}_\iota(\alpha) \prec \mathcal{I}_\iota(\alpha)\}$$

then all possible inverted frames (in fact, Φ^{-1}) including 1, 2, 3, 4, and 5 metaphysical components (in the reverse order) and all possible combinations of the parenthesis pairs '(' and ')' are determined, in the sense of Consequence 11, by the scheme (scenario) of the inverse implicative decomposition, which is

$$\left(\begin{array}{l}\Phi(\varepsilon_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \\ \quad \mathcal{C}_\iota(\alpha) \prec \mathcal{I}_\iota(\alpha))\end{array}\right) \Longrightarrow_{\text{decompose}}$$

$$\left(\begin{array}{l}((\Phi(\xi);\ \xi \in \mathcal{M}_5^{\prec^{-1}});\\ \left(\begin{array}{l}\Phi(\xi_1,\xi_2);\\ \xi_1 \prec \xi_2;\\ \xi_1,\xi_2 \in \mathcal{M}_5^{\prec^{-1}}\end{array}\right);\\ \left(\begin{array}{l}\Phi(\xi_1,\xi_2,\xi_3);\\ \xi_1 \prec \xi_2 \prec \xi_3;\\ \xi_1,\xi_2,\xi_3 \in \mathcal{M}_5^{\prec^{-1}}\end{array}\right);\\ \left(\begin{array}{l}\Phi(\xi_1,\xi_2,\xi_3,\xi_4);\\ \xi_1 \prec \xi_2 \prec \xi_3 \prec \xi_4;\\ \xi_1,\xi_2,\xi_3,\xi_4 \in \mathcal{M}_5^{\prec^{-1}}\end{array}\right);\\ \Phi(\varepsilon_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \mathcal{C}_\iota(\alpha) \prec \mathcal{I}_\iota(\alpha))\end{array}\right)$$

We see that

$$\Phi(\mathcal{I}_\iota(\alpha) \prec \mathcal{C}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \varepsilon_\iota(\alpha)) =$$
$$\Phi^{-1}(\varepsilon_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \mathcal{C}_\iota(\alpha) \prec \mathcal{I}_\iota(\alpha))$$

and vice versa is the correspondence between the original and inverted case of informational frames. □

Consequence 14 [In a Standard Way Informing Metaphysical Gestalt of an Informational Entity] *The appearance of a standard metaphysical form, with components of informing, counterinforming and informational embedding, implies the occurrence of all possible ordered cycles (operator $\prec$) in one and the other direction, that is, from informing to embedding and also vice versa. There is,*

$$\left(\begin{array}{l}(((((\iota(\alpha) \models \mathcal{I}_\iota(\alpha)) \models \mathcal{C}_\iota(\alpha)) \models \gamma_\iota(\alpha)) \\ \quad \models \mathcal{E}_\iota(\alpha)) \models \varepsilon_\iota(\alpha)) \models \iota(\alpha)\end{array}\right) \Longrightarrow$$

$$\left(\begin{array}{l}\left(\begin{array}{l}\Subset \iota(\alpha)\\ \quad \left(\begin{array}{l}\Phi(\mathcal{I}_\iota(\alpha) \prec \mathcal{C}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \\ \quad \mathcal{E}_\iota(\alpha) \prec \varepsilon_\iota(\alpha))\end{array}\right)\\ \quad \iota(\alpha) \Supset\end{array}\right);\\ \left(\begin{array}{l}\Subset \iota(\alpha)\\ \quad \left(\begin{array}{l}\Phi(\varepsilon_\iota(\alpha) \prec \mathcal{E}_\iota(\alpha) \prec \gamma_\iota(\alpha) \prec \\ \quad \mathcal{C}_\iota(\alpha) \prec \mathcal{I}_\iota(\alpha))\end{array}\right)\\ \quad \iota(\alpha) \Supset\end{array}\right)\end{array}\right)$$

The informing entity after (under) the implication operator $\Longrightarrow$ is a part of the so called metaphysical gestalt of informing of entity (operand) $\iota(\alpha)$. □

Consequence 15 [A Functional Metaphysical Gestalt of an Informational Entity] *The next question concerns the so-called functional metaphysical gestalt of an informing entity. If, in general, $\varphi^*\xi$ marks the traditional functional notation $\varphi(\xi)$, then, considering Consequence 13, the possible functions, within a metaphysical functional gestalt, are*

$\iota^*\alpha$;

$\mathcal{I}^*\iota(\alpha)$; $\mathcal{I}(\iota)^*\alpha$;

$\mathcal{C}^*\mathcal{I}(\iota(\alpha))$; $\mathcal{C}(\mathcal{I})^*\iota(\alpha)$;
$\mathcal{C}(\mathcal{I}(\iota))^*\alpha$;

$\gamma^*\mathcal{C}(\mathcal{I}(\iota(\alpha)))$; $\gamma(\mathcal{C})^*\mathcal{I}(\iota(\alpha))$;
$\gamma(\mathcal{C}(\mathcal{I})^*\iota(\alpha)$; $\gamma(\mathcal{C}(\mathcal{I}(\iota)))^*\alpha)$;

$\mathcal{E}^*\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))$; $\mathcal{E}(\gamma)^*\mathcal{C}(\mathcal{I}(\iota(\alpha)))$;
$\mathcal{E}(\gamma(\mathcal{C}))^*\mathcal{I}(\iota(\alpha))$; $\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I})))^*\iota(\alpha)$;
$\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota))))^*\alpha$;

$\varepsilon^*\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha)))))$; $\varepsilon(\mathcal{E})^*\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))$;
$\varepsilon(\mathcal{E}(\gamma))^*\mathcal{C}(\mathcal{I}(\iota(\alpha)))$; $\varepsilon(\mathcal{E}(\gamma(\mathcal{C})))^*\mathcal{I}(\iota(\alpha))$;
$\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}))))^*\iota(\alpha)$; $\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota)))))^*\alpha$;

$\iota^*\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))))$; $\iota(\varepsilon)^*\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))))$;
$\iota(\varepsilon(\mathcal{E}))^*\gamma(\mathcal{C}(\mathcal{I}(\iota(\alpha))))$; $\iota(\varepsilon(\mathcal{E}(\gamma)))^*\mathcal{C}(\mathcal{I}(\iota(\alpha)))$;
$\iota(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}))))^*\mathcal{I}(\iota(\alpha))$; $\iota(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I})))))^*\iota(\alpha)$;
$\iota(\varepsilon(\mathcal{E}(\gamma(\mathcal{C}(\mathcal{I}(\iota)))))^*\alpha$

For the inverse metaphysical functional gestalt, the inverse functions, as

$\iota^*\alpha$;

$\varepsilon^*\iota(\alpha)$; $\varepsilon(\iota)^*\alpha$;

$\mathcal{E}^*\varepsilon(\iota(\alpha))$; $\mathcal{E}(\varepsilon)^*\iota(\alpha)$;
$\mathcal{E}(\varepsilon(\iota))^*\alpha$;

$\gamma^*\mathcal{E}(\varepsilon(\iota(\alpha)))$; $\gamma(\mathcal{E})^*\varepsilon(\iota(\alpha))$;
$\gamma(\mathcal{E}(\varepsilon)^*\iota(\alpha)$; $\gamma(\mathcal{E}(\varepsilon(\iota)))^*\alpha)$;

$\mathcal{C}^*\gamma(\mathcal{E}(\varepsilon(\iota(\alpha))))$; $\mathcal{C}(\gamma)^*\mathcal{E}(\varepsilon(\iota(\alpha)))$;
$\mathcal{C}(\gamma(\mathcal{E}))^*\varepsilon(\iota(\alpha))$; $\mathcal{C}(\gamma(\mathcal{E}(\varepsilon)))^*\iota(\alpha)$;
$\mathcal{C}(\gamma(\mathcal{C}(\varepsilon(\iota))))^*\alpha$;

$\mathcal{I}^*\mathcal{C}(\gamma(\mathcal{E}(\varepsilon(\iota(\alpha))))))$; $\mathcal{I}(\mathcal{C})^*\gamma(\mathcal{E}(\varepsilon(\iota(\alpha)))))$;
$\mathcal{I}(\mathcal{C}(\gamma))^*\mathcal{E}(\varepsilon(\iota(\alpha)))$; $\mathcal{I}(\mathcal{C}(\gamma(\mathcal{E})))^*\varepsilon(\iota(\alpha))$;
$\mathcal{I}(\mathcal{C}(\gamma(\mathcal{E}(\varepsilon))))^*\iota(\alpha)$; $\mathcal{I}(\mathcal{C}(\gamma(\mathcal{E}(\varepsilon(\iota)))))^*\alpha$;

$\iota^*\mathcal{I}(\mathcal{C}(\gamma(\mathcal{E}(\varepsilon(\iota(\alpha))))))$; $\iota(\mathcal{I})^*\mathcal{C}(\gamma(\mathcal{E}(\varepsilon(\iota(\alpha))))))$;
$\iota(\mathcal{I}(\mathcal{C}))^*\gamma(\mathcal{E}(\varepsilon(\iota(\alpha))))$; $\iota(\mathcal{I}(\mathcal{C}(\gamma)))^*\mathcal{E}(\varepsilon(\iota(\alpha)))$;
$\iota(\mathcal{I}(\mathcal{C}(\gamma(\mathcal{E}))))^*\varepsilon(\iota(\alpha))$; $\iota(\mathcal{I}(\mathcal{C}(\gamma(\mathcal{E}(\varepsilon)))))^*\iota(\alpha)$;
$\iota(\mathcal{I}(\mathcal{C}(\gamma(\mathcal{E}(\varepsilon(\iota)))))^*\alpha$

can come into consideration. □

13 Conclusion

Informational Being-of is only one of the keystones within the arising informational theory. Such a keystone is also the informational Being-in [9]. What might be important in the context of the differently appearing informational Being-possibilities, is the comparison between modernistic and postmodernistic understanding of informing of entities.

The informational formulas in this essay are not consequently (rigorously) informationally deduced, induced, and abduced. The presented theory of informational Being-of and it concerning informational processes is only at the beginning of a complete and elaborated theory. The presented context of the essay shows that what is already theoretically grasped in the sense of informational Being-of, but has to be developed into emerging possibilities.

Each open, postmodernistically structured theory is not only phenomenological (e.g. in the sense of the philosophy pertaining to Husserl and Heidegger), but phenomenalistic (e.g. in the sense of informational phenomenalism). Phenomenology does not create nor presuppose *logic constructions, theories or systems.* It does not deduce from axioms nor induce on the basis of observed and noted facts. Its method roots in an *exemplary intuition,* that is, investigating particular cases qua cases, which represent essences and types in the realm of consciousness [2].

Phenomenalistic theory is not only algorithmic and does not search for a principle of principles. It is aware that some initial principles are connected with informational principles of inference (reasoning, conclusion) which have to be seen as initial principles too (e.g., informational modus ponens, tollens, rectus, obliquus, operandi, vivendi, etc.) Principles of informing (understanding, interpreting, reasoning, coming into existence) are dynamic and they change and arise according to the emerging situations and attitudes. Such an informational theory can be understood as a predecessor of the informational machine which seems as a successor of the today computer. It becomes more and more evident that processes presented in this essay could be programmed on the most powerful (parallel, fast, and data-voluminous) computer systems. One of the most significant informational system will become the so-called knowledge

machine, being a natural, a logical, and a possible consequence of the today computer technology and informational philosophy.

MODERNISM	POSTMODERNISM
mathematization; algorithmization	informational philosophy and formalization
algorithmism; preceduralness	informationalism; inform. spontaneity
recursiveness	inform. circularity
mathematical formula system; computer program	informational formula; informational formula system (gestaltism)
mathematical function	informational Being-of; inform. functionalism
mathematical inclusion	informational Being-in; inform. inclusivism
mathematical formalism; philosophical phenomenology	informational externalism, internalism, metaphysicalism, and phenomenalism
algorithmic processing	informational arising (= informing)
algorithmic cycling; programmed recursion	metaphysical cycle with informing, counterinforming, and embedding
computer system	informational machine
expert system; knowledge base	knowledge machine; knowledge archives
artificial intelligence methodologies	metaphysicalism with informing, counterinforming and embedding
theory of chance, chaos, probability, fuzziness, etc.	counterinforming and informational embedding
deduction, induction, abduction, modus ponens	inform. decomposition, hermeneutics, interpretation, deconstruction, modi informationis
determinacy, predictability, closeness	indeterminacy, unpredictability, inform. openness

Table 1. Modernistic and postmodernistic terms

Table 1 (see [1, 8]) shows some essential differences between modernistic and postmodernistic orientation regarding traditional (mathematical, algorithmic) approaches and informational sense. This table might be helpful for a deeper understanding of informational phenomenalism as exposed in this essay in the form of informational Being-of. It is to stress that modernistic items can certainly be included into the conceptualism of postmodernistic means: such a position offers a substantial advantage on the way to informational methodology, formalism and machine.

References

[1] I. Hassan: *The Postmodern Turn,* Ohio State University Press, Columbus, Ohio, 1987.

[2] G. Ryle: *Martin Heidegger: "Sein und Zeit",* The Journal of the British Society for Phenomenology **1** (1970) No. 3, 3–14. [First published in Mind, N.S. **38** (1929) No. 151.]

[3] H. Suematsu: *Current Status of the EDR Electronic Dictionary Project,* Informatica **18** (1994) 93–96.

[4] A.P. Železnikar: *Towards an Informational Language,* Cybernetica **35** (1992) 139–158.

[5] A.P. Železnikar: *Formal Informational Principles,* Cybernetica **36** (1993) 43–64.

[6] A.P. Železnikar: *Metaphysicalism of Informing,* Informatica **17** (1993) 65–80.

[7] A.P. Železnikar: *Logos of the Informational,* Informatica **17** (1993) 245–266.

[8] A.P. Železnikar: *Towards an Informational Understanding of Knowledge,* Cybernetics and Systems '94 (Ed. R. Trappl), Vol. II, 1587–1594, World Scientific, Singapore (1994).

[9] A.P. Železnikar: *Informational Being-in,* Informatica **18** (1994) 149–171.

[10] *The Oxford English Dictionary,* Second Edition (on compact disc), Oxford University Press, Oxford, 1992.

[11] *EDR Electronic Dictionary Technical Guide,* TR–042, Japan Electronic Dictionary Research Institute, Ltd., Tokyo, 1993.

CAUSALITY AND THE THEORY OF INFORMATION

Leon Birnbaum
Bloc B6, App. 26
str. Gutinului 6
RO–4650 Dej
Romania

Keywords: action, causal chain, causality, cause-effect, event, informational chain, perturbation (constructive, dynamic, functional), state, system

Edited by: Anton P. Železnikar
Received: May 10, 1994 **Revised:** July 29, 1994 **Accepted:** August 29, 1994

This article deals with problems pertaining to the cause-effect phenomenalism, stressing the structure of the informationally transitional triad emitter-channel-receiver. In an informational-causal chain (see Fig. 1), perturbations (causes) and informations (effects) are considered. In the article presented notions can be a basis for a more exhaustive study of informational transitions basing on cause-effect philosophy. The graph in Fig. 1 presents an adequate starting-point for the further research of informational transition in the most complex manner.

1 Preliminaries

We must always start with the conviction that not all is definable. This statement refers to every kind of language. In formal languages, we do not define and never will define primary notions; the primary relations between the primary notions are formulated axiomatically. In natural languages, the situation is more complicated because these languages are in perpetual evolution and we can not always separate a finite set of primary words out of their vocabularies. Taking this into account, with intention to express the treatment of our subjects in a more explicit mode, we must introduce certain fundamental notions. It can happen that these notions are neither independently definable nor accepted by scientific communities or, that the scientific references include to much (and diverse) definitions of them. Thus, let us explain what do we understand by some basic notions in this article.

We introduce the following understanding:

- A *system* is a set through which one aprioristic relation with fixed features will be realized.
- A *state* is the situation of a determinate system, together with its structure and external conditions.
- An *event* is the intervened (and intervening) change, when passing from state a_1 at time t_1 to state a_2 at time t_2, where $t_2 > t_1$.
- An *action* is the intervention of system S_p into the natural autonomous unfolding of events by system S_q.
- An *informational chain* is a system of information transition, formed by the informational triad source-channel-receiver. This triad can interfere with other elements. E.g., perturbations can drive (impact) informational chains from both the exterior and the interior of the informational chains.
- A *perturbation* is an information which alters the transmitted information within the informational chain. A perturbation performs as an action.

An example of perturbation within the informational chain is the trite feedback.

2 A Classification of Actions

Let us introduce the following basic classification of sets of actions:

A. Static actions and

B. Dynamic actions.

In turn, static actions can be classified into more detail:

Aa. Actions of prevention. (Example: launching of signals.)

Ab. Actions of defense. (Example: action of burglary warning.)

Ac. Actions of influence. (Example: action of catalysis.)

Dynamic actions can be classified in the following way:

Ba. Stabilizers. (Example: the reverse connection.)

Bb. Functional destructive actions. (Examples: noises, catching diseases, virus intrusions, etc.)

Bc. Functional constructive actions. (Examples: sun radiation, atom mutations, etc.)

Each action manifests itself as information. Also, each action is realized as to owe some information. Therefore, a reciprocal implication exists, causing not an equivalence between action and information. (E.g., one implication holds for the release and the other for the action.) Each perturbation is considered to be an action (a perturbatious one).

3 Perturbations

According to the formerly said, the classification of perturbations presupposes the classification of actions. Up to now, as perturbations only those have been considered, generated by the dynamic functional destructive actions as noises, diaphones, distortions which represent informing perturbations, transmitted along the channels of electromagnetic and sonorous waves. As we have seen, there is an immense number of channels through which information can be transmitted. On the other hand, perturbations are not necessarily destructive (negative).

Another view of the perturbator action, accepted more or less tacitly, represents the fact that we have considered only those perturbations which impact the transmission channel. This concept does not correspond to the reality. Each element of the informational triad, that is, source-channel-receiver, can perform submissively as a perturbator action.

Let us motivate the presented arguments with some further examples:

a) Actions of the perturbations of the source are, for example, the well-known action of feedback, stimulation difference in intensity of the source, etc.

b) Actions of the perturbations of the channel are caused by electronic circuits of amplifiers, by the influence of the ionic equilibrium K-Na in cerebral synapses, etc.

c) Actions of the perturbations of the receiver are filters which select and distribute the received signals (information). The separation is performed in such a way that information is filtered out from noise and distortion.

4 Dynamic Functional Constructive Perturbations

In turn, these perturbations can be classified as continuous, periodical and irregular. Let us show some examples of certain classes of perturbations as follows:

a) As a continuous perturbation the force of the attraction of the great celestial bodies (stars, sun) can be understood. This force influences the celestial bodies at the traversing the fields of attraction of a star (according to the distances and masses); so, bodies will continuously deviate from their uniform and (apparently) rectilinear movement and, consequently, will join as planets around the star. Further two examples can be observed within the biological domain: In a successive manner, the contraction and relaxation of the thorax induce the possibility of the breathing and, implicitly, the oxygenation of the blood, which feeds by oxygen all the tissues. Another example is the metabolism of consummated substances, absorbed from the ambient medium.

b. Some examples of dynamic functional periodical perturbations are: the heterosexual

provocation of attraction in fauna at the period of rut, the menstrual cycle, the zodiacal influence, the migration of birds, the tides, etc.

c) More interesting seem to be the so-called irregular perturbations which can be further classified in the following way:

   c_1) By an individual action: infection of a human by AIDS through an individual contact.

   c_2) For a group of individuals (species): the genetic mutation which outcome was the extinguishing of the great saurians, 67–70 million years ago.

   c_3) By a historical action: migration of peoples, migration of insects (locusts), outbreak of wars, great epidemic diseases, etc.

   c_4) By a zonal action of a celestial body (Terra): desertification of great productive areas (forests, plantations), glaciations, floods, etc.

   c_5) By zonal action of the universe: that what is named the "gravitational collapse", appearance of the neutronic stars, etc.

A note. Concerning c_5), we must remark that there is only an apparent gravitational collapse. In the reality, matter is retransformed in inframatter (space) of which it is being constituted. If the speed of transition of matter in inframatter (space) is bigger as the speed of light, the impression arises that the force of attraction is so great that the matter attracts even the photons.

From the previous remarks it comes out that perturbations are single active elements concerning an introduction of changes (variations—according to J.C. Ducasse [3]) which within the habitual, the daily, the usual, and the universal are accepted as "normal". The same philosophy pertains to the term "cause". From this viewpoint the notion "cause" is partly equivalent to the notion "dynamic perturbation" in the sense that the set of causes is included in the set of dynamic perturbations, that is: "Each cause is a dynamic perturbation", if each effect presupposes a cause. Mario Bunge [2] asserts that each cause is an external action which leads to changes in an object (see [2], p. 15). We can therefore give an informational definition of the cause.

Definition 1 *The cause is a dynamic perturbation.*

This perturbation is usually external, but it can also be internal. The feedback does not operate directly on the source, but impacts perturbations which drive the source. However, not each dynamic perturbation represents a cause, because a dynamic perturbation can also induce an implication being different from cause. For instance, the birth implies the death, but the birth is not the cause of the death. To this, although the perturbations at this time are static, the catalysts imply the facilitation of chemical reactions, but they do not represent the cause which produces the chemical reactions. By conclusion, merely a subset of the dynamic perturbations represents the set of causes.

5 The Relation Cause-effect

The receiver of an informational chain receives a signal and, in the case of the existence of perturbations, the signal coming from the source is already altered. If the perturbation is a cause, then the altered signal in the receiver will induce a phenomenon, called the *effect,* that is, the effect being proper to the cause (to the perturbation) and not to the source. There is not a linear relation between the cause and the effect, but a relation of intervention, grafting, change of the signal emitted from the source. If the cause would not exist, the receiver would receive the signal as it is in the source (a unmodified one). The cause (perturbation) is responsible for the alternation of the signal. Several causes (perturbations) can exist which induce the same effect. Several mutually diverse diseases induce the same effect—the fever. Between the event when a cause induces an alternation of the signal and the event of the receiving an altered signal is a temporary disparity, a gap (see Section 1), because the signal from the source did not reach the receiver yet and must be altered in a later time; the signal must be disturbed by an action of the signal alternation. Further, we must consider that in no case the speed of signal transmission can outrun the speed of the light.

The relation cause-effect is therefore a functional relation of time.

However, among different observers there can also be some of them (e.g. in a movement) who, in the first turn, realize the effect while, later, realize the cause, overtaking thus a paradoxical situation. To avoid such paradoxical situation, the change of the observer place and also of the cause-effect relation must be reported to the same referential system being adequate for the relation cause-effect.

Namely, the causality, the generic set of relations cause-effect, is thus a direct principle of the action, but it remains also a principle of an external intervention (sometimes, internal) of a usual action (e.g., signal transmission).

Because our sphere of perception is anthropological (it spreads dimensionally from the atom to the solar system), the principle of causality ("Each effect has a cause") remains limited to our anthropological domains. For this reason, the anthropological principle of causality does not retain its validity beyond the boundaries of the anthropological. In quantum mechanics, as well as in galaxy mechanics, the anthropological principle of causality leads to a plurality of results and, so, becomes indeterminate. In the subatomic as well as in the overastral worlds there are phenomena and events (e.g., for these domains the existence of ununiform movements is proved) which demonstrate perturbations through information transmission and among perturbations some of them represent certain "causes". This may mean that also in the non-anthropological domains the principles of causality exist which differ comparatively from the anthropological principles of causality. We know (assume) several phenomena and interrelations belonging to such extra-anthropological worlds not directly, but through the instruments of investigation and measure possessing a high vim of resolution. Up to now, our knowledge concerning the extra-anthropological worlds is insufficient for a generalization of phenomena belonging to these worlds and for a formulation of norms, laws and principles, which would constitute a different logic (e.g., being non-bipolar). In any case, the causality—or more exactly causalities—of those worlds fundamentally differs from the anthropological causality. To the same conclusion, although by inference in another manner, comes Mario Bunge [2]. That which was discovered in the extra-anthropological worlds is rather implications than causalities, is rather results than regularities, and is rather effects than causes.

6 Confrontations

The problem of simultaneousness of cause and effect was contributed to the theory, more or less, by I. Kant [5]. We consider that the great thinker has referred to the causes being spread over a longer time period. A cause at time t' releases the effect at time t, where $t' < t$ and thus, in the time interval $t-t'$, the cause and the effect are concomitant.

Let us consider that c_1 and c_2 are causes and that $f(c_1)$ and $f(c_2)$ are effects of the causes. If c_1 precedes c_2, then $f(c_1)$ precedes $f(c_2)$. This is a regular succession between the cause and the effect. Concerning this regular succession, foreseen by Fr. Bacon [1], after him by D. Hume [4], and now by J.C. Ducasse [3], we do not dispose and, we consider that it is impossible to dispose, with the rigorous arguments for this conformation. We consider that the causes can be multiple ones and these induce effects in different time lapses.

Concerning the influence of causality upon the determinism in science and philosophy, even in reference to the anthropological causality, we consider that its impact is limited in a several viewpoints. Let us present some conclusive illustrations:

a) One and the same cause can produce different effects, even contradictory ones, being in the function of the intensity of the preceding signal: a feeble wing extinguishes a candle, a strong wind stirs a fire.

b) As we have seen before, there is not a guaranty for a regular and parallel succession between the cause and the effect. Although thunder and flash of lightning take place simultaneously, their effects (firing of a tree and breaking of window panes because of sonorous pressure) are staggered in time because of different speeds of propagation.

c) A drug, in different doses can have different effects, even the paradoxical and contradictory ones.

7 The Origin of Perturbations and Causes

Considering the origin of perturbations (and of causes too), they can be:

a) endosystemic, which proceed from the same system, the same informational chain;

b) ectosystemic, which proceed from another system (external to the information chain).

The endosystemic perturbations are usually constructively continuous or constructively periodical. In such cases, perturbations have a regulatory or programming role. For living creatures, these perturbations root in genetics specific for each kingdom, phylum and species.

The ectosystemic perturbations are, by themselves, effects of another informational chain (on the side of receiver) belonging to the same suprasystem; they can regulate the necessities of the suprasystem, etc.

An example of the suprasystem is that of the living organisms, the nature. For keeping of its internal equilibrium, that is, its existence (e.g., the ecological equilibrium), the nature intervenes by means of its possibilities by evolution or involution of different populations.

The universe is the only system in which all perturbations are endosystemic, since the universe is a unipolar notion without the exterior.

8 Final Considerations

The cause cannot be a generator of information; this function belongs to the perturbation of an already existing information. The effect is information proceeding from the source, being perturbed, altered, mutated by a cause. The cause, in itself, can also perform as an effect, being induced by another cause within another information.

Therefore, the *causal chain* has, as a support (skeleton), a succession of (perturbed) information. Also, each effect can serve for a cause in another relation cause-effect while each cause can serve for an effect too. In any case, each effect has a cause.

The *causal chain,* in which any element is an effect of a cause and any cause induces another effect, with exception of the first and the last element, is not linear but has a scalar form, as can be seen from Fig. 1.

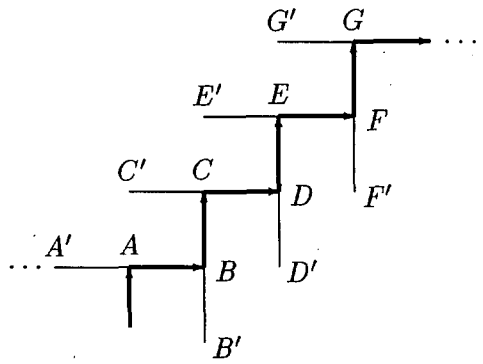


Figure 1: *The informational-causal chain: paths* $\overline{A'B}$, $\overline{B'C}$, $\overline{C'D}$, $\overline{D'E}$, $\overline{E'F}$, $\overline{F'G}$, *and* $\overline{G'\ldots}$ *are informational while paths* $\overline{AB}$, $\overline{BC}$, $\overline{CD}$, $\overline{DE}$, $\overline{EF}$, $\overline{FG}$, *and* $\overline{G\ldots}$ *are perturbational, that is, causal.*

In accordance to the revealed, the philosophical problem of the "first cause" and the "last purpose" (considering the purpose an effect of an effect of cause) are therefore deprived of the sense. Each cause is both the first cause and the last cause concerning its capacity of perturbing, changing, mutating certain information. The problem of the first cause dates in the antiquity, with the purpose to stop the regressive succession of the "why's?". It has been agreed to equalize the "first cause" with a superior power, attributed to one (or more) supernatural creature(s), or to God. Not the first cause must be attributed to God, but the first information (exactly, the first informations). The first perturbations (causes) are due to the interference of the first informations.

We have to admire the deep intuition of the Evangelist St. John: "At the beginning was the Word, and the Word was with God." (Gospel, John, 1,1.)

References

[1] Fr. Bacon: *Novum Organon,* I, af. 68, 1620.

[2] M. Bunge: *Causality. The Place of the Causal Principle in Modern Science*, Cambridge, MA, 1959 (p. 15 and 31).

[3] J.C. Ducasse: *The Nature of Causality*, Edited by M. Brand, IL, 1976.

[4] D. Hume: *A Treatise of Human Nature*, Oxford at the Claredon Press, 1958 (p. 77).

[5] I. Kant: *The Critic of the Pure Reason*, Part II, Division 2, Book 2, Chapter II, Section 2, Analogy B.

CRITICAL ANALYSIS OF ROUGH SETS APPROACH TO MACHINE LEARNING

Igor Kononenko, Samo Zorc
University of Ljubljana, Faculty of electrical engineering & computer science,
Tržaška 25, SI-61001 Ljubljana, Slovenia
Phone: +386 61 1768390, Fax: +386 61 264990
e-mail: igor.kononenko@ninurta.fer.uni-lj.si

Keywords: rough sets theory, critical analysis, machine learning

Edited by: Rudi Murn
Received: March 9, 1994 **Revised:** June 28, 1994 **Accepted:** October 5, 1994

In this paper the rough set theory (RST) approach to machine learning is analysed and its drawbacks described. RST often uses complicated formalization of rather simple notions and sometimes invents new notions that make the RST papers hard to read and understand. Some authors from the RST community tend to ignore the huge amount of work done in machine learning. This may lead to reinventings and ad-hoc solutions.

1 Introduction

Pawlak (1982) defined the rough sets theory (RST) which was later used for several applications and published in a series of papers (e.g. Pawlak, 1984; Pawlak et al., 1986). However, Wong et al. (1986) compared the RST approach with ID3 approach (Quinlan, 1979; 1986) and concluded that the inductive learning defined by Pawlak is just a special case of the ID3 approach and basically differs in some (unrealistic) assumptions. To overcome the deficiencies Wong & Ziarko (1986) defined the probabilistic RST which was later refined by Pawlak et al. (1988). The redefined probabilistic RST seems to eliminate some drawbacks of the original (discrete) RST, while still containing some problematic issues.

Later, numerous papers were published on the application of RST, which almost all use the original definition of (discrete) RST (with all its drawbacks). The most notable publications seem to be the paper (Kubat, 1991), the book by Pawlak (1991), and the edition of 27 papers in (Slowinski, 1992). Kubat ends his paper with "It is a pity that this topic (RST) has not yet received more publicity ...". Maybe the more appropriate claim would be: "It is a pity that nobody critically analysed the RST and compared the performance of RST with well known inductive learning approaches". The aim of this paper is to fill this gap.

In the next section the RST approach to machine learning is briefly described. We describe the basic terminology and definitions to give the reader the impression about the RST approach, however, in the RST literature there are many more definitions and terminology which often confuses the point. In section 3 we discuss the deficiencies of the RST approach: its formal and unreadable terminology, inflexible knowledge representation, and ad-hoc solutions. Section 4 describes experimental comparison of performance of two "classical" machine learning algorithms with the performance of RST. In conclusion we analyse the "contribution" of RST to machine learning.

2 Rough sets theory

2.1 Basic definitions

The (discrete) RST (Pawlak, 1982) is introduced through an *information system* which is defined as a 4-tuple $S =< U, Q, V, f >$, where U is a finite set of objects, Q is a finite set of attributes, V is the union of attributes domains and $f : U \times Q \rightarrow V$ is a total function that assigns a value to each attribute of each object. This function is used to define equivalence rela-

tion called *indiscernibility relation* for each subset of attributes $P \subset Q$:

$$\widetilde{P} = \{(x,y)|x,y \in U \ \& \ \forall q \in P : f(x,q) = f(y,q)\} \quad (1)$$

In other words, objects x and y are in the relation $\widetilde{P}$ if they have the same value for every attribute from P. The family of equivalence classes of relation $\widetilde{P}$ is denoted by P^*.

A *rough set* is defined as an approximation of set of objects $Y \subset U$ and is defined with two unions of equivalence classes of objects (Pawlak, 1982):

$$\underline{P}Y = \bigcup\{X \in P^*|X \subseteq Y\} \quad (2)$$
$$\overline{P}Y = \bigcup\{X \in P^*|X \cap Y \neq \emptyset\}$$

$\underline{P}Y$ is called P-lower approximation of Y and $\overline{P}Y$ is called P-upper approximation of Y. The P-lower approximation therefore contains only objects from set Y while P-upper approximation contains also objects from $U - Y$ which are "P-indiscernible" from some objects from Y. The relation between defined sets and set Y is:

$$\underline{P}Y \subseteq Y \subseteq \overline{P}Y \quad (3)$$

An approximation is sometimes represented also in terms of three regions:

- P-positive region of set Y in S :

$$POS_P(Y) = \underline{P}Y \quad (4)$$

- P-negative region of set Y in S :

$$NEG_P(Y) = U - \overline{P}Y \quad (5)$$

- P-boundary (doubtful) region of set Y in S :

$$BND_P(Y) = \overline{P}Y - \underline{P}Y \quad (6)$$

P-positive and P-negative regions contain P-equivalence classes containing objects that all belong and all do not belong to set Y, respectively. P-doubtful region is the union of P-equivalence classes each of which contains some objects that belong to Y and some objects that do not.

Approximation of family of sets is a generalization of approximation of set Y. If χ is defined as:

$$\chi = \{Y_1, Y_2, \ldots, Y_m\} \quad : \quad Y_i \subseteq U \quad (7)$$

then approximation of χ by P is defined:

$$\underline{P}\chi = \{\underline{P}Y_1, \underline{P}Y_2, \ldots, \underline{P}Y_m\}$$
$$\overline{P}\chi = \{\overline{P}Y_1, \overline{P}Y_2, \ldots, \overline{P}Y_m\} \quad (8)$$

If χ is a *classification* (i.e. $Y_i \cap Y_j = \emptyset \ : \ \forall i,j \leq m$, $i \neq j$, $\bigcup_{i=1}^{m} Y_i = U$) then measures of *roughness* of approximation of χ are defined as follows:

- *accuracy of approximation of* χ *by* P *in* S or shortly *accuracy of classification* χ

$$\beta_P(\chi) = \frac{\sum_{i=1}^{m} card(\underline{P}Y_i)}{\sum_{i=1}^{m} card(\overline{P}Y_i)} \qquad 0 \leq \beta_P(\chi) \leq 1 \quad (9)$$

- *quality of approximation of classification of* χ *by* P *in* S or shortly *quality of classification* χ

$$\gamma_P(\chi) = \frac{\sum_{i=1}^{n} card(\underline{P}Y_i)}{card(U)} \qquad 0 \leq \gamma_P(\chi) \leq 1 \quad (10)$$

2.2 Decision rules

If *classification* $\chi = \{Y_1, Y_2, \ldots, Y_m\}$ is defined in *information system* $S =< U, Q, V, f >$ and a set of attributes $P \subseteq Q$ induce P-equivalent classes $X = (X_1, X_2, \ldots, X_n)$, decision rules for set Y_j are defined (Pawlak, 1991):

$$OP_j =$$

$$\{r_{ij} : (Des(X_i) \Rightarrow Des(Y_j)); X_i \cap Y_j \neq \emptyset; i = 1..n\}$$

and decision rules for a whole system are then defined:

$$OP(\chi) = \{OP_j \ ; \ j = 1 \ldots m\} =$$
$$\{r_{ij} : (Des(X_i) \Rightarrow Des(Y_j)) \ ; \ X_i \cap Y_j \neq \emptyset \ ;$$
$$i = 1 \ldots n \ ; \ j = 1 \ldots m\} \quad (11)$$

where $Des(X_i)$ is a description of an equivalence class which is equivalent to the description of all objects from that class:

$$\forall x \in X_i : Des(x) =$$

$$(p_1 = v_{j_1}) \wedge (p_2 = v_{j_2}) \wedge \ldots \wedge (p_n = v_{j_n}) = Des(X_i) \quad (12)$$

where $x \in U$, $p_k \in P$, $v_{j_k} \in V_{p_k}$. $Des(Y_j)$ represents a description of the value of an *action* attribute (class).

Two types of rules are defined:

- r_{ij} is DETERMINISTIC $\iff$ $X_i \cap Y_j = X_i$
- r_{ij} is NONDETERMINISTIC $\iff$ $X_i \cap Y_j \neq X_i$

2.3 Learning algorithm

To generate a set of decision rules a subset P of attributes must be selected. Decision rules are generated using only attributes from P. For both these complex tasks the RST approach uses (ad-hoc) heuristics. There are some more definitions.

Let $S = (U, Q, V, f)$ be an information system.

- Set of attributes $R \subseteq Q$ *depends* on set of attributes $P \subseteq Q$ in S (denotation $P \rightarrow R$) $\iff \widetilde{P} \subseteq \widetilde{R}$.
- Set of attributes $P \subseteq Q$ is *independent* in S $\iff \forall P' : P' \subset P \Rightarrow \widetilde{P'} \supset \widetilde{P}$
- Set of attributes $P \subseteq Q$ is *dependent* in S $\iff \exists P' : P' \subset P \Rightarrow \widetilde{P'} = \widetilde{P} \quad (P' \rightarrow P)$
- Set of attributes $P \subseteq Q$ is *reduct* in S $\iff$ P is the greatest *independent* set in Q

The definition of a *reduct* is from (Pawlak et al., 1986). The "greatest independent set" is interpreted as locally greatest and not globally. In Slowinski (1992) a reduct is defined as the minimal set of attributes that define the same equivalence classes as original set of attributes (which is equivalent to the above definition). Here the word "minimal" is again interpreted as locally and not globally minimal.

From the above definitions we have the following properties:

1. If set of attributes $P \subseteq Q$ is *independent* in S $\Longrightarrow \forall p, q \in P : \neg(p \rightarrow q)\ \&\ \neg(q \rightarrow p)$
2. If set of attributes $P \subseteq Q$ is *independent* in S $\Longrightarrow \forall P' \subset P : card(P'^*) < card(P^*)$, $(\widetilde{P'} \supset \widetilde{P})$
3. Set of attributes $P \subseteq Q$ is *dependent* in S $\iff \exists P' \subset P$: P' is *independent* & $P' \rightarrow P - P'$

There may exist more than one reduct $P \subseteq Q$.

Learning algorithm as defined in RST, is divided into three steps:

1. Reduction of those attributes from set Q that do not change the Q-equivalence classes. For such attribute p holds $(Q - p) \rightarrow Q$. This leads to *independent* set of attributes A for which stands:

   $$Q^* = A^* \qquad \gamma_Q(\chi) = \gamma_A(\chi)$$

   As the number of reducts may be large one should need a preference criterion for searching for "good reduct". In the RST literature typically an ad-hoc search is performed, such as try to eliminate first attribute, then second etc. (see e.g. Pawlak et al., 1986; Slowinski & Slowinski, 1990; Slowinski, 1992a; Tanaka et al., 1992; Grzymala-Busse, 1992).

2. Elimination of attributes from reduct A (which causes joining some of equivalence classes and therefore decreases the quality of classification γ) until the lowest permitted predefined value of γ is reached. The search heuristic in each step eliminates the attribute that maximizes γ for remaining subset P of attributes.

3. Generation of rules for each class in turn using only attributes from P. Many authors from the RST community use the covering algorithm (e.g. Wong et al., 1986; Wong & Ziarko, 1986). In each iteration one rule is generated and correctly classified training instances are removed. Iterations terminate when all training instances are removed or when no more rules can be found.

   One rule is generated in a top-down manner, starting with empty condition, and by specializing the current rule in all possible ways and selecting the most promising specialization. This continues until the certain quality criterion is met. For deterministic RST, the quality criterion is the determinism of the conclusion part of the rule (Wong et al., 1986). For probabilistic RST, the quality criterion requires that the percentage of the majority class is above the user defined threshold $\alpha \geq 0.5$ (Wong & Ziarko, 1986).

Classification with decision rules is straightforward. If the description of an object is equal to the description of the condition attributes of the rule, the object is classified in the class represented with the value of the action attribute of

that rule. For nondeterministic rules the discrete RST doesn't give any priority to classes as it assumes equal distribution of classes Y_j in the equivalence class. That was improved in the probabilistic RST (Wong & Ziarko, 1986; Pawlak et al., 1988) where for the conclusion part the distribution of covered training instances was used.

Another problem is if the rule for classification does not exist (i.e. there is no rule with the description of condition attributes corresponding to the description of an object). In this case the algorithm searches for k closest rules that best match the object and their conclusion part is averaged (Krusinska et al., 1992).

3 Drawbacks of RST

3.1 Terminology

Probably the most confusing thing with RST is its complicated formalization of rather trivial (at least in the context of machine learning from examples) notions. While formalization is usually welcome and necessary to avoid confusion, in RST it adds confusion with numerous new notions and unusual terminology that often conflicts with the usual terminology of machine learning community.

The definition of boundary region BND_P, which is the central part of the RST, merely represents a part of the instance space where attributes do not suffice for discriminating between classes. The whole concept of a rough set merely states that certain classes cannot be discriminated from other classes. However, there is no information provided about the distribution of instances inside the boundary region.

The accuracy of classification β, defined with eq. (9), should not be confused with the usual meaning of classification accuracy. The quality of classification γ, defined with eq. (10), in fact represents the percentage of training objects that can be correctly classified using only attributes from P. This is equal to the classification accuracy on training data which may of course drastically differ from the classification accuracy on unseen objects. Therefore, γ is poor search heuristic and the same holds for the quality criterion based on the majority class (see step 2 and 3 of the learning algorithm in section 2.3).

The definition of *dependent* and *independent* set of attributes is unusual in the context of machine learning. It uses only logical dependency into acount which is not enough. (In)dependency is defined in the probability theory, however, RST doesn't operate with probabilities at all. Pawlak et al. (1988) try to overcome this (but only for probabilistic RST) by defining the measure of dependency in terms of entropy measure. However, they define that attributes (variables) X and Y are completely independent if

$$H(Y|X) = \log m$$

where m is the number of values of attribute Y. It is well known that $H(Y|X) \leq H(Y)$ and that X and Y are independent only when $H(Y|X) = H(Y)$, and $H(Y)$ is only in a very special (and rare) case equal to $\log m$.

3.2 Knowledge representation

The derived rules use a fixed subset of attributes and discard probably useful information contained in other attributes. The authors of RST claim that such attributes are redundant and unnecessary. This may be true for noise-free, complete data sets with exact classification, which obviously is not the case for the great majority of classification problems. For a certain problem subspace one subset of attributes may be relevant and for the other subspace another subset of attributes may be crucial.

On the other hand, there is no notion of the probability distribution and the reliability of conclusion parts of decision rules. Deterministic decision rules, which are in fact just a special case of nondeterministic rules, are supported from n training instances belonging to same class where $n \geq 1$. Obviously, for small values of n, the conclusion becomes unreliable and probability distribution should be estimated using Laplace's law of succession and m-estimate (Cestnik, 1990). Although the probabilistic RST is more flexible with this respect, almost all authors use the deterministic RST.

RST can deal with discrete attributes only and continuous attributes have to be discretized in advance. There is no obvious way how to deal with incomplete data (missing values) and noisy data. The only straightforward solution to the problem

of missing values is to define the unknown value as an additional value of an attribute, which is known to be unsatisfactory (Quinlan, 1989).

Some authors from the RST community use a *preset decision tree* for the knowledge representation (see e.g. Modrzejewski, 1993). A preset decision tree is a decision tree that has the same order of attributes for all paths from the root to the leaf. This has an obvious disadvantage to be less flexible than ordinary decision trees which are in turn less flexible than decision rules (Quinlan, 1987).

Kubat (1991) describes an algorithm for updating lower approximation $\underline{P}Y$ of concept Y and upper approximation $\overline{P}Y$ using the fixed set of attributes P. Both approximations are represented simply as sets of equivalence classes of relation $\widetilde{P}$. The equivalence classes contain objects with the same values for all attributes. Equivalence classes are disjoint and the union of all equivalence classes is equal to the set of all training instances. Theorems in the paper by Kubat are more or less trivial, although the awkward formal description is clumsy for reading and understanding. They simply describe how new objects can be added to these sets and how existing objects can be deleted. The only learning in this framework is therefore the memorization. There is no induction, no generalization and neither specialization. The knowledge is represented in the same way as it was provided to the learner, except that objects are grouped into the equivalence classes. Equivalence classes containing objects from more than one class are in the boundary region $BND(Y)$. Obviously, such framework is not very useful and, besides, it is drastically sensitive to noise and missing data.

3.3 Ad-hoc solutions

Instead of using well known results from the probability theory and the information theory, the authors from the RST community often use ad-hoc definitions and solutions. There is plenty of parameters and thresholds with poor theoretical background. While dealing with definitions of straightforward notions, the RST is strictly formal and rigorous. As soon as more interesting problems are encountered, such as finding a good reduct or searching for good rules, ad-hoc heuristics are used.

Such ad-hoc heuristics are described in section 2.3 (learning algorithm) and discussed in section 3.1 above. Note, that the definition of a *reduct* is ad-hoc, at least in the context of machine learning, as it is not connected to the class attribute at all. Having enough random attributes, a huge number of reducts can be found that does not reflect any domain regularities at all. Therefore, any definition of a "good" reduct is also ad-hoc and any search heuristic for finding a reduct is necessarily ad-hoc.

3.4 Comparisons of RST to other approaches

Although there are many applications of RST (see e.g. Slowinski, 1992), there was practically no comparison of performance with existing machine learning algorithms. Babič et al. (1992) compared the performance of Assistant (Cestnik et al., 1987) with CART (Breiman et al., 1984) on two medical data sets. They reported 74% and 84% of classification accuracy, for each data set in turn, achieved by Assistant. The same authors (Krusinska et al., 1992) tested the performance of the RST learning algorithm on the same data sets and reported 73% and 80% of classification accuracy for each data set in turn They also report about results, obtained by Assistant Professional package, but only when using unusual classification methods in combination with the naive Bayesian classifier which does not use m-estimate of probabilities (Cestnik et al, 1987); and these results were worse than results reported in (Babič et al., 1992).

Wong et al. (1986) theoretically analysed RST and ID3 (Quinlan, 1979) and concluded:

"The criterion for selecting dominant attributes based on the concept of rough sets is a special case of the statistical method if equally probable distribution of objects in the doubtful region of the approximation space is assumed."

The assumption of equally probable (uniform) distribution is far from realistic.

Teghem & Benjelloun (1992) performed similar analysis and concluded in the self contradictory statements:

"If RST certainly is an efficient tool to analyse

information systems, often more simple and comprehensive than Quinlan's method using entropy notion, nevertheless the comparison suggests that some improvements can still be effected in the rough sets approach. Further researchs are necessary to investigate how the distribution of the objects into doubtful regions can be taken into account."

4 Experimental comparison

We reimplemented the RST learning algorithm and tested its performance on several real world data sets. We compared the performance with "classic" machine learning algorithms: Assistant inductive learning algorithm for generating decision trees (Cestnik et al., 1987) and the naive Bayesian classifier with m-estimate of probabilities (Cestnik, 1990).

The important characteristics of our implementation of RST learning algorithm are:

- We used the RST learning algorithm, described by Wong et al. (1986) generalized to probabilistic RST (Wong & Ziarko, 1986; Pawlak et al., 1988). We tried different values of majority class limit α. Here we present the best results obtained for each data set. This is somehow an overestimation of the performance of the RST learning algorithm.

- In the case of searching for k "closest rules" to the testing object, k was set to 1 and the distance between rules and instances was defined as the number of condition attributes that have different value.

- Unknown value was treated as an additional value of the attribute. Namely, RST does not provide any methodology to deal with this problem.

The description of data sets used in our experiments is provided in table 1. Besides the usual numeric description of the data (number of attributes, number of classes, and number of cases) we provide also the class entopy and the proportion of the cases from the majority class. The class entropy shows how simple/hard is the classification problem while the proportion of the majority class shows the "default accuracy", i.e. the accuracy that can be achieved with a simple classifier that classifies all instances in the majority class.

One experiment (trial) consisted of dividing the set of objects into 70% for learning and 30% for testing. We performed 10 experiments, each with different split, and results were averaged. The measured parameters were:

- classification accuracy (the percentage of correctly classified testing instances), results are presented in table 2;

- average information score, a measure that eliminates influence of prior probabilities and is defined as follows (Kononenko & Bratko, 1991):

$$Inf = \frac{\sum_{i=1}^{\#testing\ instances} Inf_i}{\#testing\ instances} \qquad (13)$$

where the information score of classification of i-th testing instance is defined with the following. Let Cl_i be the class of i-th testing instance, $P(Cl)$ the prior probability of class Cl and $P'(Cl)$ the probability returned by a classifier. We define the information score for two cases:

The information score is positive if the probability of the correct class given by the classifier is greater than the prior probability of that class. The information gain is equal to the prior information minus the posterior information necessary to correctly classify that instance:

$P'(Cl_i) \geq P(Cl_i)$:

$$Inf_i = -log_2 P(Cl_i) + log_2 P'(Cl_i)$$

If the classifier decreases the prior probability of the correct class, the provided information is wrong and therefore negative. It is equal to the prior information minus the posterior information necessary to incorrectly classify that instance:

$P'(Cl_i) < P(Cl_i)$:

$$Inf_i =$$

$$-(-log_2(1 - P(Cl_i)) + log_2(1 - P'(Cl_i)))$$

Results are presented in table 3.

Table 1: Characteristic description of experimental data sets.

domain name	# attributes	# classes	# cases	class entropy	majority class
primary tumor	17	22	339	3.64bits	25%
breast cancer	10	2	288	0.72bits	80%
thyroid diseases	15	4	884	1.59bits	56%
rheumatology	32	6	355	1.70bits	66%
hepatitis	20	2	155	0.73bits	79%
lymphography	19	4	148	1.23bits	55%
criminology	11	4	723	1.34bits	64%
fresh concrete	14	4	254	1.77bits	42%

Table 2: Comparison of the classification accuracy (%) of different classifiers on various data sets.

domain name	Assistant	naive Bayes	RST
primary tumor	44	50	35
breast cancer	77	79	80
thyroid diseases	73	72	61
rheumatology	65	69	66
hepatitis	82	87	81
lymphography	79	84	77
criminology	61	61	63
fresh concrete	61	63	61

Table 3: Comparison of the average information score (bit) of different classifiers on various data sets.

domain name	Assistant	naive Bayes	RST
primary tumor	1.38	1.57	0.96
breast cancer	0.07	0.18	-0.04
thyroid diseases	0.87	0.85	0.46
rheumatology	0.46	0.58	0.16
hepatitis	0.15	0.42	0.12
lymphography	0.67	0.83	0.51
criminology	0.06	0.27	0.03
fresh concrete	0.70	0.89	0.59

All differences in the classification accuracy (table 2) that are less than 4 % are statistically insignificant (confidence level is 0.99 using two-tailed t-test). Other differences are significant. However, note that for breast cancer, rheumatology, and criminology, where the differences are the lowest, the classification accuracy is practically equal to the proportion of the majority class. For those data sets the information score is a better measure. The majority of differences in information score (table 3) are statistically significant (the exceptions are the differences between Assistant and RST in hepatitis and criminology).

Results of RST are poor when compared to Assistant and the naive Bayesian classifier with respect to classification accuracy and/or information score. Therefore, RST did not achieve the performance of "classic" machine learning algorithms. Besides, nowadays there exist better machine learning algorithms which usually obtain better performance with multistrategy learning (e.g. Quinlan, 1993; Brodley, 1993). In fact we used a multistrategy approach (in the sense of multiple sets of rules) in our implementation of RST to improve the performance of RST. However, the underlying assumptions prevented the RST learning algorithm to perform well.

5 Conclusion

Lists of references in the papers on RST contain plenty of self referencing while none or modest number of references from machine learning literature. There are some exceptions (e.g. Grzymala-Busse, 1992), however the only purpose of referencing in such cases is to inform about alternative approaches without any explicit comparison. It seems that many authors have no overview of the work that is going on in machine learning and that may be the reason for many reinventings and also plenty of ad-hoc solutions.

Complicated formalization in RST adds confusion with numerous new notions and unusual terminology that prevents global overview of the RST and prevents systematic analysis. This may be why so many authors use RST without analysing its basic assumptions, which are in most cases unrealistic. The problems with noise and incomplete data disable RST from providing efficient solutions for complex real-world problems.

Acknowledgements

We are grateful to Ankica Babič for providing the literature on RST and Matevž Kovačič and Uroš Pompe for their comments on the manuscript.

References

[1] Babič. A., Krusinska E., Stromberg J.E. (1992) Extraction of diagnostic rules using recursive partitioning systems: A comparison of two approaches. *Artificial Intelligence in Medicine*, 4: 373-387.

[2] Breiman L., Friedman J.H., Olshen R.A., Stone C.J. (1984) *Classification and Regression Trees*, Wadsforth International Group.

[3] Brodley C.E. (1993) Addressing the selective superiority problem: automatic algorithm/model class selection. *Proc. 10th Int. Conf. on Machine Learning*, (Amherst, MA, June 1993), Morgan Kaufmann, pp. 17-24.

[4] Cestnik B. (1990) Estimating probabilities: A crucial task in machine learning, *Proc. European Conf. on Artificial Intelligence*, Stockholm, August, 1990, pp. 147-149.

[5] Cestnik B., Kononenko I.& Bratko I. (1987) ASSISTANT 86 : A knowledge elicitation tool for sophisticated users, in: I.Bratko, N.Lavrač (eds.): *Progress in Machine learning*, Wilmslow: Sigma Press.

[6] Grzymala-Busse J.W. (1992) LERS - A system for learning from examples based on rough sets. In. Slowinski R. (ed.) *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publ.

[7] Kononenko I. & Bratko I. (1991) Information based evaluation criterion for classifier's performance, *Machine Learning*, 6: 67-80.

[8] Krusinska E., Babič A., Slowinski R., Stefanowski J. (1992) Comparison of the rough sets approach and probabilistic data analysis techniques on a common set of medical data. In. Slowinski R. (ed.) *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publ.

[9] Kubat M. (1991) Conceptual inductive learning: the case of unreliable teachers. *Artificial Intelligence*, 52: 169-182.

[10] Modrzejewski M. (1993) Feature selection using rough sets theory. *Proc. European Conf. on Machine Learning*, Ed. P. Brazdil, Springer Verlag, pp.213-226.

[11] Pawlak Z. (1982) Rough sets. *Int. J. of Computers and Information Sciences*, Vol. 11, pp. 341-356.

[12] Pawlak Z. (1984) On superfluous attributes in knowledge representation. Bulletin of the Pollish Academy of Sciences, 32.

[13] Pawlak Z. (1991) *Rough Sets: Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publ.

[14] Pawlak Z., Slowinski K., Slowinski R. (1986) Rough classification of patients after highly selective vagotomy for duodenal ulcer. *Int. J. Man-Machine Studies*, 24: 413-433.

[15] Pawlak Z., Wong S.K.M., Ziarko W. (1988) Rough sets: probabilistic versus deterministic approach. *Int. J. Man-Machine Studies*, 29: 81-95.

[16] Quinlan, J.R. (1979) Discovering Rules by Induction from Large Collections of Examples. In: D.Michie (ed.), *Expert Systems in the Microelectronic Age.* Edinburgh University Press.

[17] Quinlan, J.R. (1986) Induction of Decision Trees. *Machine Learning.* 1: 81-106.

[18] Quinlan J.R. (1987) Generating production rules from decision trees. *Proc. IJCAI-89*, Milan, August 1987, pp.304-307.

[19] Quinlan J.R. (1989) Unknown attribute values in induction, *Proc. 6th Int. Workshop on Machine Learning*, Cornell University, Ithaca, June 26-27, 1989, pp.164-168.

[20] Quinlan J.R. (1993) Combining instance-based and model-based learning. *Proc. 10th Int. Conf. on Machine Learning*, (Amherst, MA, June 1993), Morgan Kaufmann, pp. 236-243.

[21] Slowinski K. (1992a) Rough classification of HSV patients. In. Slowinski R. (ed.) *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publ.

[22] Slowinski K. & Slowinski R. (1990) Sensitivity analysis of rough classification. *Int. J. Man-Machine Studies*, 32: 693-705.

[23] Slowinski R. (ed.) (1992) *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publ.

[24] Tanaka H., Ishibuchi H. & Shigenaga T. (1992) Fuzzy inference system based on rough sets and its application to medical diagnosis. In. Slowinski R. (ed.) *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publ.

[25] Teghem J. & Benjelloun M. (1992) Some experiments to compare rough sets theory and ordinal statistical methods. In. Slowinski R. (ed.) *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic Publ.

[26] Wong S.K.M. & Ziarko W. (1986) INFER - An adaptive decision support system based on the probabilistic approximate classification.*Proc. 6th Int. Workshop on Expert Systems and their Applications*, Avignon, France, pp. 713-726.

[27] Wong S.K.M., Ziarko W., Li Ye R. (1986) Comparison of rough-set and statistical methods in inductive learning, *Int. J. Man-Machine Studies*, 24: 53-72.

ON THE EXPLOITATION OF MECHANICAL ADVANTAGE NEAR ROBOT SINGULARITIES

Jon Kieffer
Engineering Department, The Faculties, Australian National University
Canberra ACT 2601 Australia

Jadran Lenarčič
Jozef Stefan Institute, University of Ljubljana
Jamova 39, 61111, Ljubljana, Slovenia
E-mail: jadran.lenarcic@ijs.si

Keywords: robot, singular

Edited by: Rudi Murn
Received: August 28, 1993 **Revised:** April 18, 1994 **Accepted:** June 3, 1993

Since the earliest days of robotics research when it was recognized that kinematic singularities physically hamper the free manipulation of objects in task space, there has been a popular consensus that singular configurations are unsuitable for practical use and should be avoided. At best they may be included with the expectation of gracefully-degraded performance in their vicinity [1,2]. In this article we question the validity of such conclusions, citing examples that show how humans use singularity configurations of their limbs to gain mechanical advantage, and investigate the possibility of obtaining similar benefits in robotic systems. It is shown that minimization of joint torques in redundant systems leads to human-like behavior that favors singularities, but that stable implementation of such behavior requires a strategy which gives the robot more autonomy with respect to timing task execution. Application of such a strategy to a 2R robot performing static lifting is considered in detail.

1 Introduction

A fundamental objective of robot design and control is to mask the characteristics of the underlying machine to provide an abstraction that is easily understood and programmed. To ease task specification we would like a manipulator arm (walking machine leg) to behave as a disembodied hand (foot) whose motion, force, or impedance [3] can be arbitrarily programmed to fit a task. For convenience it is sensible to specify tasks in a cartesian space that we and our sensors can readily identify with. To this end, robot mechanisms, actuators, and control systems have been designed and implemented with varying degrees of success.

Unfortunately, all real robots have limitations in speed, workspace, and force capability that compromise the ideal abstraction of an arbitrarily-programmable disembodied hand (foot). To further complicate matters, these limitations are interrelated in a complex way for all robots, except gantry robots. To a large extent these problems can be ameliorated through design and control. Mechanical design methodologies have been developed to optimize workspace size and shape [4], minimize the directional distortion of motion and force capability [5] and even to move kinematic singularities, which severely distort directional force and motion characteristics, out of the intended workspace [6]. In addition, researchers have turned to redundant manipulators whose extra degrees of freedom offer new possibilities for avoiding singularities and improving the directional uniformity of motion and force characteristics, now called dexterity [7], through control. Nevertheless, by their vary nature, articulated robots distort cartesian force and motion

characteristics and include configurations of ultimate distortion: kinematic singularities.

Our objective in this paper is to demonstrate that kinematic singularities and ill-conditioned configurations offer untapped potentials that may be exploited through an alternative model (abstraction) for robot behavior that is better suited to taking advantage of these potentials. In doing so we hope to provoke increased interest in kinematic singularities as well as in alternative strategies for programming useful robot behavior.

2 Background

Kinematic singularities can be conceptualized as configurations of local folding in the mapping of a toroidal joint space onto an dissimilar manifold of spatial end-effector positions/orientations [8]. For the case of a 2R planar robot, two joint positions (elbow-left and elbow-right) map onto the same end-effector position and folding of the joint space manifold with respect to the mapping accounts for the transition from inside the workspace, where two joint solutions generally exist, to outside the workspace, where there are no solutions. For the special case of equal link length geometry, the inside workspace boundary (normally a circle) collapses to a point which is the image of a one parameter locus of singular configurations in joint space. Physically, the end point will be in the workspace center whenever the two links fold over each other.

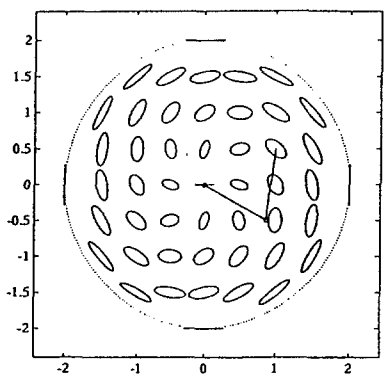


Figure 1: Velocity ellipses for 2R robot with equal link lengths.

Although singularity configurations are sparse compared to regular configurations, their influence extends to significant portions of the surrounding workspace, distorting the directional properties of nearby regular configurations in proportion to their proximity. As illustrated in Fig.1, uniform joint velocity capability, represented by circles in the joint space, become ellipses in the task space and collapse in at least one direction near singularity configurations. This is true of both force and motion characteristics and extends to dynamic properties, such as effective inertia, as well [9, 10]. It is an exaggeration to say that singularities cause this behavior, they are only extreme manifestations of it.

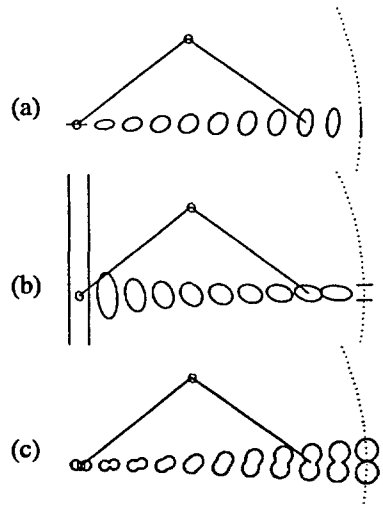


Figure 2: (a) Velocity ellipses, (b) force ellipses, and (c) torque "peanuts" along radial workspace section (2R robot with equal link lengths).

A more complete understanding can be obtained by considering the relation between the force and motion distortions. By comparing force and velocity ellipses along a radial section of the 2R manipulator's workspace (Fig.2(a-b)) we see that the distortions are complementary. At the workspace boundary, joint actuators can support infinite radial loads, but radial velocities become impossible. Conversely, tangential velocities at the workspace boundary are enhanced and tangential force capability is diminished. This complementary property persists, with diminished distortion, at regular configurations and extends to general spatial manipulators as well. An alternate way to represent directional force characteristics is to plot the norm of joint torques needed to support a unit tip force in each direction. As shown in Fig.2(c) this results in "peanut"-shaped figures that, in contrast to force ellipses, remain

bounded at singularities.

It can be shown that force and velocity ellipses share the same principle directions, but have reciprocal principle axes lengths [11]. For this reason it is convenient to consider only velocity ellipses, henceforth called manipulability ellipses, with the understanding that both force and velocity characteristics can be inferred from them.

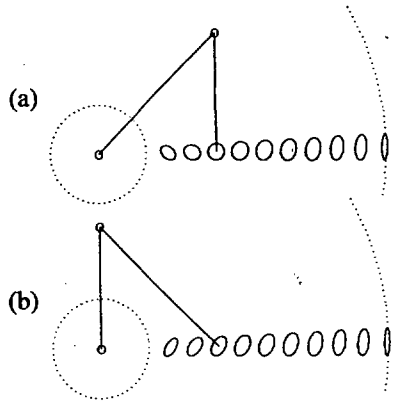


Figure 3: Manipulability ellipses: (a) the only 2R robot with isotropic configurations $L_1 = L_2\sqrt{2}$, (b) 2R robot without isotropic configurations $L_2 = L_1\sqrt{2}$.

Truly uniform properties only occur at so-called isotropic configurations [12], which only exist for special robot geometries. Figure 3 compares the manipulability ellipses for two robots with identical workspaces. The first robot is the only 2R geometry that has isotropic configurations.

3 Potentials for Exploitation

The idea that singularities may be useful was first suggested by K.H. Hunt [13] who recognized that, at singularity configurations, the space of instantaneous joint screws fails to span the entire screw space. This enables any reciprocal wrench applied to the end effector to be transmitted through the structure without loading any actuators. This means that there is a potential for applying or withstanding extremely high loads in certain directions at singularities. Although this argument has been often repeated, and suggested as useful for practical application, e.g. drilling [14], to the authors' knowledge it has yet to be applied in practice.

Nevertheless, it is easy to verify that humans take advantage of this principle when walking or standing, for example. To see this consider the 2R planar manipulator as a approximation of the human leg. When outstretched to a singularity configuration, radial loads can be transmitted through the structure without loading the joint actuators. This is how humans avoid muscle fatigue while supporting their weight when walking or standing.

Although effective for walking, this example also points out the specialization of motion required to take full advantage of singularities which are local and directionally-oriented within the workspace. In walking, the endpoint (foot) only moves tangentially around the workspace boundary where singularities support the radially-directed load. If the 2R manipulator were applied to drilling for example, it would be necessary to locate the end-effector and workpiece near either the center or the boundary of the workspace and to align them appropriately with respect to the direction of heavy loading. Furthermore, once this is done, we should expect to apply only small motions in the direction of heavy loading (drilling) because large ones would move robot too far from the region of maximum mechanical advantage. Obviously, this approach would be awkward to apply in practice.

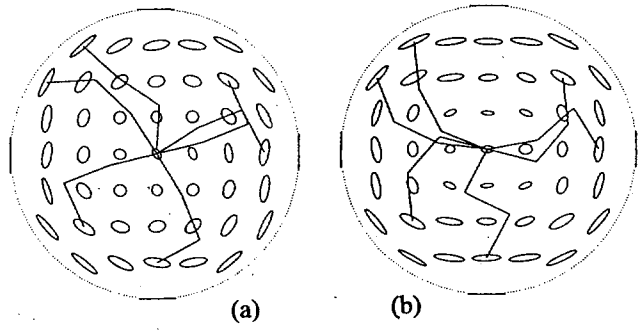


Figure 4: Manipulability ellipses for a 3R robot: (a) configurations optimized for maximum dexterity (minimum condition number), (b) configurations optimized to minimize the norm of joint torques for a vertical tip load.

Ideally, we would like to achieve maximum mechanical advantage in any chosen direction at any end-effector position in the workspace (arbitrary placement and alignment of singularities). However, because this seems impossible, we might settle for extending the influence of singularities to larger portions of the workspace. In principle,

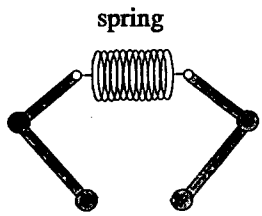


Figure 5: Archer modeled by planar dual arms with a constant force spring.

this can be achieved through redundancy: analogous to the way redundancy can been exploited to increase dexterity (uniformity in properties) throughout the workspace [9], it can also be exploited to propagate nonuniformity of properties (i.e., ill-conditioned configurations and singularities) throughout the workspace.This principle is confirmed in Fig. 4 which compares manipulability ellipses for a 3R planar robots whose configuration at each workspace position has been optimized to (a) maximize uniformity (minimize condition number), and (b) minimize the joint torques needed to balance a vertical tip load.

4 Human Exploitation of Singular Configurations

Further insight into potentials for robotic applications can be gained by considering how humans use singularities.

In performing the so-called "clean and jerk", weight lifters take advantage of the singularity configurations of their arms. From the floor, the weight is thrown upward using forces generated by leg muscles and transmitted through their arms which are stretched downward in a singular configuration. The weight is then caught with arms folded in a second singularity configuration that is similar to the center point singularity of the 2R manipulator. After a pause, the weight is again thrown upward using leg muscles and caught with arms stretched overhead in a third singularity configuration. Obviously, the weight is too heavy for the arms to manipulate except near these singularities and the lifting process is one of ballistically transferring the weight between them.

These examples show that in walking and weightlifting humans use singularity configurations mainly to support heavy loads, rather than to apply forces. In this sense, singularities transform the mechanism (limbs) into a structure with respect to certain directions of applied loads.

In drawing a bow, archers also make use of ill-conditioned configurations and singularities to minimize muscle effort in their arms. As an explanation, consider the planar dual arm system shown in Fig.5 as approximation of an archer. The bow has been modeled by a constant force spring and the system has three degrees of redundancy with respect to extending the spring. Fig 6(a) shows the pseudoinverse solution which minimizes joint motion. Figure 6(b) shows an alternate solution which minimizes joint torques (section 7) and is obviously similar to human behavior. Figure 7(a) shows plots that compare joint torques versus spring length for the two solutions, confirming that the second solution substantially reduces effort.

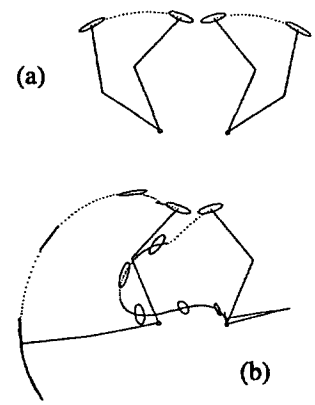


Figure 6: Archer solutions: (a) minimal joint motion (b) minimal joint torques.

The manipulability ellipsoids in Fig.6(b) show that singularity configurations play an important role in minimizing effort. Because the initial configuration is not well suited to the spring load, the flattened ellipses are first reoriented by self-motion. The resulting increase in mechanical advantage provides a dramatic decrease in torque (Fig.7(a)). The spring is then extended with the left arm outstretched and the right arm drawing backward toward its center point. The final posture can be sustained without effort due to singularity configurations of both arms. Although this model does not consider joint limits or in-

terference with the body, it is clear that human archers use essentially the same strategy, taking advantage of singularity configurations to minimize effort and to reduce fatigue while aiming.

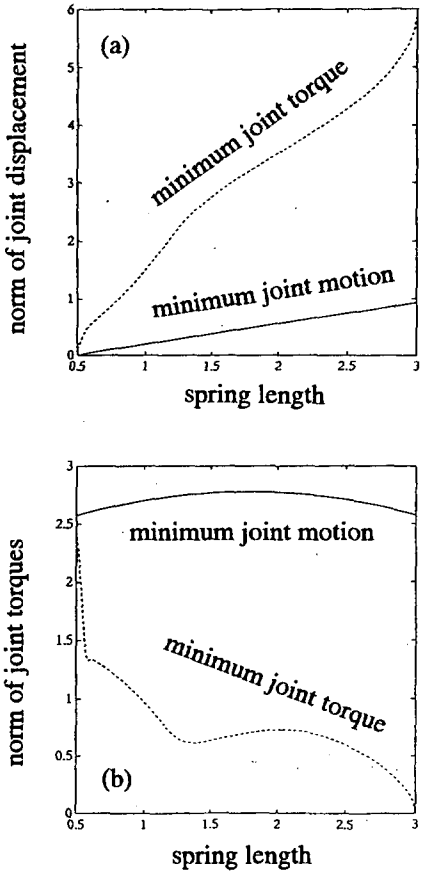


Figure 7: Comparison of archer solutions: (a) norm of joint torques, (b) joint displacement versus spring length for archer solutions.

In addition to confirming the importance of redundancy, this example shows that ill-conditioned configurations near singularities are well suited to applying forces.

5 The Need for Temporal Autonomy

As stressed in the introduction, kinematic singularities are usually regarded as nuisances that should be avoided because they confound the planning and execution of timed-end effector trajectories.

Exactly at a singularity, the Jacobian matrix relating joint velocities to the cartesian end-effector velocity loses rank and the usual joint rate solution becomes indeterminate. Although this mathematical problem can be overcome by an untimed parametric formulation [15] which considers higher-order kinematics if necessary [16], there still remains the underlying physical problem of joint rates becoming unbounded for certain directions of finite end-effector velocity. More importantly, joint rates also become unreasonably large for substantial regions of ill-conditioned regular configurations close to singularities.

Physically, the situation can be understood from the flattening of the manipulability ellipses near singularities: small end-effector velocities (displacements) in the flattened direction require large joint rates (displacements). Although, this is generally viewed as a problem, it is actually the definition of mechanical advantage. Large end-effector velocities in the flattened direction are simply not possible, but large forces are. From this perspective, redundancy can be interpreted as a mechanism for changing transmission ratio: ill-conditioned configurations provide low transmission ratios with respect to motions in flattened directions and high transmission ratios with respect to motions in the lengthened directions.

The main obstacle to making use of ill-conditioned configurations is the complexity of velocity limitations which make it very difficult to plan the timing of end-effector motions. This is illustrated in Fig.7(b) which plots joint displacement versus spring length for the two archer solutions of Fig 6. The minimum torque solution is far less uniform and requires far more joint motion, especially near singularities. This makes it almost impossible to prescribe the rate of spring extension without exceeding joint rate limits.

This means that the ideal abstraction of a disembodied hand that can be arbitrarily programmed within simple velocity bounds must be compromised and replaced with a more sophisticated view: one that includes more consideration of the machine that moves the hand. But, rather than burden the task planner with complicated details, it is better to simply relieve the planner of timing considerations altogether: let the planner specify the geometry of task execution, but let the robot control system determine timing in accordance with the robot's capability. This provides a simple abstraction which is similar to human supervision: tasks are assigned, but precise timing

of their execution is not dictated. For tasks such as welding, which require more-or-less strict velocity control along a path, this strategy may not be appropriate. But there are many other tasks in which timing is of secondary importance and can be sacrificed in favor of increased mechanical advantage.

6 Temporally-Autonomous Path Control

Hollerbach and Suh [17], resolved redundancy to minimize joint torques subject to robot dynamics. In doing so, they found the robot was likely to blunder into a region near a singularity, where joint rates become unacceptably large. In this section we will consider a similar, but simpler, problem: resolving redundancy to minimize joint torques in a robot subject to a static end effector forces, ignoring dynamics.

Various investigators [18-20] have proposed the use of generalized inverses of the form

$$\dot{\theta} = J^{+}\dot{r} - (I - J^{+}J)\nabla H \qquad (1)$$

which minimizes a potential function $H(\theta)$ subject to the kinematic constraint $J\dot{\theta}(t) = \dot{r}(t)$. Here $\dot{\theta}(t)$ represents the joint rates, $r(t)$ represents a timed end-effector trajectory, $J(\theta)$ is the Jacobian matrix, $J^{+} = J^{T}(JJ^{T})^{-1}$ is the Moore-Penrose generalized inverse, and ∇H is the gradient of $H(\theta)$.

Taking $H(\theta) = \frac{1}{2}\tau^{T}\tau$, where $\tau(\theta) = J^{T}f$ represents the configuration-dependent joint torques resulting from the application of a static tip force f, we obtain an inverse rate solution which minimizes the norm of joint torques, subject to execution of the timed-trajectory $r(t)$. Unfortunately this solution fails. As the robot moves toward singularities in an effort to minimize joint torques, larger and larger joint rates are needed to execute the timed-trajectory, $r(t)$. Eventually, joint rate limits are exceeded and control is lost. However, this is not the fault of singularities, which in fact provide the mechanism for minimizing joint torques. Rather, as discussed in the previous section, it is the fault of an inappropriate timing specification which does not consider joint rate limitations.

As an alternative strategy, consider that an untimed trajectory specification $r(\lambda)$ is given, and let timing be determined by the controller. The generalized inverse can then be expressed

$$\dot{\theta} = J^{+}r'\dot{\lambda} - \dot{\mu}(I - J^{+}J)\nabla H \qquad (2)$$

where $\dot{\lambda}$ and $\dot{\mu}$ are positive scalars determined online by the controller. The first term provides trajectory execution while the second provides self-motion that minimizes joint torques. In determining $\dot{\lambda}$ and $\dot{\mu}$ at any instant, the controller can govern the relative priority of joint torque minimization versus trajectory execution while simultaneously ensuring that joint rates remain within physically-realizable bounds.

A detailed exploration of such a strategy is beyond the scope of this paper and certainly includes significant obstacles, including numerical problems in the close proximity of singularities where the matrix JJ^{T} (needed to compute $J^{+} = J^{T}(JJ^{T})^{-1}$) looses rank. However, further insight can be gained by examining a crude example of this strategy applied to a 2R robot lifting problem.

7 Static Lifting with a 2R Robot

Figure 8(a) depicts a 2R robot lifting a weight. Ignoring the x-coordinate of the tip, the system has one degree of redundancy with respect to lifting in the y-direction. The problem is to lift the weight while minimizing joint torques. The links are considered massless and the weight static.

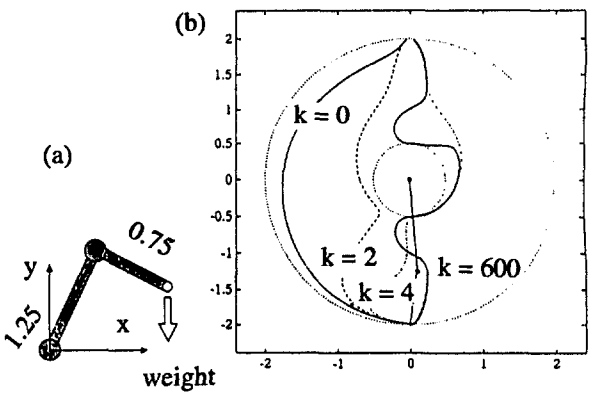


Figure 8: (a) 2R weightlifting robot, (b) tip trajectories for k=0,2,4, and 600.

One approach to generating an optimal trajectory is to perform a sequence of constant-y slice optimizations beginning a $y = -2$ and proceeding upward. However, to get more insight into the obstacles to control, we will consider a control-oriented approach based on the previous section.

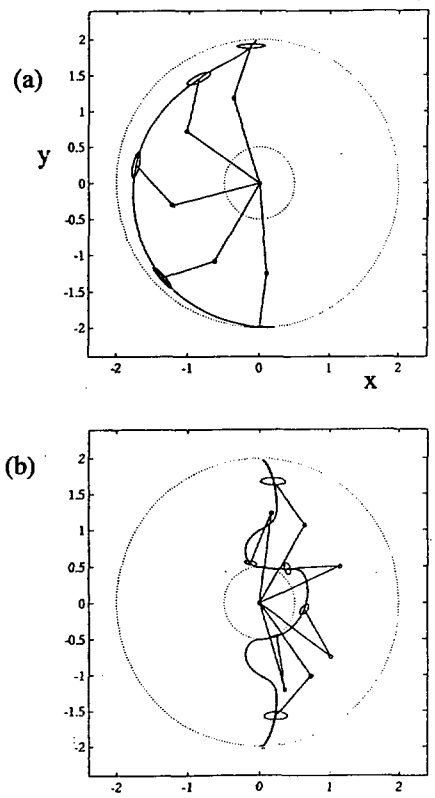


Figure 9: Weightlifting robot motions: (a) k=0: minimal joint motions, (b) k=600: minimal torques.

In particular, trajectories will be generated based on integrating equation (2) with incremental steps of constant length δ in joint space. Let each integration step be defined as follows.

$$\Delta\theta = \delta \frac{\nu}{|\nu|} \tag{3}$$

where

$$\nu = J^{+}r' - k(I - J^{+}J)\nabla H \tag{4}$$

Here k provides weighting with respect to torque minimization versus trajectory execution. Since the operator projects the torque gradient ∇H onto the null space of J, there is no chance of backward motion along the trajectory, $r(\lambda)$=$y(\lambda)$=λ. If $k(I - J^{+}J)\nabla H$ is large with respect to $J^{+}r'$, then most to the joint motion results in robot self-motion to reduce torques. Conversely, if $J^{+}r'$ is large relative to $k(I - J^{+}J)\nabla H$, then most of the joint motion is directed toward trajectory execution.

Figure 8(b) plots tip trajectories which result for four values of k. Weighting factors k=0 and k=600 represent extremes that minimize joint motion, and joint torques, respectively. As an aid to visualization, Figs. 9(a) and 9(b) show the robot and manipulability ellipses for intermediate positions of the robot executing the k=0 and k=600 trajectories respectively.

In minimizing joint torques, trajectories with large k take increased advantage of singularities near the inner circular workspace boundary. Trajectory k=4 comes very close to a singularity on the bottom of the inner workspace boundary while k=600 finds two singularities (bottom and top of inner workspace boundary). The increase in mechanical advantage which the singularities provide is apparent from Fig.10 which plots the norm of joint torques versus lifting height y.

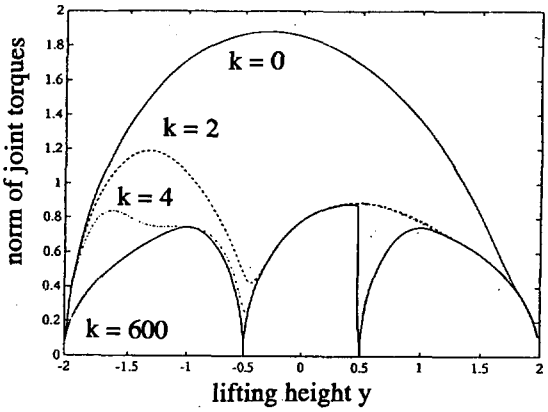


Figure 10: Norm of joint torques versus lifting height for weightlifting robot.

Some insight into the trajectory optimization can be gained from Figs. 11(a) and 11(b) which plot the surface of the joint torque norm over the workspace for elbow-left and elbow-right configuration branches, respectively. However, care must be taken, since the configuration branch can change when a singularity is encountered as is the case for the k=600 trajectory shown.

Perhaps better insight can be gained by considering the trajectories in joint space. Figures 12(a) and 12(b) show the trajectories superimposed on

contour plots of the torque norm, $|\tau(t)|$, and lifting height, $y(\theta)$, respectively. With increasing k, the algorithm takes increased advantage of singularities to minimize torques while simultaneously increasing y to perform the lifting task. It is also apparent that increasing k increases the joint-space path length, minimizing the change in y with respect to incremental joint motion, especially near singularities. This implies that timing of the trajectory in task space (i.e, $y(t)$) becomes more difficult or impossible with increasing k.

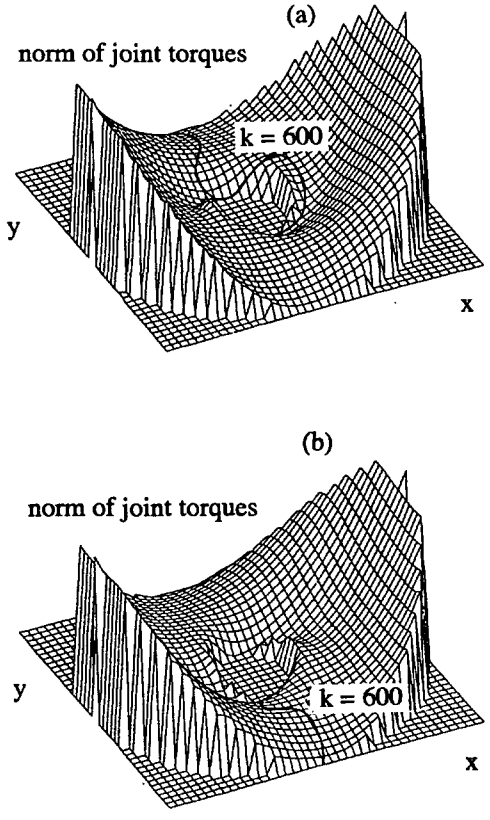


Figure 11: Surface plots of joint torque norm versus tip position for weightlifting robot: (a) elbow-left, (b) elbow-right.

In joint space, $\theta(t)$ can be prescribed relatively freely, except exactly at singularities where discontinuities imply that the robot must decelerate to a stop before accelerating onwards along the path. This means that timing, even in joint space, requires careful consideration if one is to fully exploit singularities. However, from the k=4 trajectory it is also clear trajectories only have to come close to singularities to gain the majority of torque reduction (Fig 10).

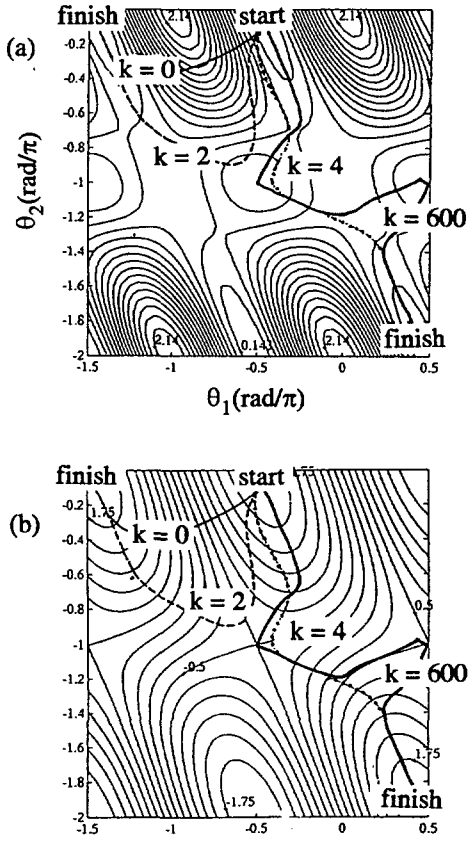


Figure 12: Joint trajectories over contour plots of (a) joint torque norm, and (b) lifting height y.

8 Conclusion

The assertion that kinematic singularities of articulated mechanisms offer untapped potentials seems indisputable from at least two points of view: (1) conventional insistence on uniform task-space velocity capability obviously limits the usable workspace and fails to take full advantage of configurations that offer extremes in motion or force capability which, by their nature, are directionally-biased, and (2) it is easy to cite examples of humans using singularity configurations to gain mechanical advantage.

In this paper, we have identified timing as the major obstacle to exploiting singularities in redundant robots and have proposed an alternative control strategy in which timing is determined online in accordance with machine limitations. Simple examples show that significant reductions in joint torques can be obtained, suggesting that

weak actuation can be compensated with more sophisticated control that exploits singularities and ill-conditioned configurations.

References

[1] Wampler, C.W. II: Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-16(1):93-101, 1986.

[2] Nakamura, Y. and Hanafusa, H.: Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control. *ASME J. Dynamic Systems, Measurement, and Control*, 109:163-171, 1986.

[3] Hogan, N: Impedance Control: An Approach to Manipulation, Parts I- III. *ASME J. Dynamic Systems, Measurement, and Control*, 107(1):1-24, 1985.

[4] Lenarčič, J., Stanič, P., and Oblak, P.: Some Considerations for the Design of Robot Manipulators. *Robotics and CIM*, 5(2/3): 235-241, 1989.

[5] Asada, H., and Youcef-Toumi, K.: Development of a Direct-Drive Arm Using High Torque Brushless Motors. In *Robotics Research - The First International Symposium*, Brady and Paul, Eds., MIT Press, pp. 583-599, 1984.

[6] Stanišić, M.M., and Duta, O.: Symmetrically Actuated Double Pointing Systems: The Basis of Singularity-Free Robot Wrists. *IEEE Trans. Robotics and Automation*, 6(5):562-569, 1990.

[7] Klein,C.A. and Blaho, B.E.: Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators. *Int. J. Robotics Research*, 6(2):72-83, 1987.

[8] Gottlieb, D.H.: Robots and Topology. Proc. *IEEE Int. Conf. Robotics and Automation*, 3:1689-1691, 1986.

[9] Yoshikawa, T: Manipulability of Robotic Mechanisms. *Int. J. Robotics Research*, 4(2):3-9, 1985.

[10] Asada, H. A: Geometrical Representation of Manipulator Dynamics and its Application to Arm Design, *ASME J. Dynamic Systems, Measurement, and Control*, 105(3):131-135, 1983.

[11] Chiu, S.L.: Control of Redundant Manipulators for Task Compatibility. Proc. *IEEE Int. Conf. Robotics and Automation*, pp.1718-1724, 1986.

[12] Salisbury, J.K. and Craig, J.J.: Articulated Hands: Force Control and Kinematic Issues. *Int. J. Robotics Research*, 1(1):4-17, 1982.

[13] Hunt, K.H.: Special Configurations of Robot-Arms via Screw Theory, (Part I). *Robotica*, 4(3):171-179, 1986.

[14] Wang, S-L., and Waldron, K.J.: A Study of Singular Configurations of Serial Manipulators. *ASME J. Mechanisms, Transmissions, and Automation in Design*, 109:14-20, 1987.

[15] Kieffer, J.: Manipulator Inverse Kinematics for Untimed End-Effector Trajectories with Ordinary Singularities. *Int. J. Robotics Research*, 11(3):225-237, 1992.

[16] Kieffer, J.: Bifurcations and Isolated Singularities in the Inverse Kinematics of Linkages and Manipulators. To appear *IEEE Trans. Robotics and Automation*, 1994.

[17] Hollerbach, J.M., and Suh, K.C.: Redundancy Resolution of Manipulators Through Torque Optimization. Proc. *IEEE Int. Conf. Robotics and Automation*, pp.1016-1021, 1985.

[18] Liegois, A.: Automatic Supervisory Control of the Configurations and Behavior of Multibody Mechanisms. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-7(12):868-871, 1977.

[19] Klein, C.A., and Huang, C-H.: Review of Pseudoinverse Control for use with Kinematically Redundant Manipulators, *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-13(3):245-250, 1983.

[20] Yoshikawa, T.: Analysis and Control of Robot Manipulators with Redundancy. In *Robotics Research - The First International Symposium*, Brady and Paul, Eds., MIT Press, pp.735-748, 1984.

DATA REORGANIZATION IN DISTRIBUTED INFORMATION SYSTEMS

A. Mahmood, H. U. Khan and H. A. Fatmi
Dept. of Electronic and Electrical Engineering
King's College London
The Strand, London WC2R 2LS
U.K.
a.mahmood@bay.cc.kcl.ac.uk

Keywords: Databases, distributed data management, data reorganization, distributed systems, distributed information systems, file migration

Edited by: Matjaž Gams
Received: April 13,1994 **Revised:** October 13, 1994 **Accepted:** October 23, 1994

Data reorganization in distributed databases due to non-stationary nature of file utilization pattern and network entities is a highly desirable event to improve system performance and to reduce communication costs. This paper considers a non-blocking approach to data reorganization and presents an algorithm which allows user transactions to run concurrently with the data reorganization activity. It presents an algorithm to identify the opportunities for parallelism in the reorganization process. It also addresses the efficiency of parallel and serial data reorganization techniques in blocking and non-blocking modes and reports some experimental results

1 Introduction

Distributed database reorganization can be defined as a process of changing the physical location of data items, such as files and fragments. Data reorganization may be performed for variety of reasons, such as to reduce the communication cost and response time. The communication cost and response time of a distributed database are functions of its physical location in the network. File usage patterns in distributed systems are characterized by the access and update frequencies for each file-node combination and may vary over time [1, 2]; thus, an optimal allocation at one period may be non-optimal at another period [4]. Hence, data reorganization becomes necessary to achieve the satisfactory system performance [3].

Data can be reorganized in either blocking or in non-blocking mode [5]. In blocking mode, the reorganizer blocks all the user transactions while it reorganizes the data. After the reorganization finishes, normal access of data resumes. Implementation of this strategy is straightforward but it may not fulfill the data availability requirements of certain systems.

In non-blocking mode, database is not brought off line and the reorganization is performed concurrently with the normal usage of the database [5]. Under this strategy, users have access to data while one or more processes reorganize the data. The reorganizer itself may be run at a low priority level so that user transactions should not face long delays [5]. Since data reorganization changes the physical location of data on a network, distributed schemas should be modified accordingly to reflect these changes.

The reorganization of centralized databases has been studied for blocking as well as non-blocking modes [5, 6, 7, 8, 9]. On the other hand, a number of algorithms have been proposed to determine the dynamic data allocations in distributed systems [4, 10, 11, 12, 13, 14] but the problem of data reorganization has not been fully addressed. The problems of monitoring the system performance and generating the near optimal allocations have been discussed in an earlier paper [14]. This paper concentrates on the data reorganiza-

tion problem. Particular attention is given to the non-blocking parallel data reorganization in distributed systems.

The rest of the paper is organized as follows. Section 2 presents algorithms to determine the optimal serial and parallel schedules. Section 3 discusses a strategy to reorganize the data in non-blocking mode. Section 4 analyses the efficiency of parallel and serial reorganization strategies in blocking and non-blocking modes. It presents a number of formulas to estimate the time required to reorganize the data under different strategies. Section 5 presents a case study to explain the working of the proposed algorithms. Section 6 describes a simulation model used to study the relative perform of various data reorganization strategies and presents the simulation results. Section 7 concludes the paper.

2 Serial and Parallel Data Reorganization

In [14], we presented an adaptive data reallocation algorithm to determine the optimal file allocations in a computer network. The reallocation algorithm iteratively improves the existing file allocation to reduce the communication and storage costs. In each iteration, a file copy with a maximum positive value of the objective function is added or deleted, provided, addition or deletion of a file copy does not violate the space, channel and file replication constraints. If a beneficial addition of a file at a particular node violates the system constraints, the less beneficial files at the same node are deleted to accommodate a more beneficial file. The reallocation algorithm not only generates the new file allocation but also generates a serial reorganization schedule (a sequence of reorganization steps) to transform the existing allocation into new optimal allocation such that no constraint is violated during its execution [14]. Each step in a schedule can be one of the following two operations:

a_{ijk} Transfer a copy of file i from node k to node j.

d_{ij} Delete a copy of file i from node j

The subscript i will be used for file index, j, k for node indices, and a * at a respective position would indicate *don't care*. S_p will be used to denote *pth* step in a serial schedule S.

A schedule generated by the reallocation algorithm may have redundant steps. These redundant steps are introduced in a schedule when a less beneficial copy addition step is undone latter in the reallocation process to accommodate a more beneficial file addition step. In this case, a serial schedule has copy addition as well as copy deletion steps for a particular node-file pair. However, a copy addition step followed by a copy deletion step for the same node-file pair is *meaningless* and does not have any effect on the final file allocation and therefore, both steps can be removed from the schedule.

The proposed algorithm to remove the *meaningless* steps from a schedule is given below. For each $S_p = a_{ijk}$, algorithm *Serial_Schedule* searches for $S_q = d_{ij}, (q > p)$ in the remaining *(Total Steps-p)* steps. If a $S_q = d_{ij}$ is found, then both S_p and S_q are removed from the schedule. The time complexity of the algorithm is $O(n^2)$, where n is the total number of steps in the schedule.

The modified schedule can be implemented either in blocking or in non-blocking mode. In blocking mode, the reorganization starts with the first step in the schedule. At the completion of the first step, the second step is initiated and so on. In serial reorganization, the reorganization transactions are initiated strictly in the order that is determined by algorithm *Serial_Schedule*. However, synchronization between the transactions has to be achieved. One approach to achieve synchronization is that each transaction signals its completion to the next transaction in the schedule which is then initiated. The sequential execution of a schedule guarantees its successful completion under normal conditions.

Algorithm Serial_Schedule

```
p = 1
While (p ≤ TotalSteps) Do
  If S_p = a_ij* Then q = p + 1
    While (S_q ≠ d_ij AND q < TotalSteps) Do
      q = p + 1
    If S_q = d_ij Then
      Remove S_p and S_q from S
  p = p + 1
```

One of the disadvantages of the serial reorganization is that a reorganization transaction cannot be started unless all of its preceding transactions successfully complete. This increases the time to complete the reorganization process. Ideally, the reorganization should be performed in parallel, whenever possible, to minimize the reorganization time and to increase the system throughput.

However, scheduling the reorganization transactions in parallel is not straightforward. The difficulty arises from the fact that the data reallocation algorithm evaluates a copy addition and a copy deletion step on the assumption that all the previous steps in the schedule have been executed and their effects have been taken into account. This may result into inter-step dependencies *i.e.,* there could be steps which cannot be executed unless some other steps in the schedule are executed prior to that step. For example, a file may not be added to a node due to space limitation unless some other files are removed from that node. Also, a file cannot be deleted if only one copy of that file is stored in the network. The following four propositions present the situations which lead to inter-step dependencies.

Proposition 1 *A file i cannot be added at a node j if the available space at j is less than the size of file i.*

Proposition 2 *If only one copy of file i is stored in a network then it cannot be deleted unless a copy of i is stored at another node.*

Proposition 3 *A node cannot supply a file if it does not have one or if the file has already been removed from that node.*

Proposition 4 *If a node has been schedule to send a file copy to some other node, it must acquire one before sending it.*

Let us consider a serial schedule $S = \{S_1, S_2 \ldots, S_n\}$. The step $S_{p-1}, (1 < p \leq n)$ is generated prior to S_p by the data reallocation algorithm. A serial schedule is *logically correct* if S_p can successfully be executed without executing $S_q, (q > p)$ or a subset of $\{S_{p+1}, S_{p+2}, \ldots, S_n\}$.

It can easily be proven that a schedule generated by the reallocation algorithm is logically correct since a step is only included in the schedule if it can be executed successfully at the time of its evaluation. To establish the logical correctness of a serial schedule S, the following tests derivable from the above propositions should be applied.

1. If there exists a $S_p = a_{ij*}$ in a schedule S and the available space at node j is not enough to store a copy of file i, then there must exists one or more $S_q = d_{*j}, (q < p)$ such that they free enough storage space to store file i at j.

2. If there exists an $S_p = d_{i*}$ in a schedule and only one copy of i is stored in the network, then there must exist a $S_q = a_{i**}, (q < p)$ in S.

3. If there exists a $S_p = a_{i*k}$ in a schedule S and k does not initially have a copy of i then there must exit a $S_q = a_{ik*}, (q < p)$ in S.

4. If there exists a $S_p = a_{i*k}$ in a schedule S and k initially has a copy of i then must not exist a $S_q = d_{ik}, (q < p)$ in S.

Definition 1 *If $(S_1, S_2, \ldots, S_n)$ is an ordered sequence of all the steps in a schedule S, and $D_p = \{S_q : 1 \leq q < p\}$ such that S_p cannot be executed unless all $S_{q's} \in D_p$ are executed before S_p, then S_p is dependent on D_p. D_p is called dependent set of S_p.*

Definition 2 *If D_p and D_q are dependent sets of S_p and S_q respectively such that $S_p \notin D_q$ and $S_q \notin D_p$ then S_p and S_q are logically independent.*

Proposition 5 *For a logically correct schedule S, $S_1 = \phi$.*

Theorem 1 *If a step S_p is executed after all $S_{q's} \in D_p$ then the logical correctness of schedule S is preserved.*

Proof: Assume that there is a step $Sr \notin D_p$ that must be executed before S_p to preserve the logical correctness of S. From the definition of the dependent set, S_r must be in D_p which contradicts that $S_r \notin D_p$. Also assume that S_p can be executed before a step $S_r \in D_p$ such that the logical correctness of S is preserved. This implies that S_r cannot be in D_p which contradicts that $S_r \in D_p$.

□

Theorem 2 *Logically independent steps can be executed in parallel.*

Proof: Suppose there exist a step S_p in a group of logical independent steps that must be executed before S_q in the very group. This means that S_q must be in dependent set of S_p. But definition of logically independence rules out this possibility. □

Using the above theorems and tests, the steps which might be executed in parallel can be determined. One approach is to arrange the steps in a hierarchy such that the steps at the same level are logically independent and hence can be executed in parallel; and the steps in their dependent sets are at lower levels of the hierarchy and should be executed before them.

The proposed algorithm which determines the parallelism in the reorganization transactions consists of two parts. The first part determines the dependent set for each step in S by scanning the serial schedule S for all the possible inter-step dependencies and identifies them accordingly. The second part construct a reorganization hierarchy (a forest like structure) by using the information of inter-step dependencies determined at the first step. The nodes of the hierarchy represent the reorganization transactions and the arcs show the inter-step dependencies. The reorganization hierarchy is arranged in such a way that the steps at the same level are logically independent and the steps in their dependent sets are at the lower levels of the hierarchy. It follows from theorem 1 and 2 that the steps at the same level can be executed in parallel provided all the steps at the lower levels are executed first. The time complexity of the algorithm is $O(n^2)$, where n is the total number of steps in the serial schedule.

Algorithm Parallel_Schedule
Determine Dependent set of each step in S
$StepsAtLevel[1] = \phi$; $l = 1$
For m=1 to TotalSteps **Do**
If $D_m = \phi$ **Then**
StepsAtLevel[l]=StepsAtLevel[l]+$\{S_m\}$
UptoLevel[l]=StepsAtLevel[l]
Repeat
$l = l + 1$
StepsAtLevel[l]=ϕ
For m=1 to TotalSteps **Do**
If $D_m \subset UptoLevel[l-1]$ **Then**
StepsAtLevel[l]=StepsAtLevel[l]+$\{S_m\}$
UpToLevel[l]=UpToLevel[l-1]+
StepsAtLevel[l]
Until (S is empty)

Algorithm Determine_DependentSet
For p=1 to TotalSteps **Do**
If $S_p = a_{ijk}$ **Then** $q = p$
AS=Free space at node j before reallocation
While($AS < FileSize[i]$ AND $q > 1$) **Do**
$q = q - 1$
If ($S_q = d_{*j}$) **Then**
$D_p = D_p + \{S_q\}$ (Test 1)
AS=AS+FileSize[*]
If ($S_q = a_{*n*}$ **Then**
AS=AS-FileSize[*]
If copies[i]=1 **Then** $q = p + 1$
While ($q < TotalSteps$ AND $S_q \neq d_{i*}$) **Do**
$q = q + 1$
If $S_q = d_{i*}$ **Then**
$D_q = D_q + \{S_p\}$ (Test 2)
$q = p$
While ($q < TotalSteps$AND $S_q \neq d_{ik}$ **Do**
$q = q + 1$
If $S_q = d_{ik}$ **Then**
$D_q = D_q + \{S_p\}$ (Test 3)
$q = p - 1$
While($q > 1$ AND $S_q = a_{ik*}$) **Do**
$q = q - 1$
If $S_q = a_{ik*}$ **Then**
$D_p = D_p + \{S_q\}$ (Test 4)
copies[i]=copies[i]+1
Else copies[i]=copies[i]-1

Having the above algorithms, one can construct the reorganization hierarchy to identify the parallelism in the reorganization process. However, the number of steps which can actually be executed in parallel depend on the processing capabilities of the nodes and the channel bandwidth.

3 Non-blocking Data Reorganization

Non-blocking reorganization is possible if the reorganization process is treated as a set of trans-

actions such that each transaction either deletes or adds a file copy. Thus, data reorganization can be performed in non-blocking mode by executing a sequence of two special transactions: Add-copy(x_{jk}) and Delete-copy(x_j). The first transaction copies file x from node k to node j and the second transaction deletes a copy of file x from node j.

Since user transactions may be issued at any time, read-write conflict on distributed schema and data files may arise. Moreover, a copy being added may be updated by the user transactions when it is in transient. The key to the non-blocking reorganization is the method that maintains serializability among the user and the reorganization transactions. A blocking mechanism is one of the approach that can be used to maintains serializability [15].

Let us assume that with every file x, a directory $d(x)$ is associated which is fully replicated. Each $d(x)$ copy stores information about the location of copies of file x and is denoted by $d(x).data\_location$. A directory copy stored at node n is denoted by $d_n(x)$. every user transaction reads the related directory entry before reading and writing a data file. The transactions can set four types of lock on a directory entry. These are: a-lock, d-lock, user-read-lock and user-write-lock. Locks on different copies do not conflict, whereas locks on the same copy conflict as given in Table 1. A x indicates a lock conflict and a - denotes no conflict. Read and write locks on a data file conflict in the normal way [15]

The concurrency control can be achieved by 2PL [16]. When a transaction commits or aborts, the system respectively commits or aborts the transaction at each node where it is active. This ensures the atomicity of transactions. The deadlock can be handled using the existing algorithm [17].

The algorithms to add and delete a file copy, user read and user write are given below. These algorithms are slightly modified version of the algorithms proposed in [18] for concurrency control in distributed databases.

Add-copy(x_{jk})
The transaction starts at node j

Step 1. Set an a-lock on $d_j(x)$

Step 2. Set a read-lock on x_k and read it

Step 3. Set a write-lock on x_j

Step 4. Set an a-lock on all available copies in $\{d(x).data\_location - d_j(x)\}$

Step 5. Do in parallel step 6 and 7

Step 6. For all available copy of $d(x)$, modify $d(x).data\_location$ by including site j in $d(x).data\_location$ and release a-locks on respective $d(x)$

Step 7. Write x_k read in step 2 and release read-lock on x_k and write-lock on x_j.

Delete-copy(x_j)
The transaction starts at node j

Step 1. Set a d-lock on $d_j(x)$ and read $d_j(x).data\_location$.

Step 2. Set a d-lock on each available copy of $d(x)$ in $\{d(x).data\_location - d_j(x)\}$.

Step 3. Delete file copy x_j

Step 4. Modify $d(x).data\_location$ of all the available copies of $d(x)$ by removing site j from $d(x).data\_location$. Each such modification releases the d-lock on the respective $d(x)$.

User Read(x)
Let the transaction starts at node j

Step 1. Set a user-read-lock on $d_j(x)$ and read it. Find an available copy with minimum transmission cost. Let this copy be x_k.

Step 2. Set a read-lock on x_k.

Step 3. If a delete-copy(x_k) transaction has no d-lock on $d_k(x)$ then read x else abort the transaction.

Step 4. Release all the locks set at step 1 and 2.

User Write(x)
Let the transaction starts at node j

lock type	a-lock	d-lock	user-read lock	user-write lock
a-lock	x	x	-	x
d-lock	x	x	-	-
user-read-lock	-	-	-	-
user-write-lock	x	-	-	-

Table 1: Lock compatability matrix for directory read-write

Step 1. Set a user-write-lock on $d_j(x)$.

Step 2. For every available copy $x_k, k \in \{d_j(x).data\_location - j\}$, set user-write-lock.

Step 3. Set a write-lock on each available copy of x.

Step 4. If a delete-copy(x_k) has no d-lock on $d_k(x)$ then write the item in every copy of x locked at step 2.

Step 5. If a delete-copy(x_k) has a d-lock on $d_k(x)$ then ignore x_k and write the item in the remaining copies locked at step 2.

Step 6. Parallel with step 4 and 5 release all write-locks obtained by the user transaction.

Step 7. Release all user-write-locks set at step 1 and 2.

Note that we allow the Delete-copy(x) transaction to run currently with the user transactions which could result in abortion of some of the user transactions. In such a case, the aborted transactions have to be restarted.

4 Reorganization Time

Reorganization time (RT) can be defined as the time required to complete the reorganization process. The RT depends on the number of files to be received and transferred among all the nodes, degree of parallelism, physical limitations of the channel and the processing capabilities of the nodes.

Let us assume that a node j receives data at a rate of g_j and node k sends data at a rate of t_k, different pairs of nodes can communicate with each other and the time to delete a file is negligible as compared to the time required to add a copy. Then, the expected execution time for *pth* add-copy(x_{jk}) transaction in the blocking mode is given by:

$$t_p = FileSize(x)/min(g_j, t_k) \quad (1)$$

If a total of F files are to be added , then

$$RT = \sum_{p=1}^{F} t_p \quad (2)$$

The reorganization time in parallel blocking mode is determined by the time taken by the longest copy addition transaction at each level of the reorganization hierarchy. However, in the non-blocking mode, the frequency of user transactions and the conflicting locks could increase RT. If t_l is the execution time of the longest reorganization transaction at level l and total number of levels in the reorganization hierarchy are L, then

$$RT = \sum_{l}^{L} t_l \quad (3)$$

Where it is assumed that the reorganization transactions at a higher level are not started unless all the transactions at the lower levels of the hierarchy are completed.

The response time of a transaction of type q, decomposed into M step by the query optimizer, can be determined using an open queuing network model. For a point to point network, assume CPU processing capability at site j is μ_j MIPS, disk has an average IO time I_j and the link between site j and k has a capacity of λ_{jk} Mbytes/s. Let us also define the follows notations. Let α_{qs} be the

processing required by step s of query q excluding the communication overhead, A_c be the average processing required for receiving and sending messages and data, γ_j be the load on CPU at site j, τ_{jk} be the communication traffic between site j and k, V_{qsjk} be the amount of traffic generated between node j and k due to step s of query q and β_{qsj} be the disk IO required at site j for step s of query q. The average response time can be expressed by

$$T = 1/\chi \sum_q \chi_q (P_q + C_q + D_q) \tag{4}$$

where

$$\chi = \sum_q \chi_q \tag{5}$$

and χ_q is the arrival rate of query q. Processing time P_q, queuing delay at a communication link C_q , and disk IO time D_q can be estimated using a M/M/1 general queuing discipline and are given below.

$$P_q = \sum_j \sum_s \frac{\alpha_{qs} x_{qsj} + A_c Y_{qsj}}{\mu_j - \mu} \tag{6}$$

where

$$\mu = \sum_q \chi_q \sum_s \alpha_{qs} x_{qsj} + \sum_q \chi_q \sum_s A_c Y_{qsj} \tag{7}$$

where

$$x_{qsj} = \begin{cases} 1 & \text{if } sth \text{ step is executed} \\ & \text{at node } j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and

$$Y_{qsj} = \begin{cases} 1 & \text{if site } j \text{ is involved in the} \\ & \text{communication at step } s \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The value of C_q is estimated by

$$C_q = \sum_j \sum_q \sum_s \left(\frac{L_{jk} \tau_{jk} Z_{qsjk}}{\lambda_2 - \lambda \tau_{jk}} + \frac{V_{qsjk}}{\lambda} \right) \tag{10}$$

where L_{jk} is the average message length between node j and k and is given by

$$L_{jk} = \frac{\sum_q \sum_k V_{qsjk}}{\sum_q \sum_k Z_{qsjk}} \tag{11}$$

and

$$Z_{qsjk} = \begin{cases} 1 & \text{if } V_{qsjk} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and finally, assuming the exponential disk service time with mean d_j,

$$D_q = \sum_j \sum_s \frac{I_j \beta_{qsj}}{d_j - \sum_k \sum_l \chi_l \beta_{klj}} \tag{13}$$

The above formulas can be used to determine the expected response time for reorganization as well as user transactions. After calculating the response time of all the reorganization transactions, the total reorganization time can be estimated by using equation 2 and 3. The analysis given in this section can easily be extended for networks in which data transmission takes place in packets rather than in bit streams.

Given the above formulas, the reorganization strategies can be evaluated under different implementation approaches and environments. For a specific environment, an accurate set of parameters may be determined via measurement or simulation.

5 A Case Study

To explains how algorithms *Serial_Schedule* and *Parallel_Schedule* work, a case study is presented in this section. Consider a network of 5 nodes and 6 files. Storage capacities at various nodes, initial and the new allocations of each file and the file sizes are shown in Table 2 and Table 3. The minimum degree of replication is one for every file. Let the serial schedule generated by the reallocation algorithm is as follows:

$$S = \{a_{412}, d_{31}, a_{613}, d_{42}, d_{54}, a_{643}, d_{61}, a_{212}, d_{63}, a_{525}, d_{55}, a_{532}, a_{654}, d_{22}\}$$

Note that a_{613} followed by d_{61} are redundant and can be removed from S by executing *Serial_Scedule* which produces the reduced schedule S', where

$$S' = \{a_{412}, d_{31}, d_{42}, d_{54}, a_{643}, a_{212}, d_{63}, a_{525}, d_{55}, a_{532}, a_{654}, d_{22}\}$$

To generate a parallel schedule, dependent sets of each step in S' are obtained using *algorithm*

Node	1	2	3	4	5
Capacity	45	50	35	40	25

Table 2: Storage capacities at each node

Determine_DependentSet and are shown in Table 4. The steps which can be executed in parallel can be obtained by *Parallel_Schedule* and the reorganization hierarchy is shown in Figure 1. The data reorganization is accomplished in 5 stages. For example, at the first stage, S_1, S_2 and S_4 could be executed in parallel, S_3, S_5 and S_6 could be executed at stage 2 and so on.

6 Simulation Results

A hybrid simulation models has been used to analyze and compare the relative performance of the serial and the parallel reorganization strategies in blocking and non-blocking modes. A discrete event simulation model [19] is used to model the arrival of transactions at network nodes and point to point transmission of data between nodes. The analytical the equations of M/M/1 queuing network and equations given in section 4 have been used to model the internal processing at each node.

In the simulation model, users of the distributed system generate transactions which retrieve and update local and remote files. The transactions are randomly generated on the defined files. At each node, a local source generates local queries with a poisson arrival distribution while a global source generates distributed transactions with a poisson arrival distribution. For every distributed transaction, an entry node is chosen with a uniform distribution. The entry node is also defined to be the result node for that query. Every transaction is restricted to query or update a single file.

The queuing network consists of a finite number of nodes. Each node has an associated queue on which a transaction may wait prior to receiving a service. A transaction is present at a node if it is waiting for or receiving a service at that node. The transaction arriving at a node are queued in a first-come-first-serve order. A maximum level of multiprocessing is defined for each node.

To make the model simpler, each transaction type is assigned a fixed required amount of processing and volume of data received and sent. A program constructs a response table for each transaction type at each node using the queuing analysis and the response time analysis given in section 4.

Moreover, it is assumed that the network is fully connected with full duplex lines connecting each pair of nodes. The processing time at the communication processor and the propagation delays are assumed to be negligible, there are enough buffers at each node to hold all the incoming data, each channel is error free, acknowledgment and locking overheads are negligible and each message consists of a single packet. Therefore, neither the time it takes to receive all the packets belonging to a message at the destination nor the re-sequencing delays are taken into account.

Simulation was run for 8 serial schedules generated for a network of 8 nodes and a database with 14 files. The optimal serial and the parallel schedules were obtained using algorithm *Serial_schedule* and *Parallel_Schedule* respectively. The synchronization between the transactions was achieved by monitoring the execution of the transactions. A reorganization transaction is initiated only if all the transactions in its dependent set have successfully been executed.

To compare the relative performance of the reorganization strategies, equivalent serial and parallel schedules were run in both blocking and non-blocking modes and the reorganization time was measured. Figure 2 shows the percentage increase in RT for serial non-blocking strategy when compared with RT for serial blocking strategy. The average increase in RT is 3.39% for the non-blocking serial strategy.

Figure 3 shows the percentage decrease in RT for the blocking and the non-blocking parallel strategies as compared to RT for the blocking serial strategy. According to the results, the parallel blocking strategy took up to 45.01% less time as compared the time taken by the blocking serial strategy. However, the blocking parallel strategy

File	File size	Initial allocation (nodes)	New allocation (nodes)
1	15	1,2	1,2
2	10	2,5	1,5
3	10	1,3	3
4	20	2,4	1,4
5	15	4,5	2,3
6	12	3	4,5

Table 3: File sizes, initial and new file allocations

step	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
D_n	ϕ	ϕ	$\{S_1\}$	ϕ	$\{S_4\}$	$\{S_2\}$	$\{S_5\}$	$\{S_3\}$	$\{S_8\}$	$\{S_7, S_8\}$	$\{S_5, S_9\}$	$\{S_6\}$

Table 4: Dependent sets

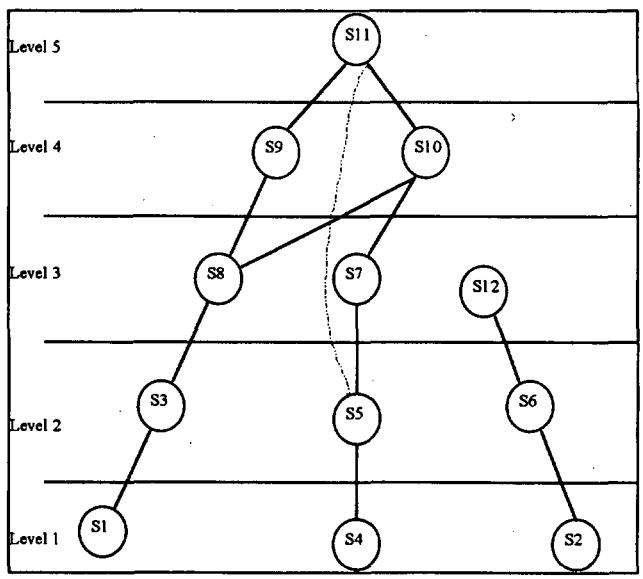


Figure 1: Reorganization hierarchy

is comparatively faster than the non-blocking parallel strategy.

Figure 4 shows the decrease in RT for the non-blocking parallel strategy as compared to the non-blocking serial strategy. The difference in RT between the non-blocking serial and the non-blocking parallel strategies is less than the difference in RT between the non-blocking parallel and the blocking serial strategies. It is evident from these results that the non-blocking strategies take more time to reorganize the data as compared to the time taken by their counterpart blocking strategies. This is because of the fact that user transactions are allowed to run concurrently with the reorganization transactions and the processing and the channel resources are shared between them which increase the RT for non-blocking strategies. Another reason is that the concurrent execution of transaction may result in conflicting locks and, therefore, to maintain serializability, either the reorganization or the user transactions are delayed. The scenario of conflicting locks may also reduce the actual parallelism that can be achieved in data reorganization activity.

The number of transactions which are aborted due to the reorganization process depends on the arrival rate of user queries for the file being reorganized and the time it takes to remove a file and update the distributed schema. In the present study, an average number of user transactions which were forced to abort, were less than 2% of the total transactions.

7 Conclusion

The non-blocking parallel data reorganization in distributed databases is a highly desirable event which allows user transactions to run concurrently with the data reorganization transactions. The number of user transactions affected by the data reorganization process is proportional to the reorganization time. Therefore, the reorganization time should be minimized by exploiting the parallelism in the data reorganization process. The algorithm *Parallel_Schedule* identifies the possible parallelism in the serial schedule. The algorithm maintains the inter-step dependencies by applying a set of tests and theorems. The overhead of this approach is the increase in the response time and abortion of user transactions.

References

[1] B. W. Wah: *File placement on distributed systems.* IEEE Computer, 17(1), pp. 23-32 (1984).

[2] H. M. Pocar: *File migration in distributed computer system.* Ph.D. Thesis, U. C. Berkeley (1982).

[3] B. Gavish and O. R. L. Sheng: *Dynamic file migration in distributed computer systems.* Comm. of the ACM, 33(2), pp. 177-189, (1982).

[4] K. D. Levin and H. L. Morgan. *A dynamic optimization model for distributed databases.* Operations Research, 26(5), pp. 824-835 (1978).

[5] G. H. Sockut and R. P. Goldberg: *Database reorganization - principles and practice.* Computing Surveys, 11(4), pp. 371-395 (1978).

[6] H. Mendelson and U. Yechiali: *Optimal policies for database reorganization.* Operations Research 29(1), pp. 23-36, (1981).

[7] S. B. Yao, K. S. Das and T. J. Teorey: *A dynamic database reorganization algorithm.* ACM TODS, 1(2), pp. 159-174 (1976).

[8] E. Omiecinski, L. Lee and P. Scheuermann: *Concurrent file reorganization for record clustering: a performance study.* in Int. Conf. on Data Engineering, pp. 265-272 (1992).

[9] C. Yu, K. Lam, M. Siu and C. Suen: *Adaptive record clustering.* ACM TODS, 10(2), pp. 180-204 (1985).

[10] J. E. Ames and D. Foster: *Dynamic file assignment in star network.* in Proc. Computer Networks Symposium, pp. 36-40 (1977)

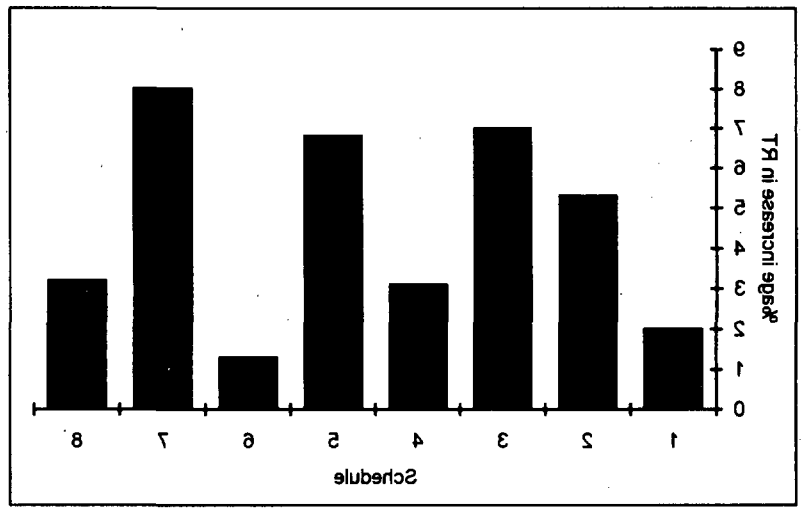


Figure 2: Percentage increase in RT for serial non-blocking strategy as compared to serial blocking strategy

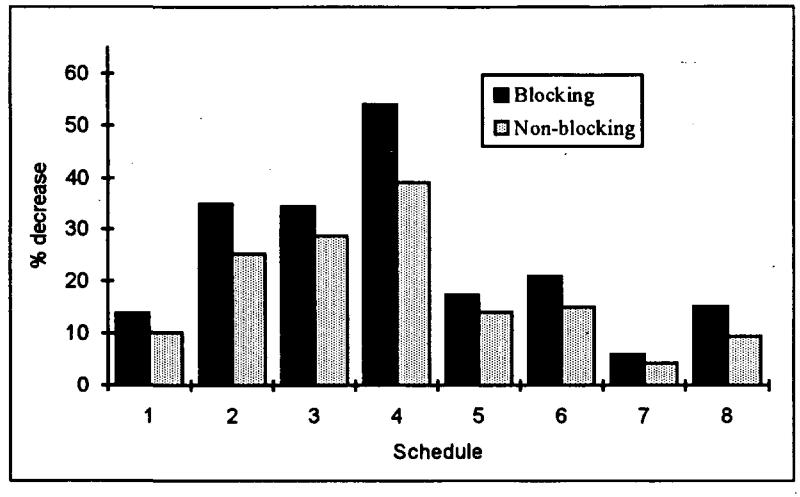


Figure 3: Percentage decrease in RT for parallel strategies as compared to serial blocking strategy

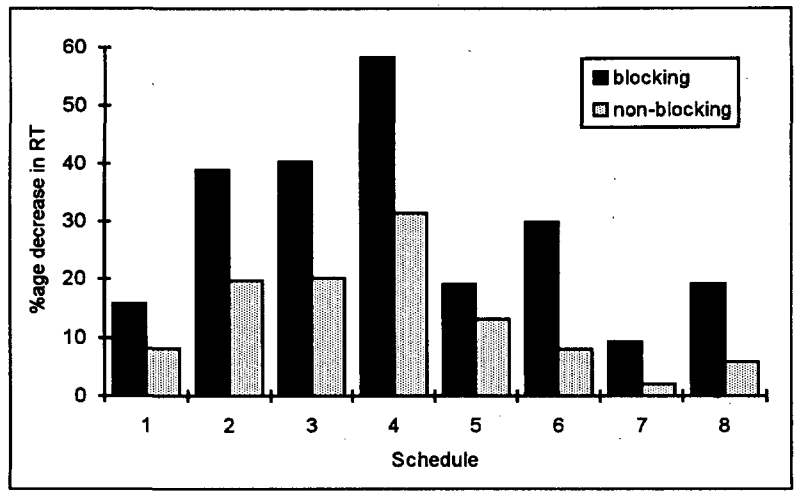


Figure 4: Percentage decrease in RT for parallel strategies as compared to non-blocking serial strategy

[11] C. T. Yu, M. Siu and C. H. Chen: *Adaptive file allocation in star computer network.* IEEE TOSE, SE-11(9), pp. 959-965 (1985).

[12] A. Segall: *Dynamic file assignment in a computer network.* IEEE TOAC, AC-21(2), pp. 161-173 (1976).

[13] X. Du and F. J. Maryanski: *Data allocation in a dynamically reconfigurable environment.* in Proc. Int. Conf. on Data Engineering, pp. 74-81 (1988).

[14] A. Mahmood, H. U. Khan and H. A. Fatmi: *Adaptive file allocation in distributed information systems.* Informatica, 18(1), pp. 37-46 (1994).

[15] D. Bell and J. Grimson: *Distributed database systems.* Addison-Wesley, England (1992).

[16] B. G. Lindsay, P. G. Selinger et al.: *Notes on distributed databases.* in Distributed databases, Drattan and Pool (eds.), Cambridge University Press, New York (1980).

[17] E. Knapp: *Deadlock detection in distributed databases.* ACM Computing Surveys, 19(4), pp. 303-328 (1987).

[18] P. A. Bernstein and N. Goodman: *An algorithm for concurrency control and recovery in replicated distributed databases.* ACM TODS, 9(4), pp. 596-615 (1984).

[19] J. Banks and S. J. Carion: *Discrete event system simulation.* Prentice-Hall, New Jersey (1978).

EVALUATING THE MANUFACTURING SIMULATOR "WITNESS" ON AN AUTOMATED MANUFACTURING SYSTEM

Vlatka Hlupic, Ray J. Paul
Brunel University
Department of Computer Science
Uxbridge, Middlesex UB8 3PH
United Kingdom

Keywords: manufacturing system, simulation, simulation package, software evaluation

Edited by: Matjaž Gams
Received: April 20,1994 **Revised:** October 13, 1994 **Accepted:** October 23, 1994

The application of the manufacturing simulator "WITNESS" to an automated manufacturing system is described. Some software tools for the simulation of manufacturing systems are reviewed, as well as the most important characteristics of the simulation package "WITNESS". The case study model is described together with problems experienced during modelling. This provides a critical analysis of the capabilities and shortcomings of the software. The conclusions outline the successes and failures of the simulator for modelling, and what lessons and further ideas were obtained from this study.

1 Introduction

Automated manufacturing systems are being used increasingly in many industries in order to improve productivity and reduce costs. These systems are complex, dynamic and stochastic. In addition, they involve large capital investments. Because of this, simulation is often used to investigate both the configuration alternatives of the system and its potential operation modes.

The growing acceptance of simulation has resulted in an increase in the number of simulation packages available on the market. Accordingly, there is an increase in the number of publications on software evaluation. Most of the publications evaluate several simulation software tools, providing basic information about the software being analyzed. Some examples of such publications are those written by Law and Haider [10], Banks et al [1], Van Breedam et al [19], Ekere and Hannam [6], Hlupic and Paul [8] and Carrie [3]. On the other hand, publications that provide a review of one simulation package are predominantly written by vendors, with a potential for bias and a lack of self-criticism.

This paper presents an example of the use of the simulation package "WITNESS" for modelling a real automated manufacturing system, with the emphasis on a critical and fairly detailed evaluation of this software tool. Following a review of some software for manufacturing simulation, the system being modelled is described. The main characteristics of the model developed are presented as well as the difficulties experienced during modelling. Positive features and shortcomings of the software used are addressed. The last two sections provide a summary of research findings and a discussion of the suitability of "WITNESS" and other similar software tools for modelling complex real systems.

2 Manufacturing Simulation Software and WITNESS

2.1 Simulation Languages versus Simulators

There are many different ways of classifying simulation software [13]. According to Law and Kelton [11], simulation packages can be classified as sim-

ulation languages and simulators. When a simulation language is used, the model is developed by writing a program using the modelling constructs of a language. This enables modelling of almost any type of system, but it might be tedious and time consuming.

On the other hand, a simulator allows the modelling of a specific class of systems with little or no programming, as it is a data driven environment for a limited problem domain.

Simulation languages are general in nature, although some of them have special features for modelling manufacturing systems. For example, SIMAN [12] and SLAM II [14] have manufacturing modules for automated guided vehicles and conveyors. Other examples of simulation languages include SIMSCRIPT II.5 [15], MODSIM II [2], GPSS/H [16], SIMULA [7], ECSL [4] and PCModel [20].

When a simulator is used for model development, models are typically developed by the specification of model parameters via menus. As little or no programming is needed, modelling time is usually significantly reduced. There are many manufacturing oriented simulators on the software market. Some of the most popular are WITNESS [21], SIMFACTORY II.5 [17], XCELL+ [5], and AutoMod II [18].

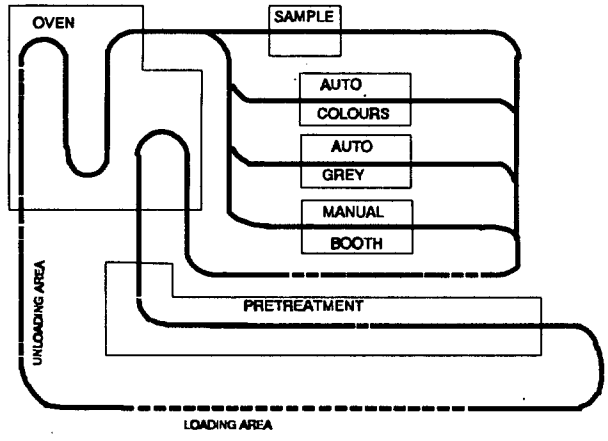


Figure 1: The layout of the system

2.2 WITNESS

WITNESS is a data driven simulator, which employs a visual interactive approach to modelling using pre-defined elements to represent manufacturing processes. Models are developed through three main phases: Define, Display and Detail.

. WITNESS physical and logical elements are created in the Define phase. Each element has its separate form, which has to be filled in with relevant data. In this phase, the names and their quantity have to be specified for the majority of elements.

In the Display phase, a graphical display of the model is created. Icons representing elements are designed using the Icon Editor. The model layout is built up by positioning elements on the screen using a mouse. Characteristics such as the position of parts and labour, icons for physical elements and directional flows are chosen from menu forms. There is also a Screen Editor which enables the drawing of text, and of different lines and shapes to enhance the graphical display of the entire model.

In the Detail phase, the user has to specify how elements operate and how they interact with other elements. For each element, pre-defined forms are filled with information such as cycle times, labour requirements, speeds, capacities and breakdown patterns. In addition to the constant values, this information may include functions, statistical distributions and variables.

Pre-defined physical elements represent manufacturing equipment such as Parts, Machines, Labour, Buffers, Conveyors, Vehicles, Trucks, Tanks and Fluids. Logical elements handle the model's logic. For this purpose, the user can define part Attributes, Variables, use in-built functions or write their own, specify Input and Output rules for part routing, write Actions to describe changes in the status of elements or specify Shift patterns.

3 The Case Study: A Powder Coating System

The automated system for the electrostatic powder coating of metal components is installed in the United Kingdom. Although this system is a part of the factory that produces electronics products, it can be considered as a separate unit due to its specific characteristics and function. The system paints various metal components using the methods of electrostatic powder coating. These components are produced in flexible manufacturing cells installed in the same factory, and after coating are assembled with other components to create final products.

The layout of the system is shown in Figure 1. The entire system consists of a large overhead conveyor chain passing through several processing areas. Components to be coated are attached to flight bars, which are mounted on the conveyor. The number of parts per flight bar depends on the product type, of which there is a range of almost two thousand different part types. For example, one large part may need two flight bars, whilst on the other hand five hundred small parts can be jigged together and hung on one flight bar. After the parts have been loaded on the flight bars in the loading area, they are transported through several processing areas, prior to their unloading after the last processing stage. The first stage is pretreatment, where parts are degreased and washed in order to be properly painted. Following pretreatment, parts are transported to the oven, where they are dried. Finally, after drying, the parts are ready for powder coating (painting).

There are two automatic and two manual booths for painting. One manual booth is used only when one or two parts are painted as a sample, whilst the other is used when the batch size is small (less than 20 parts) or when a particular batch has high priority. In all other cases automatic booths are used, where the parts are automatically coated. One automatic booth is dedicated to one special colour, whilst all other colours are painted on the other booth.

Following coating, parts are transported to the oven, where they are baked in order to preserve the coating. After this last stage of processing, parts go to the unloading area, where they are unloaded from the flight bars, separated and moved out of this system. Only the main characteristics of the system have been described here. It is not possible to succinctly describe its full complexity, because there are many additional details, such as various features of the parts which make them distinct from hundreds of different part types, details of the manpower in the system, details about breakdowns of the conveyor and spraying guns, or information about complicated shift patterns.

4 The case study

4.1 Simulation model

The simulation model was developed using the "WITNESS" software package. The main reason why this package was chosen is its ability to allow additional programming for modelling specific features of the system. This was necessary because of the complexity of the system being modelled.

The simulation model was developed gradually. An increasing number of details were added to the model and tested repeatedly. This iterative process resulted in more than sixty different versions of the model. The final version contains more than 220 model elements. Whilst physical elements of the system (parts, conveyors, processing stations, labour etc.) were modelled by specifying their parameters in predefined forms, logical elements had to be additionally programmed. Thus, many additional functions were written, conditions for routing programmed, and part attributes defined.

For example, the part attributes defined relate to the colour, batch size, batch priority, number of parts per flight bar, batch number, masking requirements, the number of the booth on which the batch will be painted, the total number of flight bars required for a particular batch, and the requirements for a manual finish to the painting if the part is complex. Conditions in the form of input and output rules were programmed for many parts of the model. For example, when the batch has to be routed to one booth for painting, attributes such as colour, batch size or priority are tested and, depending on the values of these attributes, the batch is routed in the appropriate direction. A number of functions were written in order to handle specific logical features of the system.

After each iterative change, the model was thoroughly tested. In addition to animation, which provided the most significant help for model verification, many other facilities for testing the model are available. For example, the values of functions and variables can be displayed and changed dynamically as the simulation progresses. It is also possible to obtain information about the state of particular elements, and information about the attributes of the parts present in the specific elements.

The final version of the model gave results (throughput) that varied about 1from the real values, so this model was accepted as a basis for further experimentation. Several experiments were run, and the results obtained were compared to the values obtained in the real system. A detailed description of experimentation is provided in [9].

4.2 Problems experienced

Several features of the system under consideration have not been modelled completely or have been omitted from the model because of software functionality. Examples of these features are as follows.

WITNESS allows the commencement of the machine setup only when the part arrives at the machine. In the system under consideration, setup (ie. cleaning of the painting booths and spraying guns) starts as soon as one batch is finished, and when it is known that the next batch to arrive at a particular booth is to be coated in a different colour. Production in the real system does not stop because of the setup. The batch that requires an alternative booth for coating is loaded whilst another booth is being cleaned. Waiting to model the setup until a new batch had already arrived at the machine caused an unacceptable delay in processing which had a significant impact on throughput. After many unsuccessful attempts to model this feature according to the situation in the real system, setup modelling was abandoned. This did not influence the validity of the results regarding throughput, but it had some impact on labour utilization. Since setup was omitted from the model, the performance of labour was slightly underestimated. This did not represent a problem, because the number of colour changes on each machine was monitored, and the time spent on setup during the simulated time could be calculated and added to the value of labour utilization.

Another problem experienced occurred in an attempt to model a search of the buffer content, and pulling out from the buffer a batch with a specific characteristic. For example, after one batch was loaded, the attributes of the batches placed in the buffer ought to be investigated and then the batch with the same colour (as the previously loaded batch) loaded next. When this feature was modelled, only the first part within a particular batch that satisfied the condition (the same colour) was pulled out from the buffer, and after that the program stopped. This part of the logic was abandoned and parts were pulled out from the buffer on a FIFO basis, which in the end did not influence the results significantly.

It is not possible to pull a part from a specific position of an element (e.g. a machine) although it is possible to push a part to a specific position. In the system, after baking in the oven, the parts are unloaded from any position on the conveyor apart from the newest three (which must be left for cooling of the parts after baking, prior to unloading). The parts are unloaded simultaneously from final positions. In the model, unloading stations that represented unloading activity could pull parts only from the front position on the conveyor, namely from one flight bar at a time. This did not have an important influence on the final results, but showed a weakness of the package.

The above mentioned examples describe what particular features of the system being studied it has not been possible to model that corresponds to the situation in the real system. Although in the end their abandonment has not influenced the results significantly, they are a good illustration that even such a flexible package as WITNESS does not allow the modelling of all specific details. There are many other features of the package which have not eased the modelling process. Some of them will be probably improved in subsequent versions of the package. However, a lot of modelling time could have been saved, had the shortcomings listed below not been present.

Buffers are passive, which means that it is neither possible to pull parts from, nor to push parts to, buffers. This caused problems when parts in the buffer had to be sent to masking stations. Several dummy machines were used to pull parts

from the buffer and send them to masking stations, which additionally complicated the logic of the model.

There is no automatic increase of buffer capacity. This sometimes meant that only part of a batch was placed in the buffer and that another part was lost or the model simply stopped. A separate function was therefore written in order to check for free space in the buffer, before the batch was placed in it.

The modelling process could be speeded up if the package allows the copying of physical and logical elements. Similar elements could then be modelled by copying ones already defined and making some minor alterations.

Another example of the package weaknesses listed here, and perhaps the most important one, is the problem of software reliability. That is, the program 'crashed' many times for no apparent reason, and the only thing that could be done was to reboot the computer.

Similar problems occurred due to a shortage of computer memory. The software itself has a significant hardware requirement (4MB of RAM). In addition, the model developed was quite complex and during experimentation the memory space necessarily increased. Due to this memory problem, the program 'crashed' many times during experimentation, or when the buffer content had to be listed using the EXPLODE function. These problems relating to software reliability and memory problems, which could have probably been eliminated by using a more powerful computer, cost a lot of time.

5 An evaluation of WITNESS

5.1 Positive features

Several general features of this simulator make it adequate for the simulation of manufacturing systems. WITNESS is a data driven, manufacturing oriented simulator, with the facility to add some program code. Its Windows based environment with pull- down menus makes it very user friendly and it is easy to use once it is learnt. Modelling transparency is good.

The Visual aspects are quite good, with easy to use icon and screen editors that can produce nice graphical displays of the models, using multiple colours. Icons can be stored in the icon library. These icons can also be manipulated. Full animation is provided, with the movement of elements proportional to the time needed for a change in their state. Panning and zoom function are provided. Graphics can be switched on or off. Icons can be changed during the simulation, when a change of element represented by a particular icon occurs.

With respect to coding aspects, WITNESS provides an internal language which enhances modelling flexibility. The user can write code to handle special logical features. The syntax of the code is fairly readable and precise. A number of built-in functions are provided. The user can also write his/her own functions, which can invoke built-in functions. Global variables accessible by all elements in the model can be used.

The Efficiency of this simulator is mainly expressed by its robustness, achieved by programming flexibility. In addition, it possesses a high level of interactivity and adaptability. Models can be changed at any time, and the status of elements can be inspected. WITNESS enables a model to be saved with its current status, and it is case insensitive. There is no limit to model size apart from hardware limits. Partially developed models can be retrieved and edited.

Modelling assistance is provided by several features. Prompting is provided, but it is biased towards experienced users because it mainly points at what should not be done. Code entered via the text editor is automatically formatted, and the software imposes its own use of upper and lower case letters. An easily accessible on-line help is provided, but the information it gives is somewhat general.

Several useful features that facilitate testability are provided. Error messages are supplied. It is possible to obtain a graphical display of the values of functions and variables in addition to animation. When experiments are run in the step mode, every change in model status that happens is written in the interact box. It is also possible to obtain trace files, with all the model changes that occurred during the simulation. The Explode function provides information about the status of model elements, listing all attributes of the parts positioned at these elements. Illegal inputs are rejected, with an appropriate message.

Software compatibility enables integration with spreadsheet packages for output data analysis, and integration with word processors to edit model list files, create input data files or create programs using the WITNESS Command Language.

With regard to the Input/Output group of criteria, a variety of reports are automatically provided as well as special user defined reports. Periodic reports written to a file can be also obtained. Dynamic graphical display of histograms and time series is also provided. Data can be entered into the model via a menu driven interface, or they can be read directly from the files.

Experimentation facilities provide automatic batch running of experiments. Speed adjustment is possible as well as the specification of a warm-up period for experimentation. Models can be re-started from a non-empty state.

The quality of statistical facilities is good in the sense that a variety of theoretical statistical distributions are provided as well as 100 different random number streams. User defined distributions can be specified. It is possible to perform antithetic sampling.

A high level of user support is provided by the supplier. A help-line is available to users, training courses are organized, and user group meetings are held regularly. Documentation and reference cards are supplied, but the quality of documentation could be improved.

Positive aspects of financial and technical features are software portability, its availability for standard hardware and for standard operating systems, educational discounts given to universities, and relatively frequent updating of the software.

With regard to the pedigree of WITNESS, it is claimed that it is widely used, especially in industry. It was introduced in 1986. References describing characteristics of this simulator and its successful use in simulation projects are available. It was developed from the general purpose language SEE WHY.

Many general manufacturing modelling features are supplied such as part attributes modelling, shift modelling, capacities, breakdowns modelling, machine setup modelling, rejects modelling and job lists. Parts can arrive in the model in batches. In addition, it is possible to model buffer delays and a variety of operations such as assembling, disassembling, inspection and fluid composition.

Typical physical elements existing in manufacturing systems are pre-defined and incorporated in the simulator. Different types of machines can be explicitly modelled such as single, batch, production, assembly, multi-cycle and multi-station machines. Buffers, labour, conveyors, trucks and vehicles, and continuous processing elements such as tanks and fluids are also provided.

Scheduling features are mostly supported by the programming flexibility of WITNESS. Conditional routing is possible, and a variety of input and output rules are available. Various scheduling strategies can be modelled by programming with the support of input/output rules. Different priorities can be specified for different elements and the preemption of labour can be performed. Vehicle scheduling can also be modelled.

A variety of reports regarding manufacturing performance can be obtained such as information on throughput, work in progress, the utilization of production equipment and the scrap level of the parts. In addition, special user-defined reports can be created.

5.2 Weaknesses

The main shortcoming of WITNESS regarding its general features are that, because of its comprehensiveness, it is not easy to learn so that its full potential may be realised, and its special logical features modelled. In addition, it is not possible to create run-time applications.

With regard to visual aspects, the icon library supplied is quite small and the icons are too simple. The graphical display of the models is overwritten by windows representing, for example, an interaction box. It is not possible to obtain three-dimensional graphical displays of models.

The main weaknesses of the coding aspects are the limited flexibility of the language provided for additional coding, and the restrictions on its use. For example, it is not possible to program actions when a part arrives at a machine. This is possible only when the machine starts operating. Another shortcoming relates to the text editor provided for coding. The maximum length of lines in the editor is 256 characters, which may cause problems when complex features are modelled. In addition, there

is no indication of the cursor position within the line, so it is not possible to know when the limit of 256 characters has been reached. Going over the limit is reported only when the code is to be saved. Saving is then not possible, nor is it possible to determine which parts of the lines are surplus.

Efficiency is restricted by the problems with reliability. Namely, the program might get stuck for no apparent reason, and then the computer has to be rebooted. Multitasking and model chaining are not provided. There is no automatic saving of models nor the possibility to exit to the operating system within the software. Merging of models is not possible, which is especially inconvenient when large complex models are developed.

Weakness of the modelling assistance lie in the limited usefulness of prompting and on-line help, which is to general.

Testability is generally good, but it might be useful if the quality of error messages is improved, because they do not provide advice on how the detected error can be corrected. In addition, a backward clock is not provided and it is not possible to view the workflow path of the parts.

With regard to software compatibility, at the moment it is not possible to integrate WITNESS with CAD systems, statistical packages, data base management systems, expert systems, MRP II software and scheduling software.

The shortcomings of the input/output features relate to a lack of static graphical displays of simulation results. In addition, there is no automatic rescaling of the y axis in dynamic graphical displays of time series and histograms, and the standard output report written to a file is lengthy and not comprehendible. It is not possible to obtain a summary report of multiple independent experiments.

The main weakness of the experimentation facilities is the absence of an experimental design capability and no facility to interrupt experiments run automatically. Setting up an automatic run of experiments is not straightforward.

The main limitation of the statistical facilities is the lack of an output data analysis facility. There is a fixed number of random number streams, and the user cannot specify stream seeds. Confidence intervals cannot be obtained, and a facility for distribution fitting is not provided.

The main shortcomings of user support relate to the lack of an interactive tutorial which can facilitate learning of the package, and the quality of documentation. Documentation should provide more useful examples of the functions, actions and input/output rules and it should include an explanation of error messages.

With regard to financial and technical features, the main obstacle is the high price of the package, and substantial hardware requirements (it requires a minimum of 4MB of memory to operate, and a recommended 8MB of hard disk to install). In addition, a security device is obligatory, which is not very convenient, especially if the software is used for education.

Considering the general manufacturing modelling features, it is apparent that an automatic increasing of the buffer capacity is not provided. The explicit modelling of some specific operations such as fixturing and palletization is not straightforward, whilst fluid modelling is quite basic.

Although the major physical elements typical for manufacturing systems are provided, some special ones are missing such as pallets with fixtures, pallet shuttles, containers, robots and cranes. Some of those elements that are provided, such as vehicles, are not easy and straightforward to use.

The main limitations of the scheduling features are an inability to push/pull a part from specific positions within the element, to push/pull from the element more than one part, and routing restrictions. For example, buffers are passive, which means that they can neither push not pull parts. In addition, there is no departure scheduling for the shipping area, and there is no explicit way of using the batch index. Automatic calculation of optimal scheduling is not provided.

A variety of measures of manufacturing performance are provided by the software, or could be obtained with additional programming. Nevertheless, there is no schedule related report such as a Gantt chart, and it is not possible to obtain a production sequence summary report.

6 Summary

The information presented in this paper relates to a critical evaluation of the data driven manufacturing simulator "WITNESS" performed on the basis of a case study model. This evaluation

revealed that when a complex real system is modelled using this type of simulation software, it is likely that a number of approximations must be made due to limitations of the software.

However, despite all the problems experienced and the deficiencies of the software discovered, the majority of the important features of the system under consideration were in the end successfully modelled. Those that were omitted were of no significant importance to the final results.

7 Conclusions

This research shows that despite the constant development of software packages for manufacturing simulation, and especially manufacturing simulators, these simulators still reflect that their development is on the basis of a limited number of case studies. Even if these software tools can model all the desirable features of systems under consideration, the modelling time is long, and various difficulties are to be expected.

The simulation package "WITNESS" was suitable for this study, despite all the problems perceived. This does not necessarily mean that this or any other simulation software of this type would be equally successful for other systems, or that the problems experienced would be the same. Every system is unique, and so are the logical features that have to be modelled, which imposes substantial demands upon simulation packages, and especially upon data driven simulators.

Despite regular revisions and improvements to the majority of simulation packages, and the addition of better new ones, the development of new and more sophisticated and flexible simulation software tools is needed. In the meantime, for any study, an estimation has to be undertaken of whether the approximations that have to be made due to the limitations of the simulator are acceptable, or whether bespoke programming using a simulation language is preferable.

References

[1] Banks J., Aviles E., McLaughlin J.R. and Yuan R.C.: The Simulator: New Member of the Simulation Family, Interfaces, 21:2, March- April 1991, 76-86.

[2] Belanger R.F., Donovan B., Morse K.L., Rice V and Rockower D.B.: MODSIM II Reference Manual, CACI Products Company, La Jolla, California, 1989.

[3] Carrie A.: Simulation of Manufacturing Systems, Wiley, Chichester, 1988.

[4] Clementson A.T.: The ECSL Plus System Manual, available from Clementson A.T., The Chestnuts, Princess Road, Windermere, Cumbria, UK, 1991.

[5] Conway R., Maxwell W., McClain J. and Worona S.: User's Guide to XCELL+, The Scientific Press, San Francisco, 1988.

[6] Ekere N.N. and Hannam R.G.: An evaluation of approaches to modelling and simulating manufacturing systems, International Journal of Production Research, 27:4, 1989, 599-611.

[7] Hills P.R.: An Introduction to Simulation Using SIMULA, NCC Publications 5-Ss, Norwegian Computing Centre, Oslo, Norway, 1974.

[8] Hlupic V. and Paul R.J.: Software Packages for Manufacturing Simulation: A Comparative Study, in the Proceedings of the 14th International Conference Information Technology Interfaces, (15-18 September 1992, Pula, Croatia), University of Zagreb Computer Centre, Zagreb, 1992, 387-391.

[9] Hlupic V. and Paul R.J.: Simulation Modelling of an Automated System for Electrostatic Powder Coating, in the Proceedings of Winter Simulation Conference WSC'93, Los Angeles, December, 1993.

[10] Law A.M. and Haider S.W.: Selecting Simulation Software for Manufacturing Applications:Practical Guidelines & Software Survey, Industrial Engineering, 31, May 1989, 33-46.

[11] Law A.M. and Kelton W.D.: Simulation Modelling & Analysis, 2nd Edition, McGraw-Hill, Singapore, 1991.

[12] Pedgen C.D., Sadowski R.P. and Shannon R.D.: Introduction to Simulation Using SIMAN, Systems Modelling Corporation, Sewickley, Pa., 1990.

[13] Pidd M.: Computer Simulation in Management Science, 3rd Edition, Wiley, Chichester, 1992.

[14] Pritsker A.A.B.: Introduction to Simulation and SLAM II, 3rd ed., Halsted, New York, 1986.

[15] Russell E.C.: Building Simulation Models with SIMSCRIPT II.5, CACI Products Company, La Jolla, California, 1983.

[16] Schriber T.J.: An Introduction to Simulation Using GPSS/H, John Wiley, New York, 1990.

[17] SIMFACTORY II.5 Reference Manual and User Guide, CACI Products Company, La Jolla, California, 1992.

[18] Thompson M.B.: AutoMod II: The system builder, in the Proceedings of the 1989 Winter Simulation Conference, (December 4-6 1989), eds. MacNair E.A., Musselman K.J., and Hidelberger P, Washington DC, 1989, 235- 242.

[19] Van Breedam A., Van de Raes J. and Velde K.: Segmenting the simulation software market, OR Insight, 3:2, April-June 1990, 9-13.

[20] White D.A.: PCModel User's Guide, Simulation Software Systems, San Jose, California, 1988.

[21] WITNESS User Manual, AT&T ISTEL Ltd., Redditch, UK, 1991.

Compressed Transmission Mode: An Optimizing Decision Tool

Tarek M. Sobh
Department of Computer Science, University of Utah
Salt Lake City, Utah 84112, U.S.A.
E-mail: sobh@cs.utah.edu
AND
Tarek K. Alameldin
Computer Visualization Laboratory, Graduate School of Architecture
Texas A & M University, College Station, Texas, U.S.A.
E-mail: tarek@viz.tamu.edu

Keywords: Communication, Compression, Optimization, Tools

Edited by: Rudi Murn
Received: March 1, 1994 **Revised:** June 14, 1994 **Accepted:** September 20, 1994

In this paper we address the problem of host to host communication. In particular, we discuss the issue of efficient and adaptive transmission mechanisms over possible physical links. We develop a tool for making decisions regarding the flow of control sequences and data from and to a host. The issue of compression is discussed in details, a decision box and an optimizing tool for finding the appropriate thresholds for a decision are developed. Physical parameters like the data rate, bandwidth of the communication medium, distance between the hosts, baud rate, levels of discretization, signal to noise ratio and propagation speed of the signal are taken into consideration while developing our decision system. Theoretical analysis is performed to develop mathematical models for the optimization algorithm. Simulation models are also developed for testing both the optimization and the decision tool box.

1 Introduction

Data which is transmitted over a communication medium between two computers contains some form of redundancy. This redundancy can be exploited to make economical use of the storage media or to reduce the cost of transferring the data and commands over the communication network. One of the basic issues in the design of the presentation layer is to decide whether data is to be compressed before transmission or not. Many factors may affect making this decision, one of the most important ones is the cost factor. Compressing data before sending it will help reduce cost.

Some other factors may affect the decision of compression. The time factor may be the influencing one, in fact, one should not forget that there is the overhead of the compression and decompression algorithms at the sender and at the receiver hosts. This overhead is both in time and money, as the CPU is used for running the algorithms. Thus, the designer always faces the problem of when should one compress the data. The parameters which may affect this decision might be the parameters of the physical communication medium, they might also be the parameters of the compression algorithm used or both.

The decision that the design engineer will have to make might be a decision to compress or not given a certain fixed compression and physical medium parameters, or it might be a decision to compress depending on the value of one or more of the parameters (i.e., to compress if a certain threshold is met). In our work, we try to develop a tool for making such a decision, we choose the time to be the criteria for making compression decision, where the time is both for the compression overhead and for transmission. We develop a yes/no decision box given some fixed parameters and an optimizing decision box for finding the

threshold for one varying parameter while fixing the others. Theoretical analysis is performed and also simulations for different sets of data and a decision (or a threshold) is output for each kind of analysis.

2 Physical and Compression Parameters

The parameters of the communication medium between two hosts will affect the decision regarding compression. Whether the medium be a coaxial cable, a fiber optic cable or air (in the case of radio transmissions) it can always be completely specified by parameterizing it. The parameters that may help in determining the transmission time can be listed as follows :

- The data rate in bits per second. $\{D$ bps$\}$
- The band width of the medium in hertz. $\{B$ Hz$\}$
- The distance between the two hosts under consideration in meters. $\{L$ m$\}$
- The levels of discretization. $\{l\}$
- The baud rate in changes per second. $\{b\}$
- The signal to noise ratio in decibels. $\{S$ dB$\}$
- The propagation speed of the signal in the medium, in meters per second. $\{P$ m/s$\}$

It should be noticed that there is redundancy in expressing the time for transmission using all those parameters and the number of bits sent. For example, it is sufficient to use the number of bits and the data rate to express the time. However, if the data rate is not available we can use the baud rate, the levels of discretization and the data size, or alternatively we can use Shanon's maximum data rate bound, thus using the band width, the signal to noise ratio and the data size to find an expression for the minimum time for transmission.

The other set of parameters that is involved with the computation of the time for transmitting a certain amount of data is the set of the compression algorithm parameters. The CPU run time as a function of the size of data input to the algorithm is one of those parameters. The expected compression ratio, which actually depends on what *type* of data to be transmitted is the second compression parameter of concern.

3 Mathematical Formulation

The problem can be mathematically formulated by trying to find the cost of sending a certain number of bits from a host to another. The cost will be assumed to be the time through which the channel will be kept busy sending the data plus the time that will take the CPU to perform the compression and decompression on the data that are required to be transmitted. One can use a weight in the cost expression to denote that, for example, the cost for utilizing the network cable for one second is X times the cost for utilizing the CPU for one second. Thus, the expression for the cost function may be written as :

$$\text{Transmission time} + X \times \text{CPU computation time}$$

where X is the ratio between the cost of using the network for one unit time and the cost of one unit CPU time.

3.1 The Transmission Time

If we make the assumption that we only have two hosts connected directly and ignore the other overheads of the protocol to be used, the time needed to transmit N bits from a host to another can be written as a mathematical expression in terms of the physical medium parameters. For our implementation we are going to develop the transmission expression in four different ways, using four different sets of physical parameters, where each set individually is sufficient to specify the transmission time $T1$ completely.

3.1.1 Formulation Using the Data rate

The time required for transmitting N bits can be formulated as follows :

$$T1 = \frac{N}{D} + \frac{L}{P}$$

where D is the data rate in bits per second, L is the distance between the two hosts and P is the signal propagation speed. The first term is for the emission of N bits from the sender and the second term is the time for the last bit to reach the receiver.

3.1.2 Formulation Using the baud rate

The time required for transmitting N bits can be formulated as follows :

$$T1 = \frac{N}{b \log_2 l} + \frac{L}{P}$$

where b is the baud rate in changes per second, l is the number of levels of discretization, L is the distance between the two hosts and P is the signal propagation speed. The first term is for the emission of N bits from the sender and the second term is the time for the last bit to reach the receiver.

3.1.3 Formulation Using the band width

The time required for transmitting N bits can be formulated as follows :

$$T1 = \frac{N}{2B \log_2 l} + \frac{L}{P}$$

where B is the band width in Hertz, l is the number of levels of discretization, L is the distance between the two hosts and P is the signal propagation speed. The first term is for the emission of N bits from the sender and the second term is the time for the last bit to reach the receiver. In this case, there is assumed to be no noise whatsoever, we are assuming the maximum possible data rate.

3.1.4 Formulation Using the Signal to Noise Ratio

The time required for transmitting N bits can be formulated as follows :

$$T1 = \frac{N}{B \log_2(1 + 10^{\frac{S}{10}})} + \frac{L}{P}$$

where B is the band width in Hertz, S is the signal to noise ratio in decibels, L is the distance between the two hosts and P is the signal propagation speed. The first term is for the emission of N bits from the sender and the second term is the time for the last bit to reach the receiver. In this case Shanon's maximum data rate of a noisy channel is assumed.

3.2 The Compression and Decompression Times

The run times of the algorithm for compression and decompression can be expressed as a function of the size of the input in terms of machine cycles. That is, the compression time can be expressed as **T2**(M) where M is the size of data that is input to the compression algorithm.

3.3 Total Cost Without Using Compression

The total time to send N bits without using compression would then simply be equal to the transmission time, thus it equals one of the four expressions discussed previously. The total cost is considered to be only the time during which the communication channel is utilized.

3.4 Total Cost Using Compression

The total cost for transmitting a sequence of bits using compression will be assumed to be a weighted combination of the times for transmission and the times for compression and decompression. Thus, if we assume the compression ratio of the algorithm to be equal to **R**, and X is the ratio between the cost of using the network for one unit time and the cost of one unit CPU time and if we also assume a variable page size, i,e. compression is to be performed before each transmission of a block of size M of data, the total cost to be incurred (when we express the transmission time in terms of the data rate) can be written as :

$$C = \frac{\frac{1}{\mathbf{R}}M}{D} + \frac{L}{P} + X(\mathbf{f1}(M) + \mathbf{f2}(\frac{1}{\mathbf{R}}M))$$

where **f1** and **f2** are the compression and decompression runtime functions (in terms of the input size).

Similarly, the total cost can be written for the other physical medium sets of parameters as :

$$C = \frac{\frac{1}{\mathbf{R}}M}{b \log_2 l} + \frac{L}{P} + X(\mathbf{f1}(M) + \mathbf{f2}(\frac{1}{\mathbf{R}}M))$$

or

$$C = \frac{\frac{1}{\mathbf{R}}M}{2B \log_2 l} + \frac{L}{P} + X(\mathbf{f1}(M) + \mathbf{f2}(\frac{1}{\mathbf{R}}M))$$

or

$$C = \frac{\frac{1}{\mathbf{R}}M}{B\log_2(1+10^{\frac{S}{10}})} + \frac{L}{P} + X(\mathbf{f1}(M) + \mathbf{f2}(\frac{1}{\mathbf{R}}M))$$

4 Compression Algorithms

The methods to compress data have been studied for many years. However, several problems have prevented the widespread integration of compression methods into computer systems for automatic data compression. These problems include poor runtime execution preventing high data rates, lack of flexibility in the compression procedures to deal with most types of redundancies and storage management problems in dealing with storage of blocks of data of unpredictable length. In most cases a method presents some subset of these problems and therefore is restricted to applications where its inherent weaknesses do not result in critical inefficiencies. In this section we shall review the different forms of redundancies that can be taken advantage of for compression and then look at some of the approaches taken towards this end. Then we shall present a method due to Welch [3] which avoids many of the drawbacks of most of the methods.

4.1 Kinds of Redundancies

There are four major types of redundancies that we shall mention here. The forms of redundancies discussed are not independent of each other but overlap to some extent. There are some other forms of redundancies too, but the ones we are going to discuss are the major ones.

In different types of data, some of the characters are used more frequently than others. For example, in English text we see space and 'e' more frequent than any other character and special characters are used a lot less frequently. Generally speaking, all of the string combinations might not be used in a specific data set, resulting in the possibility of reducing the number of bits per combination. This kind of redundancy is due to character distribution.

The repetition of string patterns is another form of redundancy found in some of the cases. For example, the sequence of blank spaces is very common in some kind of data files. In such cases the message can usually be encoded more compactly rather than by repeating the string pattern.

In a certain type of data set, certain sequences of characters might appear very frequently. Some pairs may be used with higher frequency than the individual probabilities of the letters in these pairs would imply. Therefore, these pairs could be encoded using fewer bits than the combined combinations of the two characters formed by joining together the individual combinations for the two characters. Likewise, the unusual pairs, can be encoded using very long bit patterns to achieve better utilization.In some data sets, certain strings or numbers consistently appear at a predictable position. This is called Positional redundancy. It is a form of partial redundancy that can be exploited in encoding.

4.2 Methods of Compression

Using the discussion on redundancy types as our basis, we shall discuss several practical compression methods, and then choose one of them and use it for our simulation.

4.2.1 Huffman Encoding

This is the most popular compression method. It translates the fixed-size pieces of input data into variable-length symbols. The standard Huffman encoding procedure prescribes a way to assign codes to input symbols such that each code length in bits is approximately $\log_2(SymbolProbability)$. Where symbol probability is the relative frequency of occurrence of a given symbol (expressed as a probability). Huffman encoding has certain problems. The first problem is that the size of input symbols is restricted by the size of the translation table. If a symbol is one eight-bit byte, then a table of 256 entries is sufficient. However, such a table limits the degree of compression that can be achieved. If the size of the input symbols is increased to two bytes each, the compression degree would be increased. However, such encoding would require a table of 64K entries which may be a high cost.

The second problem with Huffman encoding is the complexity of the decompression process. The translation table is essentially a binary tree, in that, the interpretation of each code proceeds bit by bit and a translation subtable is chosen depending on whether the bit is zero or one. This basically means a logic decision on every bit which can create a system bottle neck.

The third problem with Huffman encoding is the need to know the frequency distribution of characters in the input data which is not well known for all kinds of data. A common solution is to do two passes on the data, one to find the frequency distribution ordering and the other is to to do the encoding. This process may be done block wise to adapt to the changes in data. This is not very efficient although it might be acceptable.

4.2.2 Run-length encoding

Repeated sequences of identical characters can be encoded as a count field along with the repeated character. However, the count fields have to be distinguished from the normal character fields which might have the same bit pattern as the count fields. A possible solution is to use a special character to mark the count field. This might be suitable for ASCII text, but not when there are arbitrary bit patterns such as in the case of binary integers. Typically, three characters are required to mark a sequence of an identical character. So, this will not be useful for sequences of length three or less.

4.2.3 Programmed Compression

In formatted data files, several techniques are used to do compression. Unused blank or zero spaces are eliminated by making fields variable in length and using an index structure with pointers to each field position. Predicted fields are compactly encoded by a code table. Programmed compression cannot effectively handle character distribution redundancy and is therefore a nice complement of Huffman encoding. The programmed compression has several drawbacks. First it is usually done by the application programmers adding to the software development cost. The type of decompression used requires a knowledge of the record structure and the code tables. The choice of field sizes and code tables may complicate or inhibit later changes to the data structure making the software more expensive to maintain.

4.2.4 Adaptive Compression

The Lempel-Ziv [4,5] and related methods fall into this category. Fixed length codes are used for variable-length strings such that the probability of occurrence of all selected strings is almost equal. This implies that the strings comprising of more frequently occurring symbols will contain more symbols than those strings having more of the infrequent symbols. This type of algorithm exploits character frequency redundancy, character repetitions, and high-usage pattern redundancy although it is usually not effective on positional redundancy. The algorithm is adaptive in the sense that it starts with an empty translation table and builds the table as the compression proceeds. This type of algorithm is a one pass procedure and usually requires no prior information of the type of data. Such algorithm, gives poor compression results in the initial part of the data set; as a result the data set should be long enough for the procedure to establish enough symbol frequency experience to achieve a good compression degree over the whole data set. On the other hand, most finite implementations of an adaptive algorithm loose the ability to adapt after certain length of the input which means that if the input's redundancy characteristics vary over its length, the compression degree declines if the input length significantly exceeds the adaptive range of the compression implementation.

We have chosen a variation of the Lempel-Ziv procedure which is called LZW due to Welch [3]. This method retains the adaptive characteristics of the Lempel-Ziv procedure but is distinguished by its very simple logic which yields relatively inexpensive implementations and a potential of very fast execution.

4.3 The LZW Algorithm

The LZW algorithm is organized around a translation table, referred to here as a string table, that maps strings of input characters into fixed length codes. The use of 12-bit codes is common.

The LZW table has the prefix property that if wK is a string in the table and w is a string and K is a character, then w is also in the table. K is called the extension character on the prefix string w. The string table may be initialized to contain all single-character strings.

The LZW table, at any time in the compression process, contains strings that have been encountered previously in the message being compressed. In other words, it contains the running sample of strings in the message, so the available strings reflect the statistics of the message. The strings added to the table are determined by this parsing: each parsed input string extended by its next input character forms a new string added to the string table. Each such string is assigned a unique identifier, namely its code value.

The algorithm is quite simple and can have a very fast implementation. The main concern in the implementation is storing the string table which can be very large. However, it can be made tractable by representing each string by its prefix identifier and extension character. This will give a table of fixed length entries.

The decompressor logically uses the same string table as the compressor and similarly constructs it as the message is translated. Each received code value is translated by way of the string table into a prefix string and extension character. The extension character is pulled off and the prefix string is decomposed into its prefix and extension. This operation is recursive until the prefix string is a single character, which completes decompression of that code. Each update to the string table is made for each code received (except the first one). When a code has been translated , its final character is used as the extension character, combined with the prior string, to add a new string to the string table. This new string is assigned a unique code value, which is the same code that the compression assigned to that string. In this way, the decompressor incrementally reconstructs the same string table that the compressor used.

5 Comparing the Models

The goal of our mathematical formulations and modeling is to perform one of two basic tasks, the first one is to decide whether to use compression or not given a certain set of fixed parameters { for compression, decompression and the physical medium } , the other is to decide the threshold for a specific varying parameter before which we should perform compression and after which we should not perform compression.

Two independent situations can arise in our formulation, in the first one, we can consider the protocol in which the communication to take place is a one of varying page length. In this case, the compression is performed for one "chunk" of data at a time and is immediately sent after that. In the other case, the protocol may have a fixed page size and thus the compression is performed for large files and the compressed data is sent one page at a time. Thus comparing the two models for decision making and optimizing parameters can be performed for each one of these situations separately. It should also be noticed that there might exist hypothetical bounds and average values for the run times and compression ratios for the compression and decompression algorithms.

The way we construct our mathematical models and optimization problems depends entirely on the set of parameters described in section 2. We construct the timing models as a function in one parameter to be determined (in the case of solving an optimization problem), given all the values for the other communication medium and data parameters and solve the inequality to find the required range for the unknown parameter. In the case of decision problems, we solve the inequality given all the values of the different parameters and check to see if the inequality holds or not (the greater than or less than relation). The decision algorithm would provide an answer (whether to do compression or not) depending on the cost (time) it takes to compress with respect to the transmission time without doing compression.

5.1 Using a Varying-Length Page

The problem in this case is to either make a decision regarding compression or to optimize a parameter, the four different representations for the transmission time can each be used to formulate and express the total cost incurred in the compression and uncompression modes. In the decision problem, we choose the scheme to have the

less cost. In the optimization problems we find the range for a certain parameter such that, for example, compression is a better technique, by solving the inequality.

Assuming that we use the LZW algorithm, characters are 8 bits each, the machine's cycle rate is w cycles per second, the data size to be compressed is M bits and the compression ratio is **R**. The algorithm runtime can be formulated as :

$$\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}$$

The following inequality can be formed for the model using the data rate as the physical parameter, for cost of the compressed mode to be less than the cost of the uncompressed mode .

$$\frac{\frac{1}{\mathbf{R}}M}{D}+\frac{L}{P}+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{D}+\frac{L}{P}$$

For the model using the baud rate

$$\frac{\frac{1}{\mathbf{R}}M}{b\log_2 l}+\frac{L}{P}+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{b\log_2 l}+\frac{L}{P}$$

For the model using the band width

$$\frac{\frac{1}{\mathbf{R}}M}{2B\log_2 l}+\frac{L}{P}+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{2B\log_2 l}+\frac{L}{P}$$

For the model using the signal to noise ratio

$$\frac{\frac{1}{\mathbf{R}}M}{B\log_2(1+10^{\frac{S}{10}})}+\frac{L}{P} +X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{B\log_2(1+10^{\frac{S}{10}})}+\frac{L}{P}$$

5.2 Using a Fixed-Length Page

In this model we assume that the protocol has a fixed length page and that the compression and decompression is done for a large chunk of data M, in this situation another parameter should be taken into consideration, which is the page size m and the expression for the transmission time should now include the number of compressed pages that are sent over the communication medium. Thus the above inequalities can now be expressed as :

$$\frac{\frac{1}{\mathbf{R}}M}{m}\left(\frac{m}{D}+\frac{L}{P}\right)+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{m}\left(\frac{m}{D}+\frac{L}{P}\right)$$

For the model using the baud rate

$$\frac{\frac{1}{\mathbf{R}}M}{m}\left(\frac{m}{b\log_2 l}+\frac{L}{P}\right) +X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{m}\left(\frac{m}{b\log_2 l}+\frac{L}{P}\right)$$

For the model using the band width

$$\frac{\frac{1}{\mathbf{R}}M}{m}\left(\frac{m}{2B\log_2 l}+\frac{L}{P}\right)+X \left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{m}\left(\frac{m}{2B\log_2 l}+\frac{L}{P}\right)$$

For the model using the signal to noise ratio

$$\frac{\frac{1}{\mathbf{R}}M}{m}\left(\frac{m}{B\log_2(1+10^{\frac{S}{10}})}+\frac{L}{P}\right) +X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{m}\left(\frac{m}{B\log_2(1+10^{\frac{S}{10}})}+\frac{L}{P}\right)$$

6 The Experiment

In our experiment towards the goal of establishing a reasonable tool for the design engineer, we offer the choice for either making a decision to use a compression/decompression scheme given a certain situation, i.e, a fixed set of physical layer parameters and a certain size of a chunk of data, or choosing to optimize { obtain the threshold of} a certain parameter, such that we can use compression for all values of the parameter that are less than this threshold, as it gives a less total cost than the technique that does not use compression. The user is given the choice of choosing any one of the four different ways of modeling the cost function, to maximize the number of parameters that one can deal with. Thresholds are found by solving the above inequalities for the parameters that are to be optimized for minimizing the total communication cost.

The results of running the experiment are displayed both theoretically and realistically. In the theoretic solution, the input file to be transmitted is assumed to be a plain text, thus assuming no prior information about the kind of data that are transferred in the network, then, a more realistic { either a decision or a value } solution is given by calculating the actual compression and decompression runtimes by running the LZW algorithm on them and observing the times and the compression ratio.

In the first two examples the algorithm is run on image data. The first one contains a lot of details, the second is mainly a few number of dots in a planar surface, it is not surprising then to know that the compression ratio in the second example turned to be equal to 48 !!, especially when we remember the adaptive characteristics of the LZW algorithm. The compression ratio in the first one was equal to 5. This fact contributed to the difference in the thresholds and decisions between the "theoretic" and the "realistic" approaches to finding the required limits and decisions. The following are snap shots of three different runs for the system, the first two are for the complex image, the third is for the simple one :

```
>> project.e
Enter various input values
prompted for.

Decision or Optimization? [d/o]:o

Desired Model?
[1=using data rate, 2=baud rate,
 3=bandwidth, 4=using sig-noise
 ratio]:3

Data Size? 8389192
Cycle Rate? 14286000
Network/CPU time cost ratio? 5
Theoretical Compression Ratio? 1.8
Observed Compression Ratio? 5.156
Observed Compression Time? 1.3
Observed Decompression Time? 1.2
Levels of Discretization? 2

Theoretical Results: If band width
< 1588559.073359 Hz then compress.
Simulation Results:  If band width
< 270484.731978 Hz then compress.

>>
>> project.e
Enter various input values
prompted for.

Decision or Optimization? [d/o]:d

Desired Model?
[1=using data rate, 2=baud rate,
 3=bandwidth, 4=using sig-noise
 ratio]:4

Data Size? 8389192
Cycle Rate? 14286000
Network/CPU time cost ratio? 5
Theoretical Compression Ratio? 5
Observed Compression Ratio? 5.156
Observed Compression Time? 1.3
Observed Decompression Time? 1.2
Signal to Noise Ratio in decibels? 5
Medium Bandwidth? 1000000

Theoretical Results:
Compression would cost less.
Simulation Results:
Compression would cost more.
```

```
>>
>> project.e
Enter various input values
prompted for.

Decision or Optimization? [d/o]:o

Desired Model?
[1=using data rate, 2=baud rate,
 3=bandwidth, 4=using sig-noise
 ratio]:1

Data Size? 1966640
Cycle Rate? 14286000
Network/CPU time cost ratio? 5
Theoretical Compression Ratio? 1.8
Observed Compression Ratio? 48.468
Observed Compression Time? 0.4
Observed Decompression Time? 0.2

Theoretical Results: If data rate
< 3177118.146718  bps then compress.
Simulation Results:  If data rate
< 642021.316608  bps then compress.

>>
```

We then proceed in the experiment to try different kind of data, we try executable files and observe the results of running our toolbox. The following is a sample run for the system on a file of executable commands.

```
>> project.e
Enter various input values
prompted for.

Decision or Optimization? [d/o]:d

Desired Model?
[1=using data rate, 2=baud rate,
 3=bandwidth, 4=using sig-noise
 ratio]:4

Data Size? 745472
Cycle Rate? 14286000
Network/CPU time cost ratio? 5
Theoretical Compression Ratio? 1.8
Observed Compression Ratio? 1.526
Observed Compression Time? 0.5
Observed Decompression Time? 0.3
Signal to Noise Ratio
in decibels? 30
Medium Bandwidth? 3000

Theoretical Results:
Compression would cost more.
Simulation Results:
Compression would cost more.

>>
```

It is noticed that for a "nice" collection of information, which includes a lot of repetitiveness, the compression ratio is maximum, while it decreases for other types. The fact that there is sometimes a discrepancy between the realistic and the theoretic values is because the theoretic approach assumes a "perfect" media when it calculates the runtime for compression, however, this is not the case when performing the actual compression in software on a down-to-earth Vax workstation. Also the wide variations in the compression ratios should be taken into consideration.

7 Conclusions

We discussed the issue of efficient and adaptive transmission mechanisms over possible physical links between hosts. A tool was developed for making decisions regarding the flow of control sequences and data from and to a host. The decision of compressing data and commands is discussed in details, a yes/no decision box and an optimizing tool for finding the appropriate thresholds for a decision were implemented. Physical parameters are taken into consideration while developing our decision system. Also, the compression parameters relationships and different compression techniques suited for the task are developed, with an emphasis on adaptive ones that accommodate various data and control patterns. Theoretical analysis is performed to develop mathematical models for the optimization algorithm. Simulation models are also developed for testing both the optimization and the decision tool box. Our system is tested through a series of simulations and a comparison is performed against the theoretical results for some data and control sequences.

References

[1] V. Cappellini, *Data Compression and Error Control Techniques with Applications*, 1985.

[2] W.K. Pratt, *Image Transmission Techniques*, 1979.

[3] T.A. Welch, " A Technique for High Performance Data Compression", *IEEE Computer*, June 1984, pp. 8-19.

[4] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. Information Theory*, Vol. IT-23, No.3, May 1977, pp. 337-343.

[5] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding", *IEEE Trans. Information Theory*, Vol. IT-24, No.5, Sept. 1978, pp. 5306.

[6] H. K. Reghbati, "An Overview of Compression Techniques", *IEEE Computer*, Vol.14, No. 5, April 1981, pp. 71-76.

Adaptive Bar Implementation and Ergonomics

Matjaž Debevc, Rajko Svečko, Dali Đonlagić
Faculty of Technical Sciences
Smetanova 17, 62000 Maribor, Slovenia
E-mail: debevc@uni-mb.si

Beth Meyer
AT&T Human Interface Technology Center
500 Tech Parkway, N.W.
Atlanta, GA 30313, USA
E-mail: Beth.Meyer@AtlantaGa.ncr.com

Keywords: Adaptive User Interfaces, Intelligent User Interfaces, Human Factors, Icons Design

Edited by: Matjaž Gams
Received: March 9, 1994 **Revised:** June 20, 1994 **Accepted:** September 22, 1994

Self-adjusting, adaptive user interfaces offer automatic customisation of the computer-based working environment by checking users' procedures and typical decisions, eventually offering them adaptations or enhancements designed to make their individual work patterns easier and more efficient. This means that the users don't need deeper understanding of the application environment or its procedures, since the adaptive user interface itself recommends solutions and possible adjustments.
This article classifies user interfaces and their roles. Positive and negative aspects of adaptive user interfaces are also discussed. Using the adaptive bar as an example, we discuss the implementation and ergonomics of the adaptive bar, which represents the adaptive part of the interaction. During the working sessions, but without disturbing them, the user interface suggests the addition or removal of command icons and their resizing depending on the priority, which is based on the frequency of use. The article also offers a convenient solution to present the priority of icons.

1 Introduction

Self-adjusting (in following text called 'adaptive') systems offer automatic adaptation of the working environment by keeping track of user's procedures and decisions, and eventually offer the possibility of adjustment or easier ways to doing something (Geiser 1990). The user therefore doesn't need detailed knowledge about the working environment and possible procedures, since the adaptive system itself recommends best solutions and optimisation of repeating procedures. The same goes for adaptive user interfaces in human-computer interaction, since, for computer applications, the user interface is the user's working environment.

Graphical user interfaces, which represent an important component of current computer systems, are often the most sensitive elements of the communication. Advances in computer controls and displays make it possible for computers to be more widely accessible and lead to new, innovative interaction methods. One of these methods is interface adaptation, which could widen the group of users who would potentially switch to computerised processing or controlling. An ideal computer system would automatically adapt to the present user by identifying problem areas and offering help for the present work, therefore lowering the stress level and necessary concentration.

As a result, there are many research projects going on in this field (e.g., Browne *et al* 1990,

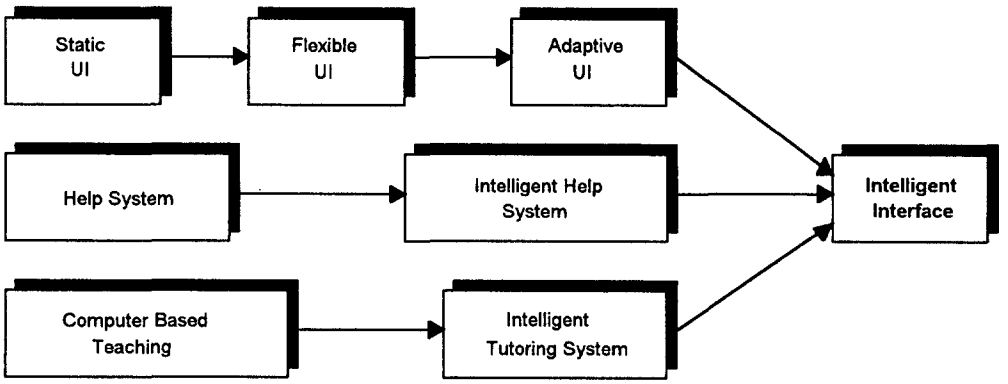


Figure 1: Adaptive user interface in relation to Intelligent Interface (in Kühme 1992)

Benyon and Murray 1988, Kossakowski 1989). The research is especially promising if we regard all the advantages of such a user interface in computer systems and applications, which are getting more and more complex. Studies such as those reported in Benyon and Murray 1988 and Kossakowski 1989 have demonstrated or implied significant human performance advantages for adaptive interfaces.

2 Classification

The user interface can be classified depending on the methods of adaptation. Kühme (Kühme *et al* 1992 and Hufschmidt *et al* 1993) suggests a scheme that represents multi-dimensional classification and is more convenient for representation of all existing viewpoints and prototypes of systems.

Figure 1 shows the scheme of classification of various existing user-interface concepts. The first generation of user interfaces was static; the system developer designed and implemented the user interface, and the user had to learn how to use it.

Later, flexible user interfaces began to appear. These user interfaces allow the user to make changes, although these changes are initiated and managed by the user. Today, many user interfaces are adaptable to at least some extent – for example, many offer the possibility of changing colours or resizing and moving the windows.

Adaptive user interfaces actively change according to conditions and user needs, either automatically or with user input. For example, an adaptive system might suggest and provide a special tool to perform a set of tasks that the user frequently performs together. If the functionality and demands of the application also adapt, the system can get quite complex.

According to Figure 1 the intelligent user interface is an assembly of adaptive user interfaces, intelligent help systems and intelligent learning tools.

Kühme has also defined **stages** and **agents** which influence the flow of the adaptation process. In every process of adaptation various tasks are executed. We can identify them as stages, which are carried out during the process of any single adaptation.

The **stages** are:	The **agents** are:
– Initiative	- System
– Proposal	- User
– Decision	
– Execution	

If the system is the agent that carries out all the stages we call it **Self-Adaptation**. In this case the system observes the communication, creates and evaluates various possible adaptations, and in the end selectss one of them and accomplishes it. Another type of adaptation is **User-Controlled Self-Adaptation**, in which the user determines whether the adaptation occurs and the system performs all of the other functions.

Systems that offer self-adaptation or user-controlled self-adaptation should also make it possible for the user to manually initiate the process

of adaptation. These variations are called **User-Initiated Self- Adaptation** and **Computer-Aided Adaptation**. In the first case the user initiates the adaptation process and the rest is carried out by the system, whereas in the second case the user both initiates the process and decides whether to accept the suggestion.

In the last two combinations the user performs all of the stages except, perhaps, initiation. Depending on the agent that performs the first stage we have **System-Initiated Adaptation**, in which the system initiates the process, and normal **Adaptation**, in which all stages are carried out by the user. The simple adaptation makes it possible for the user to arrange the system according to his preferences and goals. Almost every window manager offers, for instance, colour palette customisation and changing of window size and menu display.

Generally, an information system may function as an **adaptive user interface** if the system acts as an agent in any stage of adaptation. This definition is taken from the user's point of view, so it does not specify anything about implementation or necessary information (Haaks 1992, Zeidler 1992).

3 Negative and positive aspects of adaptive user interfaces

An adaptive user interface adapts itself according to the present user and the present common procedures. Because of continuous changing of working environment and user's tasks there are negative and positive aspects of adaptive user interfaces (Norcio and Stanley 1989).

a) negative aspects

The most inconvenient aspect of adaptivity is its potential to prevent the user from developing a clear model of a system, if the system is changing all the time. For example, the system could require the user to learn a new procedure to accomplish a task, just as the user was beginning to learn the initial procedure. This aspect could reduce the user's productivity and confidence in the system, since it could keep the user from understanding how the system will behave.

Another problem with adaptive user interfaces is that the user can lose the feeling of competence or of being in charge. It can even happen that the user's actual goals and demands are concealed by attempts to control the user interface behaviour. Therefore the user interface should make it possible for the user to exert control over the working environment. The user interface also must not take the initiative away from the user, but should give him the best and most proper assistance with the present task.

A typical negative effect is when the system suddenly changes and the user is forced to interrupt his work and determine what happened (Shneiderman 1992). The user may start worrying, because he is unable to predict what will the next adaptation look like, when it will happen and whether he will be able to restore the original state. This is why it would be better for the user if the general appearance would remain the same and only a part of it would change, and even that only on his request (Debevc 1993, Debevc *et al* 1993).

b) positive aspects

On the other hand we can also list some positive aspects of introduction of adaptive user interfaces. For example, automatisation of systems is a domain which expresses certain needs for adaptive interfaces. A system that dynamically changes its tasks must be able to adapt to individual users. The tasks should be assigned to the user as well as to the computer. The way in which such an assignment would take place depends on who has better overview upon the information and procedures at work with the automatisation system.

The user who has a great amount of information available at the same time has to make a decision and choose one of these pieces of information. But we know that the user is not always capable of making decisions equally well and quickly. In order to be able to assure general optimal performance of the system it is necessary for the computer to be able to link the user's previous decisions and eventually show him only a few pieces of information, for which the probability of being interesting for the user is the highest at the moment.

Use of adaptive user interfaces is especially convenient for the growing group of users that do not have the time to gain deeper understanding on either the computers generally or the particular application they are working with. The adaptivity is most convenient for novice users who

have to adapt to the new working environment as quickly as possible. Besides, the user is enabled to complete his task faster and more efficiently. Plus, the possibility of disappointment is lowered. For example, if an adaptive word processor user interface detects that a user is trying to print onto envelopes for the first time,, it could provide additional instructions for printing envelopes in unobtrusive help windows. These instructions could then be omittedafter the user has successfully printed onto envelopes several times.

The positive aspects of adaptive interfaces can be summarised in the goals of these interfaces. These goals are (Kühme 1992):

- easy, efficient, effective use
- making complex systems usable
- presentation of what the user wants to see
- faster use
- a user interface that fits heterogeneous user groups
- a user interface that considers increasing experience

The question exists, whether the positive aspects can outweigh the negative ones and to what extent are the adaptive user interfaces actually needed? If they are correctly and carefully designed, they can help in building more useful systems for treatment of larger amounts of information. In this way both beginners and experts can easily use the system. Adaptive interfaces help various types of users in making their work more efficient by showing them better ways to complete certain tasks and offer them the proper amount of support according to their individual needs.

4 Reasons for using the adaptive bar

One of the successful ways to adapt to the user can be found in the toolbar, which offers graphical representation of most often used commands and macros (Figure 2). The icons can be accessed quickly and simply; there is no need to spend time browsing through menus.

Figure 2: Toolbar in the Microsoft Word 2.0 for Windows^{TM}

The use of command icons is a relatively simple way to access various commands, but there are also some problems related to their use. The basic problem is in the size of the icons and poor overview of them, since we can barely distinguish between them on some screens because of their small sizes.

Although in some programs it is possible to adapt the bar to personal needs by adding or removing icons, changing the bar requires additional understanding of the structure of the program and the procedure for changing icons. Especially for the beginners, it can be a very difficult task to perform, since in many cases they cannot decide which commands will be used frequently and which they won't even need.

Especially for them and for others who often need a smaller number of repeating routine operations, we have designed the **adaptive bar**. It follows the frequency and manner in which the user performs individual commands, and, without interrupting the user's work, it suggests **addition** or **removal** of certain icons and shows their priority.

Perfect adaptivity is unfortunately impossible, but auto-adaptivity in our case means that the user interface tries to determine the priority of available commands. The bar is then adapted to the differing priorities, but the interface never acts on its own accord; it merely informs the user that an adaptation is possible. The user can then either confirm, deny or even ignore the suggestion. According to Kühme's classification, it is a **User-Controlled Self-Adaptation** system.

During the testing and empirical research (interviews, questionnaires) (Mayhew 1992) we determined that it is better for the end-user if the general appearance of the user interface remains static and only one part, which is simple and easy to use, is subject of change.

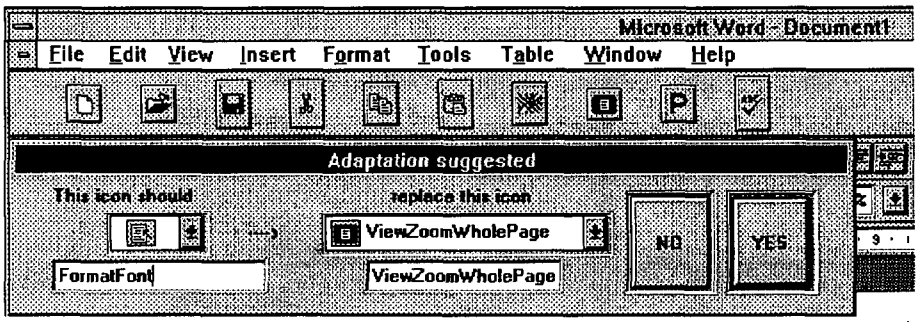


Figure 3: Adaptive bar with the dialog window

5 Adaptivity of the bar

The adaptive bar (Figure 3), which we have designed for Microsoft Word for Windows 2.0 (Microsoft 1991), has following characteristics:

- automatic addition and removal of icons
- automatic resizing of icons according to the frequency of use
- storing of present arrangement of the bar

The procedure of the **removal of a command icon** starts when the user interface, while measuring the frequency with which icons are used, determines that a particular icon hasn't been used for a pre-specified period of time. An indicator-icon, accompanied by a sound signal, appears and informs the user that a change to the bar is recommended. The **indicator of change** is represented by the background of the bar. The user eventually chooses the dialog window, which can be accessed by double-clicking on the background of the bar, and decides whether to accept the proposition. In this way we do not hinder the user's work dynamics. It is up to him to decide whether and when to change the bar. If the user ignores the proposition, the background remains in the warning state. This state is represented by the relief and intensive colour, distinguishable from the others. Further suggestions of changes, if not accepted, are stored and inserted in the waiting queue until the user decides about them.

For the user interface, the most complex and difficult operation is the **installation of command icons** in the bar. The system has to determine the frequency of commands and options from the menus and to decide which ones have to be inserted in the bar. A similar procedure is the determination of macro commands. In order to be able to determine the need for a change the user interface requires information about the application and its capabilities, the present user, previous events and the real-time system being controlled (Debevc *et al* 1992). On the basis of this knowledge it is able to check and compare user's actions. After a certain **frequency of repetition** it uses the indicator of change to inform the user that an adaptation would be suggested. After the adaptation is finished, which happens during the surveillance mode, the indicator of change returns into the passive state, which is represented by the flat appearance and lack of colour. Nevertheless, the bar-modification dialog box can still be accessed by double-clicking on the background of the bar.

The dialog window is designed to look simple and offer a clear overview of all its functions. The user can either place the suggested icon where adaptive bar recommends or he can instruct the bar to replace a different icon, simply by choosing a different destination icon in the dialog window.

The look of the bar can often change due to the adaptivity. Therefore the **storage of the present state** in a file has to be provided. In this way the adaptive bar can be used on the same system by various users with the possibility that everyone uses the bar developed during his own sessions. Of course, if there is a new user, the system opens the default type of bar, which can be either accepted or adapted to user's needs right away.

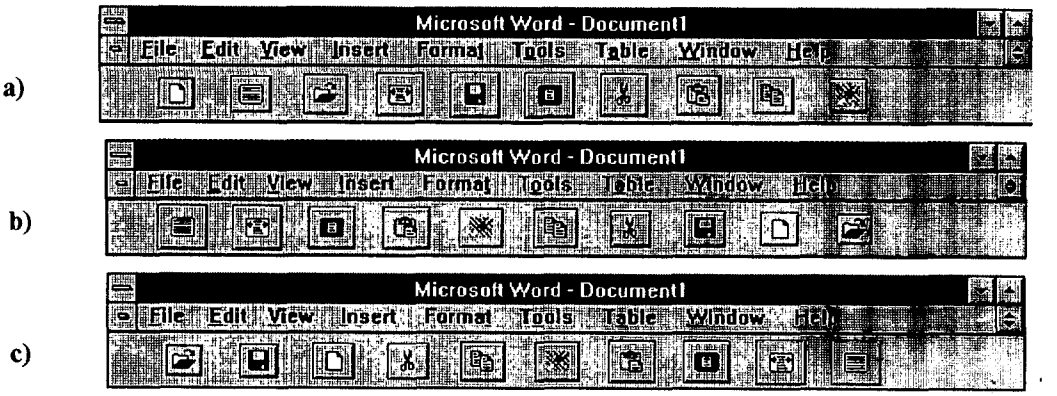


Figure 4: Models of the adaptive bar with varying positions of icons

6 Ergonomic point of view

In order to satisfy the ergonomic demands, which are described in ISO-standards (ISO-9241, 1992), we designed the bar in following two ways:

- Varying positions of icons:
- bigger icons in the centre of the bar, smaller ones on the sides
- bigger icons on the right side, smaller ones on the left side
- bigger icons on the left side, smaller ones on the right side
- Fixed positions of icons:
- varying widths and heights of icons
- varying widths of icons
- varying heights of icons

Even though we had to implement many different models of the bar, we used a single one and designed others quickly and simply by changing only four parameters:

- positions of icons: fixed or varying
- height range of icons (min. and max. height)
- width range of icons (min. and max. width)
- icon arrangement pattern

The testing sessions were followed by research using various prototype testing techniques (Mayhew 1992) (Structured observation, Benchmarking, Classic experiments). We also used an empirical research method with a questionnaire, written according to ISO-9241 Part 10 norms (Prümper 1993).

6.1 Varying position of icons

In the first test we introduced three models with varying positions of icons (Figure 4) to different kinds of users (novice users, students taking computer science classes, experienced users of interactive systems and experts). The first model has the most important icons placed in the centre of the bar and these are also the biggest ones, whereas other less important ones are getting smaller and are arranged towards both sides. The arrangement of these icons is very similar to the Gauss' random curve. The second model (Figure 4b) has the most important icons on the right hand side and the priority decreases from right to left side of the bar. The last model (Figure 4c) has the most important icons from left to right. In all three models the size of the icons decreases with their decreasing priority. The spacing between the middle points of icons are constant throughout the bar, which means that spacing between edges of bigger icons is smaller than the spacing between edges of smaller icons.

The results of the empirical research are shown in Figure 5. It is clearly visible from the table that toolbar "b" was the most convenient. The num-

bers represent the average number of points that the particular model gathered during the testing. The majority of subjects shared the opinion that this bar seems to be the most convenient because of its icon-arrangement from left to right, just the way we are used to read. In our findings, 24% of the subjects liked the changeable behaviour of the bar, while 72% of them preferred fixed positions of icons. Only 4% did not express a preference (Figure 6).

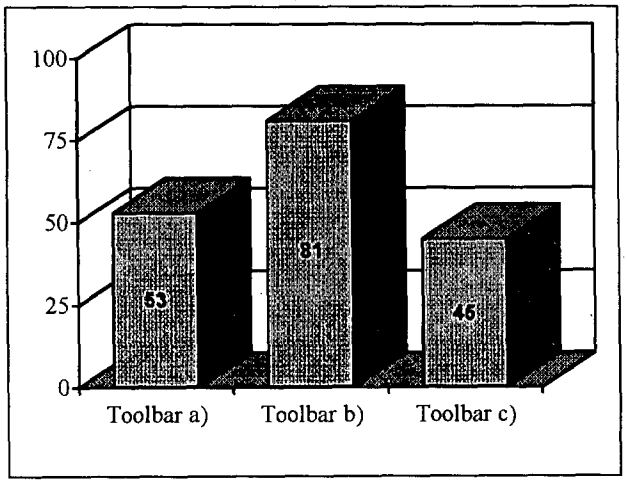


Figure 5: Results of the empirical research (collected points)

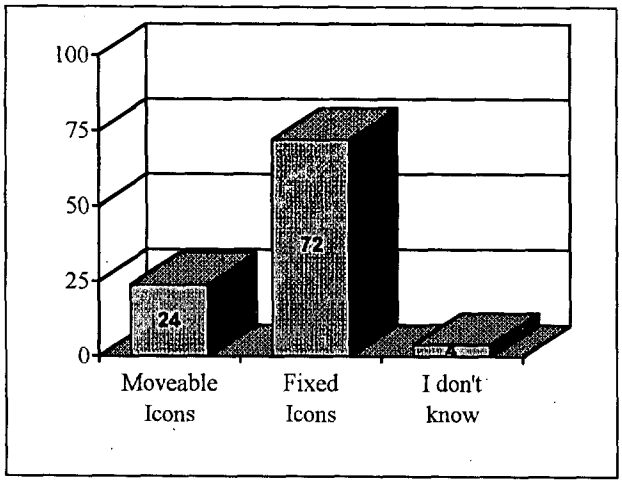


Figure 6: Results of the questionnaire about positions of icons (in percentages).

The results of the testing revealed this important fact: adapting the position of the icons was disliked by users and probably would not work well. It may be that users who are working with icons that change positions within the bar, even with their consent, would take more time to perform certain tasks and make more errors than the people working with icons in fixed positions. We make this prediction because such an adaptation requires the user to change his actions. It requires the user to move in a totally different direction to select an icon that he is already accustomed to finding elsewhere in the bar. This is a kind of adaptation that could cause problems for the user.

For example, if the user wants to select a certain icon that has just changed position, he would probably move the cursor to its former position, not to its current position. He then either selects the wrong icon, causing a time-consuming error, or he must make a second movement to get to the correct icon. Either way, the cost of moving the icon after it has been frequently used would be greater than the cost of leaving the icons unordered in terms of their priority. Whatever the order of the icons, it is likely that users will develop habits of moving quickly to the icons that they often use, perhaps without even looking at them.

Therefore the model of icons in fixed positions as long as they are in the bar is clearly better.

6.2 Representation of the priority of icons

The next step with testing was trying to find the best way to represent the priority of the icons, since it is not visible from their positions in the bar. The idea occurred to us, that it might be easier for the user to select his "favourite" icons if they are larger.

There are three possibilities:

- varying both width and height of icons
- varying width of icons
- varying height of icons

The most appropriate seems to be to vary the height of icons. The reasons for that are:

- When we change the width in any way, we must either change the position of icons as well, or we must have the spacing large enough in advance, which means potential waste of space.

– Increasing the height of an icon would probably make it easier to select it than changing its width.

However, these statements are based on the study of horizontal toolbar behaviour. For a vertical bar, the height is the dimension that should remain constant and width should be subject to change.

Returning to the horizontal bar, we can see that changing the widths of icons is impractical if we wish to keep them on the same positions. If we start with relatively small icons, and then increase their widths, we may have to move them as shown in Figure 7:

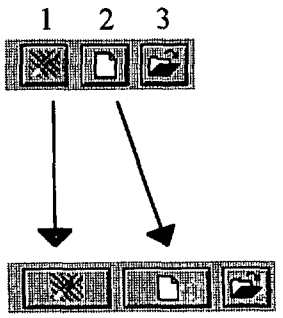


Figure 7: Changing widths and positions of icons after icons 1 and 2 are used frequently

It is clearly visible that icon 3 is almost hidden next to icons 1 and 2, which are much bigger, and is also moved far to the right.

Another possibility is to start with icons with very large spacing, in order to give them room to expand (Figure 8):

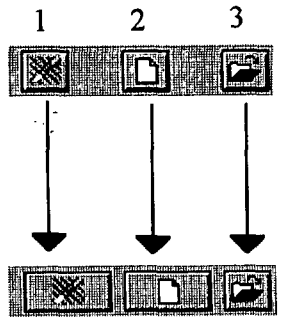


Figure 8: Changing widths of icons after icons 1 and 2 are used frequently

The later approach, while it does prevent the icons from changing their positions, greatly reduces the maximum number of icons in the bar. That is a problem, since a very useful feature of the adaptive toolbar is the possibility to add icons for frequently used commands.

This is the first reason for introduction of the third model. The second one is that the increase in height is likely to be more useful in making the icon easy to select than an increase in width. This prediction is based on Fitt's Law, the rule by which the difficulty of a hand movement can be predicted (Fitts 1954).

Fitt's Law states that the time required to make a movement is related to both the distance of the movement and the size of the target in the direction of the movement. The formula may be expressed in this way (from Bullinger, Kern, and Muntzinger 1987):

$\mathbf{MT = a + b^2 \log (2A/W)}$, where

MT is movement time,

A is "amplitude", or the distance from the starting to the centre point of the target,

W is the length of the target **in the direction of movement**(in the case of a circular target it would be twice its radius, not its area),

a and b are constants.

This means that, from the two movements below, which both start in the point 's' and finish in the centres of the targets, the one on the left is more difficult and takes more time to perform than the one on the right (Figure 9):

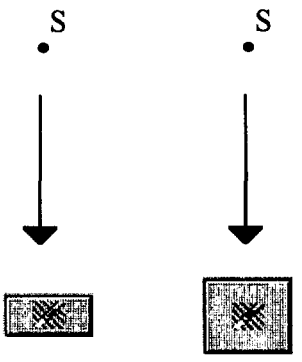


Figure 9: The right icon is easier to select that the left one

This is because the value of W for the target in the case on the right is larger, thus making the movement time (MT) smaller.

Furthermore, the two movements below should be equally difficult to perform (Figure 10):

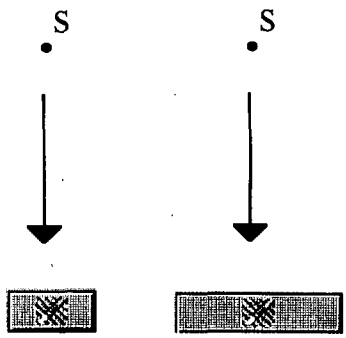


Figure 10: Equally difficult movements

This is because W, the length of the target in the direction of the movement, is the same for both cases. The extra width of the target on the right does not help when the movement starts directly above it. (It would help, however, if the movement started from the side of it.)

Someone moving the mouse cursor to an icon in the toolbar on the top of the application window, most probably starts moving it from a point on the screen well below the bar. In most cases, the movement will be primarily vertical. That means that increasing the height of an icon should have a greater effect on the ease of use than increasing the width of an icon.

One effect that increasing the width of an icon could have is that of reducing the distance that a user's hand must move to reach the icon. It could do this by decreasing the angle at which the user must move. However, this effect is unlikely to be significant. It is not practical to change the width of icons very much, for reasons given previously. Therefore, the vertical component of most movements is likely to be much greater than any possible change in icon width. For example, a movement from the center of the screen to a toolbar at the top of the screen might require 13 cm of vertical movement, while the maximum change in icon width might be about 1.3 cm. In a case where the center of an icon was 13 cm to the right or left of the center of the screen, and the user was moving to the icon from the center of the screen 13 cm below the toolbar, increasing the width of the icon by 1.3 cm would decrease the necessary movement distance by at most only 0,5 cm, or 2.5 percent. This figure is obtained by using the Pythagorean Theorem on two right triangles, one with sides of 13 cm (vertical distance) and 12.3 cm (horizontal distance), and the other with sides of 13 cm and 11.7 cm, respectively. This calculation compares two movements aimed at the very edge of the icon; if movements are consistently aimed at the center of the icon (the more probable case), the effect of a change in width is even smaller, since the center points of icons in fixed positions would always be at the same locations. Hence, it is reasonable to predict that increasing the width of an icon in a horizontal toolbar will not significantly increase its usability. However, this prediction should be tested in future experiments on the adaptive bar.

7 Conclusion

Generally speaking, adaptive user interfaces can be convenient for novice users, who have to learn quickly how to perform in an unfamiliar working environment. Since the user interface adapts itself to the user's knowledge and work, the user will soon lose the anxiety about the new environment and will start working on the actual problem more quickly. If the user interface is carefully and accurately planned it can be very useful for a wide group of users. These interfaces offer better efficiency of work by giving the proper amount of suggestions and help tailor information to different kinds of users and their individual preferences. Given the increasing complexity and functionality of computer systems, the benefit and the need for adaptable and adaptive user interfaces will also increase.

At testing of the adaptive bar, which represents the adaptive part of our user interface, the results have shown that the models which included changing positions of icons were unsuitable for the user's needs, since they were unable to remember and automatize movements for all the changes. The best solution appeared to be the model with fixed positions of the icons, in which priorities of the icons were represented by their height.

With help of the adaptive bar, we offer the user an opportunity to easily use the advantages of a self- adaptable user interface without being confused by changes, since the general appearance of the user interface remains unchanged. We also defined several principles for making the adapti-

bar ergonomically acceptable. These principles can be specifically tested in future studies of user performance with the adaptive bar.

References

[1] Benyon, D.R., Murray, d., 1988, Experience with Adaptive Interfaces, *The Computer Journal*, Vol. 31, No. 5, 465-473.

[2] browne, D.,Totterdell, P., Norman, M., 1990, Adaptive User Interface, *Academic Press*, London.

[3] Bullinger, H.-J., Kern, P., Muntzinger, W. F., 1987, Design of Controls, in *G. Salvendy (Ed.), Handbook of Human Factors, 577-600, John Wiley & Sons,*New York.

[4] Debevc, M., Donlagić, D., LEŠ, M., 1992, User Interface Styles, *Informatica*, Ljubljana, 16, 49-54.

[5] Debevc, M., 1993, Adaptive Bar, *INTERCHI '93, Conference on Human Factors in Computing Systems, INTERACT '93 and CHI '93, Adjunct Proceedings*, Amsterdam, 24-29 April, 117-118.

[6] Debevc, M., Donlagić D., SVEČKO R., 1993, Adaptive User Interfaces, *IEEE Electrotechnical and Computer Conference*, Portorož, Slovenija, September 27-29, 91-94.

[7] Fitts, P.M., 1954, The information capacity of the human motor system in controlling the amplitude of movement, *Journal of Experimental Psychology*, Vol. 47, No. 6, 381-391.

[8] Geiser, G., 1990, Mensch-Maschine-Kommunikation, *R. Oldenbourg Verlag*, München.

[9] Haaks, D. 1992, Anpaßbare Informationssysteme, *Verlag für Angewandte Psychologie,*Götinngen, Stuttgart.

[10] ISO-9241, 1992, Ergonomic requirements for office work with visual display terminals (VDTs), Part 10:*Dialogue principles - Second commitee draft.*

[11] Kossakowski, M., 1989, Adaptive Man-Computer Dialogues, in*F. Klix, N.A. Streitz, Y. Waern, and H. Wandke (Eds.), Man-Computer Interaction Research, MACINTER-II*, 201-212, *Elsevier Science Publishers*, Amsterdam.

[12] Kühme T., Dieterich, H., Malinowski, H., Schneider-Hufschmidt, M., 1992, Approaches to Adaptivity in User Interface technology: Survey and Taxonomy, *Proceedings of the IFIP WG2.7 Working Conference on Engineering for Human-Computer Interaction,*August 10-14, Ellivuori, Finland.

[13] Mayhew, D.J., 1992, Principles and Guidelines in Software User Interface Design, *Prentice Hall, Englewood Cliffs*, New Jersey.

[14] Microsoft Corporation, 1991, Microsoft Word for Windows 2.0, User's Guide.

[15] Norcio, A.F., Stanley, J., 1989, Adaptive Human-Computer Interfaces: A Literature Survey and Perspective, *IEEE Transaction on Systems, Man, and Cybernetics,*Vol. 19, No.2, 399-408.

[16] Prümper, J., 1993, Software Evaluation Based upon ISO 9241 Part 10, *Proceedings of the Vienna Conference Human Computer Interaction*, Austria, September 20-22, 255-265

[17] Schneider-Hufschmidt, M., Kühme, T., Malinowski, U., 1993, Adaptive User Interfaces: principles and practice, *Elsevier Science Publisher*, Amsterdam.

[18] Shneiderman, B., 1992, Designing the User Interface, Strategies for Effective Human-Computer Interaction, Second Edition, *Addison-Wesley Publishing Company*, New York.

[19] Zeidler, A., Zellner, R., 1992, Software-Ergonomie, Techniken der Dialoggestaltung,*R. Oldenbourg Verlag*, München.

Electrotechnical and Computer Conference ERK'94

Portorož, September 26–28, 1994

Electrotechnical and Computer Conference is becoming every year meeting of the professionals on the electrotechnical and computer field, under the protection of Slovenia IEEE Section. Once a year it joins interests of numerous professionals organizations. This organizations are paying attention to the conference programe on their professional areas and are organizing their meetings, round tables and invited lectures, where it is possible to discuss technical problems, technological professional and scientific development, international cooperations, educational programs, financial investments, employment and so on.

Also this year conference ERK'94 was mosaic of ten very specialized workshops, which were therefore on high quality level. Because of many different events on the same place it was oppurtunity again for a variety of interdisciplinary discussions and projects and to attend some of 8 available invited lectures interesting for a broader group of listeners.

All invited lectures and many other papers were in English language. So understanding was complete. Some papers were in Slovenian language, because we must develop Slovenian technical language all the time. Nobody else could do this for us. And such a forum, a kind of congress, is appropriate oppurtunity to do this. On the conference, there were 10 professionals areas with different number of sections but 31 all together. Proceedings have two parts.

In Part A we find Eelectronics in 2 Sections with 15 papers, Telecommunications in 2 Sections with 13 papers, Automatic Control in 5 Sections with 35 papers, Power Engineering in 5 Sections with 38 papers and Measurement technique in 4 Sections with 28 papers.

In Part B there is Computer and Information Science in 4 Sections with 29 papers, Artificial Intelegence in 2 Sections with 15 papers, Robotics in 2 Sections with 13 papers, Robotics in 2 Sections with 13 papers, Pattern Recognition in 3 Sections with 24 papers and Biomedical Engineering in 2 Sections with 15 papers. The interest for the Conference was more than expected and discussion after majority of presentations was very lively.

Let me tell you still about the Student paper contest. This action is going on with ERK Conferences from the very beginning. Paper must be much longer and applying directly to the contest. Later we always send the winning paper for the student paper contest in Region 8 of IEEE, which consists of Europe, Near East and Africa, the biggest area on the Globe among all 10 Regions.

I would like to invite you to participate on the next ERK'95, which will be held on the end of September 95 for which we need four page camera ready paper until July 24, 1995. Please plan your participation at the next Conference. For more information contact the contact editor of this magazine and we will send you Call for papers sometimes in February 95 when it will be ready.

Baldomir Zajc
Conference Chairman

Fourth International Conference on Information Systems Development - ISD'94

Bled, September 20 - 22, 1994

During the past few years, many new concepts and approaches emerged in the Information Systems Development (ISD) field. The various theories, methods and tools available to system developers also bring problems such as choosing the most effective approach for a specific task. The aim of this International Conference on Information Systems Development - ISD was to establish an international forum for the exchange of knowledge, experience and new ideas, and to share and stimulate new solutions. This conference provided a meeting place for IS researchers and practitioners from Eastern and Western Europe, as well as those from other parts of the world. Participants of the conference had an opportunity to express ideas on the current state of the art in information systems development methods and tools, and to discuss and exchange views about new methods, tools and their applications. An objective of the conference was not only to share scientific knowledge and interests but to establish strong professional ties among the participants.

In response to the Call for Papers, 95 papers coming from 25 countries were submitted for the conference. Papers were refereed by the members of the International Programme Committee, helped by ad hoc reviewers. Papers based on the results of research work dealing with theoretical and practical aspects of information systems development were selected for presentation at the conference and publication in the Proceedings, which includes 3 invited papers, 64 submitted papers and 16 presentations of the Work in Progress. Together with invited guests, 125 participants from 26 countries participated the conference, among them 55 from Slovenia.

The majority of papers focused on the following topics:

- Modelling IS development process: models and meta models, modelling techniques and tools
- Human, social and organizational dimension of IS development
- Reconciliation of human and technical factors of IS development
- Theoretical foundations, new paradigms and trends of IS development
- Education and training of IS personnel and users

The panel discussion "Relational versus Object Data Bases: Concatenation or Coexistence" which concluded the conference, also attracted a considerable number of delegates to express their views and participate in the discussion. Positions of the panellists will be published and mailed to participants of the conference.

Invited papers presented at ISD'94 dealt with the impact of information technology on teaching practice of information system development (Milton Jenkins, University of Baltimore, USA), information systems planning in small business (Georgios Doukidis et al., Athens University of Economics and Business, Greece) and development of information systems to support electronic commerce (Jože Gričar, University of Maribor).

The Forth International Conference on Information System Development - ISD'94 continues the concepts of the first Polish-Scandinavian Seminar on Current Trends in Information System Development Methodologies, held in Gdansk in 1988, and the Second and the Third International Conferences on Information Systems Developers Workbench (Gdansk 1990, 1992). The next conference - ISD'96, will be held in Gdansk again in September 1996.

Jože Zupančič

19^{th} ÖAGM and 1^{st} SDRV Workshop

Visual Modules

11–13 May 1995

Conference Cochairs

Walter Kropatsch
Technical University of Vienna
Institute for Automation
Department for Pattern Recognition and
Image Processing
Treitlstr. 3 / 1832,
A-1040 Wien, Austria
Tel: 43 (1) 58801 8161,
Fax: 43 (1) 569 697
E-mail: krw@prip.tuwien.ac.at

Franc Solina
University of Ljubljana
Faculty of Electr. Eng.
and Comp. Science
Tržaška 25, 61000 Ljubljana, Slovenia
Tel: 386 (61) 1768 389,
Fax: 264 990
E-mail: franc@fer.uni-lj.si

Call for Papers

19^{th} ÖAGM Workshop and 1^{st} SDRV Workshop

11–13 May 1995

Technical Faculty, University of Maribor
Maribor, Slovenia

The Slovenian Society for Pattern Recognition (Slovensko društvo za razpoznavanje vzorcev – SDRV) and the Austrian Society for Pattern Recognition (Österreichische Arbeits-Gemeinschaft für Mustererkennung – ÖAGM) organize a joint workshop on pattern recognition which will be held in Maribor, Slovenia.

Scope of the workshop:

image analysis and understanding, computer vision, neural networks, speech analysis and understanding, pattern recognition, applications.

Authors who wish to present a paper at the workshop should send three copies of their full-lenght draft paper (in English) to the Workshop Secretariat by 15 February 1995.

The papers should include:

1. the title of the paper,
2. author's name, address, telephone and fax number, and e-mail address,
3. abstract,
4. up to ten pages of text and figures.

A book of abstracts will be available at the workshop. Camera-ready papers which will be published in workshop proceedings will be due after the workshop.

Sponsors of the Workshop:
IAPR – International Association of Pattern Recognition
Slovenia Section IEEE
SATENA – Slovenian Association of Technical and Natural Sciences

Workshop secretariat:
Franc Solina
University of Ljubljana
Faculty of Electrical Engineering and Computer Science
Tržaška 25, 61000 Ljubljana, Slovenia
Tel: 386 (61) 1768 389, Fax: 264 990
E-mail: `franc@fer.uni-lj.si`

TEX and TUG News

Volume 3, Number 3, July 1994

In 'Typographers Inn', shading, typeset conventions, books, a kerning test, accent test, metafonts, and top publishing with FrameMaker are reported. A public list of all known METAFONT fonts is available.

In 'New Publications' interesting subjects are short book and article reviews. E.g.,

Leslie Lamport: *LATEX: A Document Preparation System.* Second Edition. Addison-Wesley, 1994. 180pp. US$41.50. ISBN 0-201-52983-1 (paperback). —This new edition of the official LATEX manual describes LATEX2e. It is announced to be available soon.

J. Grosvenor, K. Morrison and A. Pim: *The PostScript Font Handbook: A Directory of Type 1 Fonts.* Revised edition. Workingham: Addison-Wesley, 1992. ix,425pp. US$41.50. ISBN 0-201-56893-4 (paperback). —For all those interested (or forced) to use PostScript fonts this directory provides a comprehensive collection of font samples from the font libraries of Adobe, Monotype, Linotype, and Agfa. Each font is represented by a full-page profile including the character set, a brief history, and hints for recommended usage. It includes an introduction on PostScript fonts.

M.S. Margolis: *MG Mathematical Graphics System.* Notices of the American Mathematical Society, vol. 41, no. 3 (March 1994) 200–201. —The MG Mathematical Graphics System creates and displays two- and three-dimensional mathematical graphics on an MS-DOS-based personal computer and produces high-quality PostScript output. The developers, R.B. Israel and R.A. Adams, [...] primarily designed the relatively small program to assist authors to create graphs to include in TEX-typeset documents. It is available from MG Software, 4223 West Ninth Avenue, Vancouver, B.C., Canada V6R 2C6.

H. Werntges: *Grafik-Import in LATEX.* c't: 12/92, 252–258. —This survey article (in the German computer magazine, c't) discusses the various possibilities for including graphics into LATEX documents. It has since been reprinted (slightly updated) in *Die TEXnische Komödie* (vol. 5, no. 2, 38–53 (1993)).

In 'Hey — it works!', several short TEX programs and advises are described as A page-numbering scheme, Addresses and headings, A double summation sign, and Matrix icons via LATEX (rectangular, triangular, and upper Hessenberg form).

In 'Welcome to LATEX News', news on the newest LATEX, that is, LATEX2ε are published.

LATEX2ε—the new LATEX release is information on the most important release of the new version of the LATEX software. This version has better support for fonts, graphics and colour, and will be actively maintained by the LATEX3 project team. Upgrades will be issued every six months, in June and December.

Why a new LATEX? Over the years many extensions have been developed for LATEX. This is, of course, a sure sign of its continuing popularity but it has had one unfortunate result: incompatible LATEX formats came into use at different sites. Thus, to process documents from various places, a site maintainer was forced to keep LATEX (with and without NFSS, SLITEX, $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LATEX, and so on. In addition, when looking at a source file it was not always clear for which format the document was written.

To put an end to this unsatisfactory situation a new release of LATEX was produced. It brings all such extensions back under a single format and thus prevents the proliferation of mutually incompatible dialects of LATEX 2.09. The new release was available for several months as a test version, and then final release of 1 June officially replaces the old version.

Processing documents with LATEX2ε. Documents written for LATEX2.09 will still be read by LATEX2ε. Any such document is run in *LATEX 2.09 compatibility mode.*

Unfortunately, compatibility mode comes with a price: it can run up to 50% slower than LATEX2.09 did. If you want to run your document in the faster *native mode,* you should try replacing the command

```
\documentstyle[<options>,
```

`<packages>]{<class>}`
with
`\documentclass[<options>]{<class>`
`\usepackage{latexsym,<packages}`

New packages. LaTeX2ε has much better support for graphics, colour, fonts, and multi-lingual typesetting. The following software should be available from the distributor who brought you LaTeX2ε:

babel for typesetting in many languages.

color for colour support.

graphics for including images.

mfnfss for using bitmap fonts.

psnfss for using Type 1 fonts.

tools other packages by the LaTeX3 team.

The packages comes with full documentation, and are also described in *LaTeX: A Document Processing System* or *The LaTeX Companion.*

Further information. More information about LaTeX2ε is to be found in:

LaTeX: A Document Preparation System, Leslie Lamport, Addison-Wesley, 2nd edition, 1994.

The LaTeX Companion, Gossens, Mittelbach and Samarin, Addison-Wesley, 1994.

The LaTeX distribution comes with documentation on the new features of LaTeX:

LaTeX2ε for authors describes the new features of LaTeX documents, in the file `usrguide.tex`.

LaTeX2ε for class and package writers describes the new features of LaTeX classes and packages, in the file `clsguide.tex`.

LaTeX2ε font selection describes the new features of LaTeX fonts for class and package writers, in the file `fntguide.tex`.

For more information on TeX and LaTeX, get in touch with your local TeX Users Group, or the international TeX Users Group, P.O. Box 869, Santa Barbara, CA 93102–0869, USA, EMail: `tug@tug.org`.

In '(LA)TeX News' the following is reported: The new LaTeX on CTAN, TIQWAH (a TeX-XeT/METAFONT/Flex package for the typesetting of Biblical Hebrew), User group info on CTAN, RELABEL on CTAN, New group forming, and News from vendors. E.g.,

> CDs are all the rage! Prime Time Freeware latest TeXoffering is called "Prime Time TeXcetera". "Prime Time TeXcetera" consists of a 100-page book and an ISO-9660 CD-ROM. The disc contains essentially the entire CTAN, as of mid-May, 1994. It uses Info-ZIP format, allowing use with a variety of computer systems.
>
> To make the disc easier to use, a substantial amount of annotation and indexing is added. The list price of Prime Time TeXcetera is US$60. For more information, send email to: `ptf@cfcl.com`.

In 'Reports on Meetings' some interesting issues have to be mentioned. At the annual conference of DANTE e.V. (the German-speaking TeX user group), a young student presented his self-programmed editor called Eddi4TeX – EddiTeX for short. This is a fantastic editor and user shell for TeX, running under DOS and OS/2.

> At the 13th meeting of the NTG (the Netherlands), Wietse Dol demonstrated the full installation process needed to run the NTG 4allTeX CD-ROM on a 'virgin' PC. A little less than 45 seconds was all it took, after inserting the CD-ROM and before the first `sample.tex` was compiled, viewed and ready to print. The price of CD-ROM plus booklet is Dfl 60 or US$35 at NTG Secretary, P.O. Box 394, 1740 AJ, Schagen, The Netherlands or e-mail `ntg@nic.surfnet.nl`.

The enframed information may be of essential interest for the authors and readers of *Informatica.* On the first CD-ROM the entire stuff concerning TeX is given. On the second CD-ROM a collection of everything concerning the European emTeX is not only copied but also edited and supported for an easy-way installation. Prices are quite acceptable in comparison to some TeX, LaTeX, etc. vendors.

A.P. Železnikar

THE MINISTRY OF SCIENCE AND TECHNOLOGY OF THE REPUBLIC OF SLOVENIA

The Ministry of Science and Technology also includes the Standards and Metrology Institute of the Republic of Slovenia, and the Industrial Property Protection Office of the Republic of Slovenia.

Scientific Research and Development Potential

The statistical data for 1991 showed that there were 230 research and development institutions, organizations or organizational units in Slovenia, of which 73 were independent, 32 were at the universities, and 23 at medical institutions. The remainder were for the most part departments in industry. Altogether, they employed 13,000 people, of whom 5500 were researchers and 4900 expert or technical staff.

In the past 10 years, the number of researchers has almost doubled: the number of Ph.D. graduates increased from 1100 to 1484, while the number of M.Sc.'s rose from 650 to 1121. The 'Young Researchers' (i.e. postgraduate students) programme 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 fields of research:

	Ph.D.	M.Sc.
Natural Sciences	315	217
Engineering-Technology	308	406
Medical Sciences	262	174
Agricultural Sciences	122	69
Social Sciences	278	187
Humanities	199	68
Total	1484	1121

Financing Research and Development

Statistical estimates indicate that US$ 260 million (1.7% of GNP) was spent on research and development in Slovenia in 1991. 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 crisis. This crisis 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 R&D spending: its share of GNP doubled from the mid-seventies to 0.86% in 1993.

Overall, public funds available for Research & Development are distributed in the following proportions: basic research (35%), applied research (20%), R&D infrastructure (facilities) (20%) and education (25%).

Research Planning

The Science and Technology Council of the Republic of Slovenia, considering initiatives and suggestions from researchers, research organizations, professional associations and government organizations, is preparing the draft of a national research program (NRP). This includes priority topics for the national research policy in basic and applied research, education of expert staff and equipping institutions with research facilities. The NRP also defines the mechanisms for accelerating scientific, technological and similar development in Slovenia. 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 parliament as a whole. 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 provides organizational support for the NRP, but it 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: International quality standards of groups and projects, relevance to social development, economic efficiency and rationality of the project. The Ministry finances research or co-finances development projects through public bidding and partly 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 cooperation 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.

In evaluating the programs and projects, and in deciding on financing, the Ministry works closely with expert organizations and Slovene and foreign experts. In doing this, it takes into consideration mainly the opinions of the research leaders and of expert councils consisting of national research coordinators and recognized experts.

The Ministry of Science and Technology of the Republic of Slovenia. Address: Slovenska c. 50, 61000 Ljubljana. Tel. +386 61 131 11 07, Fax +38 61 132 41 40.

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 1259 199, 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

REVIEW REPORT

Basic Instructions

Informatica publishes scientific papers accepted by at least two referees outside the author's country. Each author should 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. 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. The names of the referees should not be revealed to the authors under any circumstances. The names of referees will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees.

It is highly recommended that each referee writes **as many remarks as possible directly on the manuscript**, ranging from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks, and if accepted also to the Contact Person with the accompanying completed Review Reports. The Executive Board will inform the author that the paper is accepted, meaning that it will be published in less than one year after receiving original figures on separate sheets and the text on an IBM PC DOS floppy disk or through e-mail – both in ASCII and the Informatica LaTeX format. Style and examples of papers can be obtained through e-mail from the Contact Person.

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

Date Sent:

Date to be Returned:

Name and Country of Referee:

Signature of Referee:

Name of Editor:

Title:

Authors:

Additional Remarks:

All boxes should be filled with numbers 1-10 with 10 as the highest rated.

The final mark (recommendation) consists of two orthogonal assessments: scientific quality and readability. The readability mark is based on the estimated perception of average reader with faculty education in computer science and informatics. It consists of four subfields, representing if the article is interesting for large audience (interesting), if its scope and approach is enough general (generality), and presentation and language. Therefore, very specific articles with high scientific quality should have approximately similar recommendation as general articles about scientific and educational viewpoints related to computer science and informatics.

☐ SCIENTIFIC QUALITY

☐ Originality

☐ Significance

☐ Relevance

☐ Soundness

☐ Presentation

☐ READABILITY

☐ Interesting

☐ Generality

☐ Presentation

☐ Language

☐ FINAL RECOMMENDATION

☐ Highly recommended

☐ Accept without changes

☐ Accept with minor changes

☐ Accept with major changes

☐ Author should prepare a major revision

☐ Reject

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 (attached) 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).

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 (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: .....

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 18 Number 1 April 1994 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Contents: