Volume 13 Number 2 April 1989
YU ISSN 0350-5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Informatica

A Journal of Computing and Informatics

Subscription Information

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

The subscription price for 1989 (Volume 13) is US$ 30 for companies and US$ 15 for individuals.

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

Printed by Tiskarna Kresija, Ljubljana.

Informacija za naročnike

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

Letna naročnina v letu 1989 (letnik 13) znaša za podjetja 48000 din, za zasebne naročnike 12000 din, za študente 4000 din; posamezna številka 16000 din.

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

Tisk: Tiskarna Kresija, Ljubljana.

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

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

Volume 13 Number 2 April 1989
YU ISSN 0350-5596

Informatica

A Journal of Computing and Informatics

EDITOR-IN-CHIEF
Anton P. Železnikar
Iskra Delta Computers, Ljubljana

ASSOCIATE EDITOR
Rudolf Murn
Jožef Stefan Institute, Ljubljana

The Slovene Society INFORMATIKA
Ljubljana

Volume 13 Number 2 April 1989 YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

CONTENTS

INTEGRAL, IMPLICITY INTELLIGENT SYSTEMS

INFORMATICA 2/89

Keywords: intelligent system, knowledge, brittleness, inflexibility, frame problem

Ivo Špigel
OZIR, Zagreb

ABSTRACT Artificial intelligence systems today suffer from problems that are widely acknowledged within the discipline: brittleness, inflexibility, the frame problem and others. These problems are largely due to insufficient methodological foresight in system design. In particular, reduction of a system into components and the explicit representation of knowledge (frames, rules etc.) are misused. Research has begun at OZIR to investigate a different class of systems: integral and implicitly intelligent. This paper explains the hypotheses involved and the direction of research.

INTEGRALNI, IMPLICITNO INTELIGENTNI SUSTAVI Sustavi umjetne inteligencije kakve danas poznamo opterećuju problemi kao što su nefleksibilnost, problem okvira i drugi. Nedovoljna metodološka analiza u pristupu velikim dijelom je uzrok ovakvom stanju. Konkretno, rastavljanje sustava na dijelove i eksplicitno predstavljanje znanja pogrešno se ili nepotrebno primjenjuju. U OZIR-u je započeto istraživanje nove klase sustava: integralnih, implicitno inteligentnih. U ovom radu iznesene su hipoteze na kojima se istraživanje zasniva te osnovni koraci istraživanja.

I.

Artificial intelligence systems as we know them today suffer from various widely recognized problems and limitations. Some of these problems are due to lack of methodological insight on the part of researchers uncritically rooted in the rationalistic tradition. This paper is the announcement of a research project which has begun at OZIR. The aim of this project is the introduction of a new class of machines which should overcome some of the problems limiting current implementations. These systems, of course, will suffer from problems of their own. Hopefully, a shift in problem focus will mean a little progress for the field.

Within the paper, I will focus on some fundamental problems of artificial intelligence that are:

a) omnipresent throughout the discipline
b) important enough to be stiffling scientific progress

Solutions to these problems will be proposed in the paper and explored in implementations.

The structure of the paper is as follows. In the first part, the problems are identified. An explanation of their origin and importance is provided. In the central part of the paper, a new approach is advocated. Finally, research motivations and some inevitable social issues are considered.

Before we continue, a brief summary of the problems and the proposed solutions:

a) AI today studies and designs separate parts of intelligent systems. As these parts are in fact inseparable, this decomposition leads to inherent limitations. The solution is to study and design integrated systems.
b) Knowledge is represented explicitly. What needs to be represented is not knowledge, but the world of an intelligent subject, and this representation should be implicit.

A brief remark. In this discussion, I consider intelligence to be exhibited by animal life in general. I also consider human intelligence to be essentially of a higher order than that of other animals. I will usually refer to general intelligence as »intelligence«, and explicitly denote »human intelligence«, except where the context makes the difference clear.

II

The Reductionist Eastern Sin

AI is a young discipline. Not many people within the field have engaged in an analyisis of the methodological structure at hand, Winograd & Flores being a notable exception {WF}. The views expressed here are strongly influenced by theirs, as well as by those of Doug Hofstadter {DH}.

The basic approach taken in AI research faithfully follows the rules of Western rationalistic tradition, above all the principle of *divide et empera*. »Let us model our devices after man, and since this is a very complicated model, let us look at some of the parts before assembling the whole.« Some of the parts, however, when assembled, give only a sum of the parts, and this is not what an intelligent system is about.

We have today a variety and richness of subfields and techniques. Computer vision, speech recognition and others study perception. Robot manipulators and legged robots study the limbs and motion – the motoric system. Expert systems, cognitive science and many more study the cognitive domain. It has become obvious by now that none of these domains have integration with other domains as their long-term goal. This is not surprising.

An intelligent system must be conceived as an integrated system.

What exactly does this mean? Well, it amounts to a statement that reductionism and holism must be well balanced in the methodological structure of our discipline. Let me explain further.

In the present situation, reductionism is misused. The model is reduced in *structure*, while retaining *scale*. We need to reduce the scale, while holistically retaining the structure.

Look at the way children do things. There is always a lot to be learned from them. When a child models a man or a woman, it does not practice on an arm, or a head, or a leg for years, before moving on to the next part. The child »constructs« the whole person, although in a simplified, rudimentary way. The same holds for toy automobiles, or castles in the sand. Children do it that way because it is the *natural* thing to do.

Let us suppose we still wanted to model part by part, function by function. Let's start with the motoric system. How do we walk? We walk with legs. Or do we? Can you imagine one leg walking by itself? Or does it take two legs to walk? Two legs — and no body? Our whole body walks. Our hands walk. Our shoulders, ears and nose walk. And especially our eyes — they do a lot of the walking.

So it isn't at all easy to seperate the bodily functions into picture-book parts. The way to proceed is to study intelligent systems as integrated, and design them as such.

Time & Space

Steps In Time

The development of intelligence was, and still is, a process — not a sudden event. This is reflected in the structure of intelligence. Given that, we can accelerate this process, but we cannot skip it altogether, if machines are to be more intelligent than they are today. Let us now consider time on two scales of magnitude: the scale of the species/society and the scale of the individual.

Regarding time and intelligence, I consider some facts to be completely obvious and beyond interesting discussion. There was a time when there was no animal life, and thus no intelligence. At some point later in time, animal life had appeared, and with it intelligence. At this point, there were no people, and no human intelligence. Today, there are people, and they embody human intelligence.

On the scale of the individual, there was a time for each of us when we did not exist. At some later point, we exist, exhibiting intelligent behavior.

These are constant points in time where the relationship between time and intelligence is obvious and simple. The periods of time of interest to AI researchers lie between these points. They are the transient periods:

1) The period of emerging intelligence. During this period, the process of formation of general intelligence took place.
2) The period of the formation of man, relating to the process of the formation of human intelligence.
3) The prenatal/postnatal period in the life of a baby, during which each of us goes from splitting cells to saying »mama«. This period extends from roughly 2 months after conception to roughly 24 months after the birth of the child.

If machines are to be intelligent, they must go through the processes bounded by these transient periods.

The analogies are obvious. Intelligent machines correspond to animal life. The best intelligent machines might correspond to people. Each individual machine must go through a process of intelligence formation.

Let's look more closely at what this implies. We shall not begin by modelling man. Study and design shall begin from the simplest forms of intelligent life — maybe worms or insects[1]. If and when we succeed in building a true artificial simple animal, we might move on to higher forms. *In this way, scientific research can naturally reflect evolutionary processes in nature.*

On the other hand, the machines we design must go through individual processes of intelligence formation. This means that we must find out as much as we can about the structure of an intelligent system at the beginning of the third transient period, i.e. what's hard-wired into an animal at the moment it starts learning by itself. We must then reproduce this structure appropriately in the machine and let it develop, if you will, through self-organization (a favorite buzzword lately).

A Spatio-Temporal Outlook on Life

Time and space are forms of perception. There may be others, but we certainly don't know about them and cannot imagine them. Intelligence as we know it exists in a spatio-temporal framework. Try imagining a world without time, or a world without space {IK}. Thus, time and space are essential to intelligent systems in an even more important way than as a crucial factor in their development.

An intelligent system, must therefore have the ability to perceive both time and space. This »mechanism« cannot, of course, be simplistic, in the form of, say, predicate logic: **before(X,Y)**, **after(X,Y)** etc. The system must be organized in such a way that temporal perception is a consequence of the organization, meaning that every activity of the system and every entity within the system is in some way spatio-temporally situated.

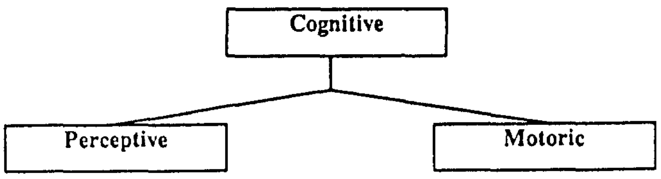
III

Integral, Implicitly Intelligent Systems

Some of the reasons for the belief that intelligent systems must be integral to a degree, and their intelligence implicitly defined, have been laid down. I would now like to go into this in some more detail.

Structure of the System

Integral system design does not mean unstructured design. Integral as it is, the system must have a fundamental structure. This structure is defined by three basic functions:



These basic functions are realized by basic subsystems of an intelligent system. It is essential that they be tightly coupled. Analysis of the internal structure of each subsystem and the organisation of the complete system, defined by the relationship between the subsystems, as well as design based on this analysis, is to be a key aspect of the research being proposed in this paper.

It is necessary for intelligent systems to have at least two sources of perception, in order for them to be able to provide feedback for each other[2]. This coincides nicely with the structure given above, even if the basic perceptive function contains only one source of perception itself. Namely, for the motoric function to be intelligent, i.e. to be part of an intelligent system, it must be able to provide feedback information to the two other functions. Given a system with only one »explicit« perceptive subsystem, *feedback from the motoric system* can be used as another source of perception.

The Body

If anybody needs to be convinced that locomotion is an essential characteristic of intelligent systems, let us consider an intuitive argument: All animals move. No plants move. As has already been said, it is animal life that we consider to be intelligent. If that is not convincing, we can say that for present purposes the necessity of motoric abilities for intelligence is a conjecture, or hypothesis.

Man, as we see, does not walk by legs alone. In fact, no animal does. Birds fly with their whole bodies, and fish and sea mammals swim with theirs. The whole body of an animal, then, is it's locomotion system. The mechanical finesses of live bodies seem too complex to be modelled at the present moment. However, we need not copy nature in every detail. Locomotion can be achieved through the simple machinery available to us, as long as one condition is met.

There is more intelligence in the movement of the humblest worm than in the nicest robots of today. This seems quite obvious to me, considering the complexity of motion of either system. The question is: how come? The reason is that the worm, in it's wormlike rudimentary way, *learned to move*. This fact is represented in it's nervous system by flexibility. A worm will have no trouble at all climbing up any number of different branches during it's life – it just doesn't appear to care whether they are the same or not.

An intelligent system, then, cannot be preprogrammed to move. It must be preprogrammed to *learn to move*, and then it must have a *necessity to move*, forcing it to learn what it has been programmed to learn – to move. Only in this way can truly intelligent motion be designed, making the motoric system an intelligent subsystem of the whole.

The Senses

If the intuitive argument for the necessity of motoric ability for an intelligent system doesn't seem quite convincing, I hope nobody will have to be convinced that perception is a necessary prerequisite for intelligence. Nevertheless, ideas of a »brain in a bottle«, meaning a completely isolated »intelligence« might sound intriguing to some. I should refer those interested to Kant {IK} and Dreyfus {HD}, where the impossibility of such a concept is explained. Dreyfus, of course, takes his argument too far, deducing that AI is impossible in principle, which does not follow from his discussion. For present purposes, we shall assume that perception is essential for intelligence.

Intelligence, on it's part, is essential to perception. Artifical intelligence researchers have learned this the hard way, especially in vision research. It has long been realised that brute-force perception is impossible and »knowledge-based« systems are the only answer. Interestingly enough, none of the researchers have concluded what seems rather obvious. You cannot simply »add« intelligence to a perception system. Intelligence is not pepper or paprika. Perception systems can truly function only as subfunctions of an integrated intelligent system.

To give a vivid example, let's focus on vision for a while. How is it that we see so well? Many factors are vital to our ability, and I shall mention only a few.

Prediction. When we open a book, we don't expect a computer to fall out of it. We expect to see letters, numbers, pictures. Two-dimensional symbols on a piece of paper. These symbols exist as such for us, they are a part of our world. Our perceptive system is, therefore, guided and aided by the cognitive system in the process of vision. Automatically, the area of interest within our vision field is determined by this expectance. We don't scan the whole scene stupidly like the vision systems of today, since we know what to look for and where to find it.

Concentration. Do we see everything we see, and do we hear everything we hear? How many times have you been caught reading comics in class, having completely forgotten about the teacher and what she had been saying? »Humberto, repeat what I just said!« »Sorry, teacher, I didn't hear you.« The perceptive and cognitive system work together to focus only on the issue of importance to the whole system, ignoring others and eliminating or reducing unnecessary processing.

An analysis of the »tricks« intelligent systems use to be able to cope with the truckloads of information perceptive subsystems are constantly receiving has not been conducted within this research effort yet, but will obviously be necessary to guide us in design. Once again: there is no intelligence without perception, and no perception without intelligence. The two cannot be separated. Tightly coupled with the motoric system, they constitute an integrated intelligent system.

Cognition

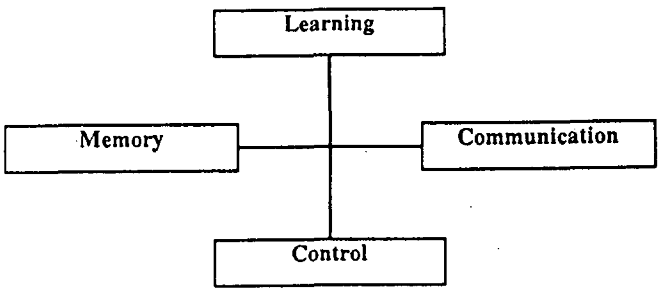
I would have liked to call this section »The Mind« in accordance with previous headings. It would have been misleading, however, strengthening further the old belief that we walk by legs alone, see by eyes alone, and use our brain only when doing mathematics (excuse the exaggeration).

Within the context of this discussion, I prefer using the term »cognitive subsystem« for the subsystem that does what we usually call »reasoning«. In analogy with considering animals intelligent in a general sense of the word, I consider them to be able to »reason«, or, if you will, reason, in a very, very general sense. I am convinced, however, that the reasoning of a frog is far superior to the »reasoning« of any existing expert system, including MYCIN, XCON, Urologist etc. etc. Far superior, in fact, to any expert system that will ever be built and still deserve the name.

An expert system doesn't know what it's talking about. This is such a notorious fact and has been so nicely illustrated by Doug Lenat that I call it »the GENSYM problem« after his example {DL}. The fact that Lenat is not capable of deriving the consequences of his own insight is sad, but not a central matter at the moment.

Imagine having MYCIN talk to you and use, instead of those nice English words, unique variables generated by the LISP »gensym« function. »The patient should be treated with neocarboanimalis« turns out to be »lkhj llkhj lčkkhjč kj ljkčlhjk uiooz mnnmbqwertzuiopš« , and you get a certainty factor of 0.8. You wouldn't be too happy. The machine, however, couldn't care less. Putting it more precisely, the symbols generated would have the same meaning as the English wording – none whatsoever.

This is the result of formulating knowledge explicitly. Conceptual dependency, frames, scripts, etc. etc. – nothing solves this problem. Just how unaware AI researchers are of this problem can be seen when proffesors claim that the essence of AI is developing better, more precise formalisms. Neural networks appear to be a very sound step in the right direction, but back to those later. Right now, let's look at the basic functions of the cognitive subsystem:



Learning

A fascinating characteristic of natural learning systems is their non-linearity. A child will need many months to speak her first word. The next one will soon follow, and the speed will increase with the size of the vocabulary. This holds for other domains: motion (walking) for instance. The seeming difficulty, however, of grasping elementary concepts is not a drawback of the system, but a reflection of it's strength. The same structural complexity that makes it hard to enter a knowledge domain endows the system with power and flexibility later on.

There is a well known word in many parts of Yugoslavia for uncreative hard-working pupils. They are called »shtreberi« and not very much respected by their bright, lazy colleagues in this country or elsewhere. The fundamental flaw in the knowledge of these kids is a lack of real understanding of the subject matter. In extreme cases, they learn it by heart without having any idea of what they are saying.

No matter how sophisticated machine learning schemes may be today, they are truly ideal »shtreberi«. They have absolutely no understanding of the subject matter and do not relate their »knowledge« to reality.

One of the key reasons for this situation is that AI systems have no real contact with the real world, i.e. they are completely isolated from experience. In a metaphorical way, authors of expert systems talk of their »experience« in the domain and the way they learn from it, but this is of course quite far from truly experiencing the world of an intelligent subject.

The only road to *knowledge* is through *learning* from *experience*, and so these three key issues are inextricably interwoven. This does not mean we are lost in a vicious circle. We must carefully analyse the rudiments of knowledge and the mechanism for learning that are present in existing intelligent systems in the period in which they grow from bunches of splitting cells to animate organisms (albeit prenatal) capable of learning from their experience. It is only this much, or rather an analogy to it, that we shall explicitly program into the system we are designing.

The fact that systems learn through contact and experience in the real world means that we cannot build two identical systems, since they cannot share worlds. »The objective world«, namely, does not exist as such. The world that matters to an intelligent system is it's own, unique world, so through learning from this world each system develops unique knowledge.

As the system develops and ages, it loses flexibility. Is this only because the intelligent systems we know are biological? I should think the reason lies in the structure that provides it with learning capabilities in the first place.

Memory

The concept of artificial memory is as old as computers are. One could speak of earlier data storage systems, such as libraries, as memories, but this would be stretching the concept a bit.

The ability of computers to store and retrieve data can be quite confusing when discussing intelligent systems. Apparently, we don't have to worry about enabling the system to remember the way we have to worry about enabling it to learn. This, of course, is a misconception, much in a way electric motors and wheels do not automatically guarantee intelligent motion. We have to purposely make systems remember the hard way, in an intelligent manner. RAM and ROM can only give us the technological foundations.

In biological intelligent systems, a bunch of neurons do not memory make. There are intricate and complex memory mechanisms at hand, and even neurophysiologists do not completely understand them. Since we are mere engineers, it is not our job to analyse memory from a neuroscientist's point of view, but we must at least be in touch with what *is* known. If we model this well enough, they might benefit from the insight gained in empirical experiments.

Among other aspects, the structure of our memory provides us with the notion of time. The way short–term and long–term memory function and communicate, the way earlier events are »buried deeper« in memory, these and other mechanisms are essential to intelligent behaviour. We shall therefore provide our designs with at least rudimentary analogies of what we know, in hopes of being able to introduce more complex structuring with the passing of time and the growth of our own experience.

Control, Communication

Some views on control have already been explained in the section on motion. Let it suffice for the present to say that there is a certain distinction between control and communication from the position of the intelligent subject that may not be acceptable to all. Namely, communication can be seen as an act of controling one's communication mechanisms, or subsystems. I consider communication to be important enough to be considered separately.

Apparently, communication is as widespread in the world of intelligent systems as motion. That is, all intelligent systems communicate in some way, and no unintelligent ones do.

We are only beginning to understand the structure of animal communication, and we are constantly pushing the limits of what we see as the potential for animals to communicate in a way more familiar to us.

Whatever forms communication takes, however, it is always intelligent in the sense that the communicating agent is. Bees do not talk. AI, however, has been trying to make completely unintelligent systems communicate in spite of this. The GENSYM problem in communicating with expert systems has already been explained. Another nice example are speech generation systems, neural network based or not. These systems do not communicate in any substantial sense of the word – they simply transform text from one form into another. This is not what is needed. An intelligent system must communicate for a reason, and in a way that corresponds closely to it's structure and complexity. We shall start by building simple systems with rudimentary communication abilities in order to gain a deeper understanding of the processes involved and, more importantly, because the gradual evolution of one system from another is crucial to the structure of higher order intelligent systems.

IV

Where Do We Begin?

What, then, are the implications of what has been said to design and research? The reason for research is an inquest into the nature of intelligence. A set of design principles should follow from insights thus gained, enabling experimental validation of various hypotheses.

The first stage of research involves building a working model of an animal of rudimentary intelligence. This stage breaks up into four steps:

1. Selection of an animal to model. We need to decide upon a specific animal. Our selection criteria are that the animal be as simple as possible, while still displaying rudimentary intelligence. The earthworm is a potential candidate at the moment.
2. Analysis of the structural and functional organization of the animal in terms of the motoric, perceptive and cognitive subsystems described above.
3. Mapping of this organization onto a system lending itself to practical realization based on the computer technology available to us. This, of course, is the crucial step, and amounts to building the core of the model.
4. Specification of a system based on this mapping and physical implementation. The result should basically be a moving robot, which need not necessarily be a physical analogy of the animal (e.g. in terms of a similar locomotion system), but the functional and structural mapping should preserve the basic organization of the original.

Various basic functions and subsystems of our design represent different design problems. One of the fundamental differences is that in some areas – sensing, learning – the problem is development of the function, while in others – motion, memory – there is double trouble because of the need to refrain from the tantalizing possibilities offered by technology. In a Baconian way, we must hang weights on the wings of technology, providing the system perhaps with wheels and motors, but depriving it of the luxury of ready–made control software. The system, like a child, must be forced to learn to do things by itself. Otherwise, it will be the perfect spoiled child – completely unable to cope to a degree that will make it unintelligent.

This fairly short specification implies quite a few design problems, relating to issues already mentioned as unclear in the text. If we can solve these problems even in a rudimentary way, the step from an artificial bug to an artificial frog might be much easier to take, much as the child has the most trouble grasping elementary concepts.

V

Related Issues

Knowledge Representation

The fundamental problem with knowledge representation has already been mentioned in the text, but since this is such an important and favorite child of AI researchers, I would like to say a few more words about it.

Widely accepted concepts usually have an implicit justification that is not necessarily true. The justification for parliamentary government, for instance, is the technological inability of society to enable everybody to directly influence decisions of general importance. This was O.K. until yesterday, but information and telecommunications systems are rapidly challenging the justification.

A similarly implicit, only completely false justification for knowledge representation schemes is that we *must* represent knowledge in some way, since we shurely don't carry people, houses and elephants around in our heads. The only problem with this is that *knowledge* is not what is represented in the cognitive system – it is the world, our world, the unique world of the subject itself. Various knowledge representation schemes are actually representations of their author's understanding of how people's mind's work. This understanding is not necessarily complete in each particular case.

Storing knowledge explicitly, in the form of frames, rules, logic etc. amounts to creating an illusion that the system understands. Joseph Weizenbaum realised the implications fully a long time ago, and Margaret Boden has extensively commented on the matter {JW, MB}. The systems and schemes, however, proved quite useful and have remained with us to this day.

I don't claim to have any deep insight into the workings of the mind or of the brain. However, I am painfully aware of this, as well as of the fact that I cannot devote the rest of my life to any of the numerous scientific disciplines covering the domain. My colleagues and I must learn enough to be able to communicate creatively with psychologists, neurophysiologists and many others. A collective, interdisciplinary effort is required to solve these difficult problems, and no ad hoc, simplistic solutions will do.

Knowledge must be implicit, and intelligence is an epiphenomenon of the organization of the system.

The Frame Problem

The frame problem is notorious in AI. One of the most frustrating aspects of this problem is the apparent ease with wich people handle it. Now, where did this *enfent terrible* originate in the first place?

The roots of the frame problem lie in Eden, in the days before AI commited it's eastern sin. When knowledge is represented explicitly through clever ad hoc schemes, the problem can be solved only by devising still cleverer and clevererer and ... counterschemes to tackle it.

When the organization of the system embodies knowledge, everything that is known is distributed within the system as far as is natural to the context. The consequences of any action are then limited by the distribution of the entities involved in the system and naturally affect only the appropriate environment.

The frame problem is a non-problem if the system is intelligent in an integrated, implicit way.

Neural Networks

The architectural concept of neural networks has begun to answer one of the fundamental problems mentioned in this text – the problem of implicit representation of knowledge. Immediately, results have shown them to be superior to standard techniques in many application areas.

This concept, however, does not address the other fundamental issue – integration. The idea of an integrated system built around a neural network seems promising. Interestingly enough, even the staunchest critics of AI {HD} seem to be sympathetic, or reserved in the worst case, when discussing NN's.

Motivation & Social Aspects

In my experience, it has not usually been the case that scientists analyse their personal and social motivations for doing what they are doing. Again, there is an implicit justification for research, rooted in the Western Judeo–Christian tradition, stating basically that scientific advances automatically benefit all humanity. This has, of course, been questioned strongly in this century, and many people believe today that the major contribution of the space program to society has been the teflon pan. I would almost agree with this view, even if it is a bit extreme.

I believe that a strong personal motivation is present in doing AI research. There is a starnge feeling of playing God about this discipline: create something in your own image, something which behaves remarkably like yourself. This motivation has been furthered by ad hoc concepts such as the Turing test, which even define artificial intelligence as the ability of a machine to imitate a human being.

AI is among the few sciences which have the potential of rapidly breeding enormously powerful technology, thus potentially threatening many people, either through weapons or social unrest resulting from industry transformation. This does not mean it shouldn't be investigated. We who are doing it, however, must be intensely aware of the implications and possible consequences of our work. It is our social responsibility to guide and control the results of our research whenever we can, and avoid involvement with projects where they might be misused.

I agree largely with Weizenbaum {JW} that application domains of AI systems should be carefully selected. Where exactly to draw the line can be a matter of discussion, but it won't do to just blindly stumble into any domain that one considers interesting without giving some thought to the consequences of developments within that domain.

An AI School

AI research as it has been described here requires a different sort of education than any of us get today, in Yugoslavia, in the States or elsewhere. Specialization, so dominant a trend in the decades past, will not suffice any more.

An institution is needed that will provide AI students with a wide knowledge of domains central to the discipline: mathematics, philosophy, computer science, biology, neurophysiology, linguistics, psychology and others. Students would, of course, specialize in one aspect and research subject, but the depth of insight into one particular domain must be partially sacrificed to make way for a breadth of knowledge that is a necessary prerequisite for studying and designing integral, implicitly intelligent systems.

Prologue

There can be no strictly formal theory of artificial intelligence. Human intelligence is creative, thereby having the potential of always going one step beyond any formal definition. This means that we cannot define intelligence in general, because we cannot define it's most interesting form: creative intelligence. Since we cannot define one of the central concepts, we cannot ever hope to construct a full, complete system deserving to be called a »theory of artificial intelligence«.

Given this, we at OZIR have stopped worrying about definitive solutions and are trying to do the best we can with what we have. This is all we are attempting by embarking on the study and design of integral, implicitly intelligent systems.

Acknowledgments

I wish to thank all my colleagues at the Department for the Study of Intelligent Systems at OZIR for many hours of inspiring discussion and a few years of AI research, especially Ivan Marsic who began it all. This is the end of the paper but only the beginning of interesting research.

Notes

1. I owe this idea to Ivan Maršić.
2. Vukašin P. Masnikosa, Ivan Maršić. I would certainly like to see stronger biological support for this view, which I nevertheless consider intuitive enough to be convincing.

References

1. {MB} Margaret Boden: *Artificial Intelligence and Natural Man*, second edition, The MIT Press, London, England, 1987.
2. {HD} Hubert Dreyfus: *What Computers Can't Do*, Nolit, Belgrade (in Serbo–Croatian)
3. {DH} Douglas Hofstadter: *Godel, Escher, Bach: An Eternal Golden Braid*, Penguin Books, Harmondsworth, England, 1987.
4. {IK} Immanuel Kant: *A Critique of Pure Reason*, BIGZ, Belgrade, 1976. (in Serbo–Croatian)
5. {DL} Douglas Lenat et al.: *CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks*, AI Magazine, Vol. VI, No. 4, Winter 1986
6. {WF} Terry Winograd, Fernando Flores: *Understanding Computers and Cognition – A New Foundation for Design*, Ablex Publ. Co. Norwood, U.S.A., 1987.
7. {JW} Joseph Weizenbaum: *Computer Power and Human Reason*, Rad, Belgrade (in Serbo–Croatian)

INFORMATIONAL LOGIC IV

INFORMATICA 2/89

Keywords: logic of information, transformation rules, informational modus

Anton P. Železnikar
Iskra Delta

In this part of the essay the following topics of the informational logic (IL) are discussed: transformational rules of IL and a surveying conclusion concerning the formal IL. Various informational modi of informational transformation are presented. This part of the essay includes also the concluding remarks which concern IL in its entirety (references [15], [16], [17], and this essay).

Within transformational rules of IL, the following rules and modi are determined and examined: uniform and non-uniform informational substitution, informational replacement, and modus informationis with the topics as informational implication, informational modus ponens, modus tollens, modus rectus, modus obliquus, modus procedendi, modus operandi, modus possibilitatis, modus necessitatis, and further rules of Informing and the openness of introducing new transformational rules.

INFORMACIJSKA LOGIKA IV. V tem delu spisa se obravnavata še dve naslovni poglavji *informacijske logike* (IL): transformacijska pravila IL in pregled sklepov, ki zadevajo IL. Prikazanih je nekaj informacijskih modusov informacijske transformacije. Ta del spisa vključuje tudi sklepne opombe, ki se nanašajo na celoten spis o IL (na navedbe [15], [16], [17] in na ta spis).

V okviru transformacijskih pravil IL se opredeljujejo in raziskujejo tale pravila: uniformna in neuniformna informacijska substitucija, informacijska zamena in modus informationis z naslovi kot so informacijski modus ponens, modus tollens, modus rectus, modus obliquus, modus procedendi, modus operandi, modus possibilitatis, modus necessitatis in dalje pravila informiranja in odprtost uvajanja novih transformacijskih pravil.

II.4. TRANSFORMATION RULES OF INFORMATIONAL LOGIC

Information is the fuel of cognition. At its most basic level, information is a matter of structure interacting under laws. The notion of information thus reflects the (relational) fact that a structure is created by the impact of another structure. The impacted structure is an encoding, in some concrete form, of the interaction with the impacting structure. Information is, essentially, the structural trace in some system of an interaction with another system; it is also, as a consequence, the structural fuel which drives the impacted system's subsequent processes and behavior.

Radu J. Bogdan [13] 81

II.4.0. Introduction

By transformation rules, informational formulae can be transformed into different ones, which might have simpler, more complex, and also essentially different form and meaning in regard to the previous formulae. It is not always quite clear if formatting, axiomatizing, and transforming approaches can be separated from each other in a strictly evident or clear way. For instance, operations of informational particularization and universalization can have formatting as well as axiomatizing and transforming nature. Within IL, transformation rules transform axioms and already transformed formulae (iwffs) in a uniform, non-uniform, and modal (conditional, dependent, ontological, possible, necessary, true, false, random, etc.) way.

In regard to the uniform and non-uniform substitution there is nothing essentially new to saying. A uniform substitution of variables in a formula is the most common mood of substitution in mathematical formulae. With uniform substitution all variables of the same type will at a time be replaced by a determined formula. In the case of a non-uniform substitution this principle can be violated, thus, in some occurrences a variable will be replaced by a given formula and some not. In this way, non-uniform substitution offers more freedom as compared with uniform substitution.

The next possibility of substitution is the so-called informational replacement. In this case, a formula in a given formula can be replaced by another formula. Such a replacement can be uniform as well as non-uniform which depends on particular occurrences of a formula. As we shall see, the approach of informational replacement can lead to ambiguities when occurrences of distinct formulae overlap each other. In such cases strict rules of substitution must be determined to enable, for instance, substitutions in a parallel or simultaneous manner.

The most diverse transformation of formulae is possible by the use of the so-called informational modi. These various kinds of transformation, of information in general and of iwffs in particular, can be marked simply by modus informationis (MI). MI belongs to the central notions which concern informational transformation rules. MI is in fact a metainformational transformation rule, which by itself as an informational formula (iwff) can be, for instance, non-uniformly particularized, universalized, or informationally modified (by formatting, axiomatizing, and transforming). When particularizing or universalizing the so-called modus informationis, the following modi can be observed: modus ponens, modus tollens, modus rectus, modus obliquus, modus vivendi, modus procedendi, modus operandi, modus possibilitatis, modus necessitatis, etc. Various kinds of informational transformation arise within Informing of information with its arising, and various transforming principles are simply adopted with the embedded (incoming) information. Thus, transformational modi can be understood as essential, existential, and arising phenomena of the entire informational realm.

The main characteristics of any informational modus is the so-called informational extraction (coming into existence) of an arising informational part, which follows as an informational consequence from the current state of a relevant informational phenomenon. This process of extraction of information may concern very different notions, such as implication in traditional logic, detachment in modal logic, modus vivendi under circumstances of survival, modus operandi under circumstances of a possible success, etc. Particular informational modi appear to be only intentional, believing, teleological, etc. mechanisms of informational arising from an antecedent, conditioning, basic, causal, etc. into a consequent, resultant, non-basic, sequential, etc. informational relevance.

To shortly summarize the possibilities of iwffs transformation we can state the following: A set of informational transformation rules (ITR) licenses various informational operations on informational axioms and also on iwffs obtained by previous application of the ITRs. The iwffs obtained by applying of ITRs will be called informational theorems. An iwff is either an informational axiom or informational theorem of a given informational system. Within this system, an iwff is often called informational thesis.

II.4.1. Rules of Uniform and non-Uniform Informational Substitution

II.4.1.0. Introduction

Substitution belongs to the most general procedures of replacement of variables by formulae within symbolic formulae. A variable, or generally a symbol, is simply replaced by another sequence of symbols (formula) throughout a given formula or only some of variable occurrences are replaced while others are left unchanged. In fact, the process of substitution can be strictly determined or can be free in regard to the replacements of occurrences of a variable. In the first case we have to do with the so-called uniform, and in the second case with the so-called non-uniform substitution.

II.4.1.1. Rules of Uniform Substitution within an IWFF

For uniform substitution (without particularization and universalization) it is possible to state the following rule:

[Transformation Rule]^{DF1}:
We can adopt the following ITR of the uniform informational substitution: the result of a uniform replacing of any informational variable (the operand as well as the operator one) in an informational thesis by any iwff and sub-iwff, respectively, is itself an informational thesis. This rule can be formalized in the following way: let $_u\mathfrak{S}$ be the operator of uniform substitution and φ an iwff in which operand and operator variables $\xi, \eta, \ldots, \zeta$ occur, so that it is possible to write the functional form $\varphi(\xi, \eta, \ldots, \zeta)$. Let arbitrary iwffs $\alpha, \beta, \ldots, \gamma$ be given and let token "|" be the delimiter, which marks the end of $\mathfrak{S}$-operation. Then the result of the operation of uniform substitution is as follows:

$$_u\mathfrak{S}^{\xi,\, \eta,\, \ldots,\, \zeta}_{\alpha,\, \beta,\, \ldots,\, \gamma}\, \varphi(\xi, \eta, \ldots, \zeta)| = \varphi(\alpha, \beta, \ldots, \gamma)$$

Instead of this symbolism of substitution we can use the informational one, for instance,

$$\alpha, \beta, \ldots, \gamma \models_{\mathfrak{S}} \xi, \eta, \ldots, \zeta \perp_u$$
$$\varphi(\xi, \eta, \ldots, \zeta) \models_{=} \varphi(\alpha, \beta, \ldots, \gamma)$$

The meaning of this formula is the following: $\alpha, \beta, \ldots, \gamma$ substitute ($\models_{\mathfrak{S}}$) $\xi, \eta, \ldots, \zeta$ uniformly in ($\perp_u$) the formula $\varphi(\xi, \eta, \ldots, \zeta)$ resulting in ($\models_=$) the formula $\varphi(\alpha, \beta, \ldots, \gamma)$. Uniformly means that informational sets of entities $\alpha, \beta, \ldots, \gamma$ and $\xi, \eta, \ldots, \zeta$ are in the one-to-one correspondence. ■

II.4.1.2. Rules of Non-Uniform Substitution within an IWFF

If the uniform substitution within a formula $\varphi(\xi, \eta, \ldots, \zeta)$ always gives a single result, denoted as a formula $\varphi(\alpha, \beta, \ldots, \gamma)$, then the non-uniform substitution can give many different results, which can be denoted by a set of formulae $\{\varphi(\xi, \alpha, \eta, \beta, \ldots, \zeta, \gamma)\}$. Thus, we can adopt the following rule:

[Transformation Rule]^{DF2}:
We take ${}_n\mathfrak{S}$ as the operator of a non-uniform substitution and $\varphi(\xi, \eta, \ldots, \zeta)$ as an iwff, in which $\xi, \eta, \ldots, \zeta$ are occurrences of informational operand and operator variables. Now, let $\alpha, \beta, \ldots, \gamma$ mark arbitrary iwffs and sub-iwffs, respectively. Then we have a set $\{\varphi\}$ of results of the non-uniform substitution, i.e.,

$$ {}_n\mathfrak{S}^{\xi, \eta, \ldots, \zeta}_{\alpha, \beta, \ldots, \gamma}\, \varphi(\xi, \eta, \ldots, \zeta) | = \{\varphi(\xi, \alpha, \eta, \beta, \ldots, \zeta, \gamma)\} $$

where the appearance of informational variables $\xi, \eta, \ldots, \zeta$ in particular elements of $\{\varphi\}$ is not necessarily certain. Informationally, we can symbolize this formula also by

$$ \alpha, \beta, \ldots, \gamma \models_{\mathfrak{S}} \xi, \eta, \ldots, \zeta \perp_n \varphi(\xi, \eta, \ldots, \zeta) \models_= \{\varphi(\xi, \alpha, \eta, \beta, \ldots, \zeta, \gamma)\} $$ ■

II.4.2. Rules of Informational Replacement

The rule of replacement is a generalization of the rule of substitution, which concerns only particular variables like operands and operators. Replacement does not search only for variables but for symbolic sequences within an iwff, which may be particular iwffs or sub-iwffs occurring within a source iwff. In general, by a replacement operation, the occurring iwffs can be replaced by other iwffs. This kind of operation is generally not uniform and cannot be always unique because of the occurring formulae overlapping within an iwff. But, it is more or less obvious that through an informational replacement very complex changes or essential transformations of existing iwffs can be achieved. Informational replacement will belong to the legal rules of informational formula transformation.

From the philosophical point of view, informational replacement is an operation, by which given informational associations are replaced by other associations. Here, an association can be understood as a complex, actualized informational entity, which calls for an adequate informational completion, change, or reduction. In this respect, informational replacement is also a very habitual process of living information.

Let us determine the rule of informational replacement! The question is what to do in case of overlapping of the occurring iwffs within a given iwff. It is of course possible to prescribe particular strategies or rules of formula replacements. However, we shall not deal with such particular "algorithms" yet. We can simply state that it will not be prescribed in advance how the replacement process is to happen precisely. So, let us have the following rule:

[Transformation Rule]^{DF3}:
Let ${}_x\mathfrak{R}$ be an x-ized informational operator of replacement, where x is a replacement operator particularization. Let φ be a given iwff upon which the operator ${}_x\mathfrak{R}$ will act. In general, this formula may or may not include some particular iwffs, relevant to the replacement, which could be replaced by formulae $\alpha, \beta, \ldots, \gamma$. Let it be

$$ \varphi = \varphi(\mathfrak{A}, \mathfrak{B}, \ldots, \mathfrak{C}) $$

where $\mathfrak{A}, \mathfrak{B}, \ldots, \mathfrak{C}$ mark the occurring or non-occurring iwffs within φ. Then,

$$ {}_x\mathfrak{R}^{\mathfrak{A}, \mathfrak{B}, \ldots, \mathfrak{C}}_{\alpha, \beta, \ldots, \gamma}\, \varphi(\mathfrak{A}, \mathfrak{B}, \ldots, \mathfrak{C}) | = \{\varphi(\mathfrak{A}, \alpha, \mathfrak{B}, \beta, \ldots, \mathfrak{C}, \gamma)\} $$

As it is seen from the last expression, the operation of replacement results in a set of possible iwffs. ■

We can understand how the operator '=' in the last formula could be replaced also by a particularized operator '$\models$', when the meaning would be that the operation of replacement on the left side of '=' informs the set of possible iwffs on the right side of '='.

II.4.3. Rules of Modus Informationis

II.4.3.0. Introduction

Modus informationis will embrace the broadest realm of informational inferring or of informational syllogism. In this respect, modus informationis will be a kind of observational, investigational, and comprehensional development of information, by means of which a part of arising information will be extracted (recognized, comprehended, and separated) from an existing informational entity (unity). Modus informationis has to be understood also as special, additional (special) mechanism for the development of informational formulae (iwffs), of their arising. In real cases, under modus informationis it will be possible to comprehend any informational arising from an already existing arising of information.

In this section we shall introduce the notion of a suitable class of informational moods or modi of information and its Informing. The goal of this determination will be to get a general, powerful, and indefinitely arising set of transformation rules in the form of iwffs, by which other and also informational modi-concerned iwffs will be transformed from one form to another. The so-called modus of information in our case will be regular information (a concrete iwff) for transforming iwffs. For such a modus we shall introduce the general name modus informationis (MI).

As information (a given iwff for transformation purposes) an informational modus describes the arising of information, which can concern, for instance, Being, existence, state, form, process, structure, organization, etc. of information. Modus is informational property, essence, existence of informational extraction through changing, arising, and vanishing of information, is information of extracting phenomenology and is as such the immanent and regular property of information. A steady, unchangeable modus is similar to an attribute or informationally to a datum. In general sense, modus is information of changing of attributes, which can be understood as informational constants or informationally unchangeable types. Modus is a regular informational process with intention how to extract and by its application to change, generate, develop, or dismiss certain information on which it is applied, how to modify information and enable its arising into new, contrary, richer, poorer, or essentially different information. Informational modus is a general characteristics of information and we use this term to explicate it as a principle, which is relevant to the development, deduction, induction, inference, reasoning, or, generally, to the arising of informational formulae.

II.4.3.1. The Rough Structure of Modus Informationis

What is modus informationis? Modus informationis (MI) means any informationally arising transformation of information. MI is information by itself, is an arising transformational Informing. Let us list some of necessary and possible conclusions:

(1) It is evident that MI as a generalization of the known modi has to preserve the so-called informational transformation by detachment or possibilities of informational extraction of subinformation from a broader informational realm. Thus, MI includes the informational operator of detachment, the most general one, which can be marked by $\models_{\rightarrow}$ or $\models_{\leftarrow}$ or shortly by $\rightarrow$ or $\leftarrow$, respectively. It is to understand that there exists a semantic difference between informational implication ($\Rightarrow$ and $\Leftarrow$) and informational extraction, i.e. detachment ($\rightarrow$ and $\leftarrow$).

(2) What do we have on the antecedent (or "numerator") side of a detachment formula? There are usually several informational components, denoted by variables $\alpha, \beta, \ldots, \gamma$ and connected by an informational operator of the type "$\models_{\text{comma}}$", "$\wedge$", or ",".

(3) On the consequent (or "denominator") side of a detachment formula let it be an informationally simple or complex component marked by δ.

(4) According to paragraphs (1), (2), and (3), the rough structure of MI has the form

$$(\alpha \models_{\wedge} \beta \models_{\wedge} \ldots \models_{\wedge} \gamma) \models_{\rightarrow} \delta$$

(5) How is the consequence δ structurally dependent on the variable (arising) antecedent components $\alpha, \beta, \ldots, \gamma$? What are informational differences among antecedent components? Components $\alpha, \beta, \ldots, \gamma$ are mutually dependent and thus informational differences among them can constitute the nature of the consequence δ.

(6) The general case of MI exposed in paragraphs (1) through (5) can now be particularized and universalized to obtain, for instance, the cases of modus ponens, modus tollens, modus rectus, modus vivendi, etc.

II.4.3.2. Informational Implication

Informational implication, marked by informational operator $\Rightarrow$ and used in several previous definitions, might be viewed as the most primitive form of MI. If information α implies information β, then this fact within IL may sound as a rule, that the occurrence of α within an iwff can be replaced by β. Of course, the notion of informational implication embraces also several forms of the so-called mathematical implications, for instance, the so-called substantial (material), primitive, traditionally logical, effectively logical, effectively true, critical, basic implication, etc.

Further, informational implication as an iwff of the form $\alpha \Rightarrow \beta$ has to be understood as a particularization of the most general formula of Informing $\alpha \models \beta$. However, formula $\alpha \Rightarrow \beta$ has to be understood as universalization of, for instance, known mathematical (logical) forms of implication.

II.4.3.3. The Case of Informational Modus Ponens

Common sense had almost no inkling that physical reality is mathematical. Why would it be better off when it comes to the formal character of cognition?

Radu J. Bogdan [13] 118

Modus ponens concerns, for instance, one of the very elaborated and practiced rule of formula transformation in mathematics. It is the most known modus in mathematical theories. In fact, it is a modus of limited reasoning or strict inference which uses the so-called disjunctive syllogism, where affirming one of given possibilities excludes other possibilities and vice versa. In this section we shall determine various informational possibilities of the so-called informational modus ponens.

[Transformation Rules]^{DF4}:
Let us determine the traditional and most common rule of modus ponens! Let α and β be informational entities and let $\Rightarrow$ be the operator of informational implication. The rule is the following:

$$\frac{\alpha,\ \alpha \Rightarrow \beta}{\beta}$$

To be more precise, this rule can be rewritten as

$$(\alpha \wedge (\alpha \Rightarrow \beta)) \Rightarrow \beta$$

which comes closer to the iwff of IL. But it must be kept in mind that the traditional logic deals with truth and falsity, and so the traditional interpretation of modus ponens within IL would be

$$((((\alpha \models_T) \wedge ((\alpha \Rightarrow \beta) \models_T))\models_T) \Rightarrow (\beta \models_T)) \models_T$$

This formula enables the understanding of the so-called detachment of β (or extraction of β from the antecedent of modus ponens) as a true informational entity within the informational realm of $\alpha \Rightarrow \beta$. ■

The meaning of the last formula is that modus ponens, in its entirety, informs true or that it is by itself a true proposition. The detachment of β means, that β informs true and that on account of this truth it can be recognized as a valid proposition. However, two presumptions must be true, namely, that α informs true and that the formula $\alpha \Rightarrow \beta$ in this particular case informs true (this yields that the conjunction of α and $\alpha \Rightarrow \beta$ informs true too).

Let us now show further possible informational universalization of modus ponens in the last definition! This could be a regular way how from a particular case (traditional modus ponens) a more universal case can be obtained.

[Transformation Rules]^{DF5}:
Let us rewrite the basic formula of modus ponens in the following manner:

(1) $\qquad (\alpha \models_\wedge (\alpha \models_\Rightarrow \beta)) \models_\Rightarrow \beta$

This formula has up to now not been essentially different from the traditional formula. The next step can be its radical universalization by replacing all explicit operators in the formula by the most universal operator $\models$:

(1a) $\qquad (\alpha \models (\alpha \models \beta)) \models \beta$

This formula says that α in some way informs the process $\alpha \models \beta$ and that the entire process $\alpha \models (\alpha \models \beta)$ finally informs β. It means simply that the entire process $\alpha \models (\alpha \models \beta)$ informs one of its components, namely β. This result is a pure consequence of the radical universalization of modus ponens. Simultaneously, this universalization shows the essential point of modus ponens, namely, that no other component than β is informed by the process $\alpha \models (\alpha \models \beta)$ so far. It means that, for instance, α must remain as it is or at least must not be informed by $\alpha \models (\alpha \models \beta)$. This universalization shows evidently the problem which could appear in case of a real, living information where the Informing to α has to be blocked (inhibited) against the Informing of $\alpha \models (\alpha \models \beta)$. This request can be expressed explicitly by the attributed formula (modus)

(1b) $\qquad (\alpha \models (\alpha \models \beta)) \not\models \alpha$ ■

[Transformation Rules]^{DF6}:
As a rule, modus ponens informs true in its details and in its entirety, as shown in [Transformation Rules]^{DF4}. Let us rewrite this rule in the following (postfix) manner:

(2) $\qquad ((((\alpha \models_T) \models_\wedge ((\alpha \models_\Rightarrow \beta) \models_T)) \models_T)$
$\qquad\qquad \models_\Rightarrow (\beta \models_T)) \models_T$

The symmetric (prefix) version of (2) would be

(3) $\qquad \models_T ((\models_T ((\models_T \alpha) \models_\wedge (\models_T (\alpha \models_\Rightarrow \beta))))$
$\qquad\qquad \models_\Rightarrow (\models_T \beta))$

The next step can be a radical universalization of formulae (2) and (3) in the following way:

(2a) $\qquad ((((\alpha \models) \models ((\alpha \models \beta) \models)) \models)$
$\qquad\qquad \models (\beta \models)) \models$

(3a) $\qquad \models ((\models ((\models \alpha) \models (\models (\alpha \models \beta))))$
$\qquad\qquad \models (\models \beta))$

These formulae tell that Informings of α, where α informs ($\alpha \models$) and is informed ($\models \alpha$), inform the Informing of the process $\alpha \models \beta$ and that entire Informings of processes $(\alpha \models) \models ((\alpha \models \beta) \models)$ and $(\models \alpha) \models (\models (\alpha \models \beta))$, respectively, finally inform Informings of β ($\beta \models$ and $\models \beta$, respectively). Similarly to (1b) in the previous definition, the following two formulae can be attributed to (2a) and (3a), respectively:

(2b) $\quad (((\alpha \models) \models ((\alpha \models \beta) \models)) \models) \not\models (\alpha \models)$

(3b) $\quad (\models ((\models \alpha) \models (\models (\alpha \models \beta)))) \not\models (\models \alpha)$ ■

In some cases it could be useful to introduce the so-called extraction (separation, detachment) line to improve the visibility of an informational modus. In modus ponens it would be, for instance,

$$\frac{\alpha \wedge (\alpha \Rightarrow \beta)}{\beta} \quad \text{or} \quad \frac{\alpha \wedge (\alpha \Rightarrow \beta)}{\beta}$$

instead of the traditional expression.

We see how formulae of informational modi are becoming iwffs and can be understood as such. We have to keep in mind that modi are informational rules for transforming other informational formulae. In this respect the meaning of the extraction operation (line of detachment) is, for instance, 'affirms', 'asserts', 'maintains', 'puts_out_to_interest', 'considers', etc. Thus, operation of informational extraction can be understood as an informational particularization.

[Transformation Rules]^{EX1}:
Within informational logic it is possible to construct an infinite set of informational modi ponens. Let us list some characteristic examples! The first example is, for instance,

the modus ponens of belief, where $\models_B$ is the informational operator of believing. There is:

$$\frac{\models_B \alpha,\ \models_B (\alpha \Rightarrow \beta)}{\models_B \beta}$$

This rule says: if α is believed and if $\alpha \Rightarrow \beta$ is believed, then β is believed. To be consequent to resulting from our believing, we have to attribute to this formal believing implicitly the following:

$$\models_B (\models_B \alpha,\ \models_B (\alpha \Rightarrow \beta))$$

and

$$\models_B (\models_B (\models_B \alpha,\ \models_B (\alpha \Rightarrow \beta)) \ / \ (\models_B \beta))$$

We certainly have to believe the entire antecedent as it is composed and we have to believe in modus ponens (of believing). Informational operator '/' was introduced to replace the usual detachment operation.

A similar example can be constructed for the case of knowledge, where

$$\frac{\models_K \alpha,\ \models_K (\alpha \Rightarrow \beta)}{\models_K \beta}$$

etc. However, we can still put the question what would the so-called modus ponens of Informing be. ■

II.4.3.4. The Case of Informational Modus Tollens

Without a clear teleological hold on distal targets, and a clarification of what this means, we might only get proximal semantics, and we do not want that. For if proximal semantics makes sense, then my entire approach to semantic information doesn't. Hence the urgent need for modus tollens.

Radu J. Bogdan [13] 100

In general, the modus tollens invalidates, negates, or informationally abolishes a piece of complex information and, in this respect, represents an informational transformation which can be understood as, in some sense, opposite to informational transformation by modus ponens. Of course, modus tollens can be used in traditional theories as a rule of negation. In fact it is a modus of limited reasoning or strict inference which uses the so-called hypothetical syllogism: negating the consequent causes negation of the antecedent.

[Transformation Rules]^{DF7}:
First, let us define the traditional modus tollens! Let α and β be informational entities, $\Rightarrow$ the operator of informational implication, and $\neg$ the symbol of logical negation. By these terms, the rule of traditional modus tollens is the following:

$$\frac{\alpha \Rightarrow \beta,\ \neg \beta}{\neg \alpha}$$

This rule can be logically rewritten into

$$((\alpha \Rightarrow \beta) \wedge (\neg \beta)) \Rightarrow (\neg \alpha)$$

and represents an iwff of IL. However, there is a slight difference when comparing modus ponens and modus tollens, due the appearance of operator $\neg$. Thus, instead of the first interpretation of modus tollens by the formula of detachment, it could be also

$$\frac{\alpha \Rightarrow \beta,\ \beta \neg}{\neg \alpha}$$

This is due to $\beta \neg \alpha$, where the meaning of $\neg$ is the following:

$$\neg =_{Df} (\text{'negates'} \vee \text{'negate'} \vee \text{'is\_negated\_(by)'} \vee \text{'are\_negated\_(by)'})$$

By modus tollens the consequent negates the antecedent.

In terms of traditional logic, modus tollens has to be understood through categories of truth and falsity (at least of some parts of the formula). Thus, a traditional interpretation of modus tollens becomes

$$(((((\alpha \Rightarrow \beta) \models_T) \wedge ((\beta \neg) \models_T)) \models_T) \Rightarrow ((\neg \alpha) \models_T)) \models_T$$

This formula gives the detachment $(\neg \alpha) \models_T$ out of the premise of modus tollens. But, in a certain case, it is possible to explicate the non-informing nature of components which bear the operation of negation $\neg$, for instance

$$(((((\alpha \Rightarrow \beta) \models_T) \wedge (\beta \not\models_T)) \models_T) \Rightarrow (\not\models_T \alpha)) \models_T$$

We have only combined $\neg$ and $\models_T$ into a universal operator $\not\models_T$ which can again be particularized for a certain case. ■

The meaning of the last formula is that modus tollens, in its entirety, informs true. The detachment of $\alpha \not\models_T$ means that α does not inform true. Prior to this, two presumptions have to be true, namely that $\alpha \Rightarrow \beta$ informs true and that β does not inform true.

Now, it is possible to show further informational universalization of modus tollens. Similar to the [Transformation Rules]^{DF5} we can construct the following rule:

[Transformation Rules]^{DF8}:
Let us rewrite the basic formula of modus tollens in the following manner:

(1) $((\alpha \models_{\Rightarrow} \beta) \models_{\wedge} (\beta \neg)) \models_{\Rightarrow} (\neg \alpha)$

This formula of modus tollens has up to now not been essentially different from the traditional formula. The next step of its modification can be its radical universalization by the replacement of all particularized explicit operators in the formula by the most universal operators $\models$ and $\not\models$:

(1a) $((\alpha \models \beta) \models (\beta \not\models)) \models (\not\models \alpha)$

This formula tells that the process $\alpha \models \beta$ informs, in some way, the process $\beta \not\models$ and that the entire process $(\alpha \models \beta) \models (\beta \not\models)$ informs the process $\not\models \alpha$ which concerns one of the components of the process $\alpha \models \beta$, namely, α. This result is a pure consequence of the radical universalization of traditional modus tollens. Simultaneously, this universalization shows the essential point of modus tollens, namely, that no other component than the process $\not\models \alpha$ is informed by the process $(\alpha \models \beta) \models (\beta \not\models)$. This universalization shows the problem which arises in case of a real, living information, where the Informing to $\beta \not\models$ has to be blocked (inhibited) against the Informing of the process $(\alpha \models \beta) \models (\beta \not\models)$. This request can be expressed explicitly by the attributed formula (modus)

$$\text{(1b)} \qquad ((\alpha \models \beta) \models (\beta \not\models)) \not\models (\beta \not\models) \qquad \blacksquare$$

[Transformation Rules]^{DF9}:
As a rule, modus tollens informs true in its details and in its entirety, as shown in [Transformation Rules]^{DF7}. This is a fact which roots in the usual true-false categorization of the traditional logic. Let us rewrite this rule in the following (postfix) manner:

$$\text{(2)} \quad (((((\alpha \models_{\Rightarrow} \beta) \models_T) \models_\wedge ((\beta \models_\neg) \models_T)) \models_T) \\ \models_{\Rightarrow} ((\models_\neg \alpha) \models_T)) \models_T$$

The symmetric (prefix) version of (2) is

$$\text{(3)} \quad \models_T (\models_T ((\models_T (\alpha \models_{\Rightarrow} \beta)) \models_\wedge (\models_T (\beta \models_\neg))) \\ \models_{\Rightarrow} (\models_T (\models_\neg \alpha)))$$

The next step can be a radical universalization of formulae (2) and (3) in the following way:

$$\text{(2a)} \quad (((((\alpha \models \beta) \models) \models ((\beta \not\models) \models)) \models) \\ \models ((\not\models \alpha) \models)) \models$$

$$\text{(3a)} \quad \models (\models ((\models (\alpha \models \beta)) \models (\models (\beta \not\models))) \\ \models (\models (\not\models \alpha)))$$

These formulae tell that Informings of the process $\alpha \models \beta$, where $\alpha \models \beta$ informs $((\alpha \models \beta) \models)$ and is informed $(\models (\alpha \models \beta))$, inform the Informing of the process $\beta \not\models$ and that the entire Informings of processes $(((\alpha \models \beta) \models) \models ((\beta \not\models) \models)) \models$ and $\models ((\models (\alpha \models \beta)) \models (\models (\beta \not\models)))$, respectively, finally inform Informings $(\not\models \alpha) \models$ and $\models (\not\models \alpha)$, respectively. The first of these integral informational entities informs and the second is informed. Similarly to (1b) in the previous definition, the following two formulae can be attributed to (2a) and (3a), respectively:

$$\text{(2b)} \quad ((((\alpha \models \beta) \models) \models ((\beta \not\models) \models)) \models) \not\models \\ ((\alpha \models \beta) \models)$$

$$\text{(3b)} \quad (\models ((\models (\alpha \models \beta)) \models (\models (\beta \not\models))) \not\models \\ (\models (\alpha \models \beta)) \qquad \blacksquare$$

We have to mention again that operators $\models$ and $\not\models$ can be non-uniformly replaced by particularized operators and that operators of the type $\not\models$ can be understood as any informational operators of particular non-Informing. Thus, $\models$ and $\not\models$ are in general not operators which exclude exactly each other, but have to be understood as operational variables belonging to various particular classes.

Instead of the traditional expression of modus tollens we can use also expressions

$$\begin{array}{c}(\alpha \Rightarrow \beta),\ \beta \neg \\ \xRightarrow{\hspace{6em}} \\ \neg\,\alpha\end{array} \qquad \text{or} \qquad \begin{array}{c}(\alpha \Rightarrow \beta),\ \beta \neg \\ \models\!\!=\!\!=\!\!=\!\!=\!\!= \\ \neg\,\alpha\end{array}$$

Expressions of these kind explicate clearly the extraction or detachment operation, which in the context of modus tollens can be particularized (in the second case) or universalized (in the first case).

[Transformation Rules]^{EX2}:
Within IL we can construct an infinite set of informational modi tollens. Firstly, this infiniteness follows from the unforeseeable possibilities of particularization and universalization of appearing informational operators in a formula (iwff) representing modus tollens. Secondly, as we have learned from several previous cases, a distinct formula of modus tollens can be developed through consideration (introducing) of various forms of Informings of operand variables and processes. This procedure of formula development can lead to a more and more complex expression and the stopping of complexness can be impacted by distinct circumstances (semantics, modus vivendi) in the phase of formula development. Let us look at some of these possibilities.

The first two examples are, for instance, the modi tollens of belief, where $\models_B$ and $\not\models_B$ are informational operators of believing and non-believing. There is:

$$\frac{\models_B (\alpha \Rightarrow \beta),\ \models_B (\neg\, \beta)}{\models_B (\neg\, \alpha)} \quad \text{and} \quad \frac{\models_B (\alpha \Rightarrow \beta),\ \models_B (\beta\, \neg)}{\models_B (\neg\, \alpha)}$$

or also

$$\frac{(\alpha \Rightarrow \beta) \models_B,\ (\neg\, \beta) \models_B}{(\neg\, \alpha) \models_B} \quad \text{and} \quad \frac{(\alpha \Rightarrow \beta) \models_B,\ (\beta\, \neg) \models_B}{(\neg\, \alpha) \models_B}$$

The first rule says: if it is believed that α implies β and if it is believed that β is negated, then it is believed that α is negated. The second rule says: if it is believed that α implies β and if it is believed that β negates, then it is believed that α is negated (informationally in an implicit manner by β). The third rule says: if information 'α implies β' believes (or is believable) and if information 'β is negated' believes (is believable), then information 'α is negated' believes (is believable). The fourth rule says: if information 'α implies β' believes (or is believable) and if information 'β negates' believes (is believable), then information 'α is negated' (informationally in an implicit manner by β) believes (is believable).

Similarly to [Transformation Rules]^{EX1} it is possible to express the belief into modus tollens for the upper four cases in the following way:

$$\models_B ((\models_B ((\models_B (\alpha \Rightarrow \beta)) \wedge (\models_B (\neg\, \beta)))) \,/ \\ (\models_B (\neg\, \alpha)))$$

$$\models_B ((\models_B ((\models_B (\alpha \Rightarrow \beta)) \wedge (\models_B (\beta \neg)))) \; / \; (\models_B (\neg \alpha)))$$

$$((((\alpha \Rightarrow \beta) \models_B) \wedge ((\neg \beta) \models_B)) \models_B) \; / \; ((\neg \alpha) \models_B)) \models_B$$

$$((((\alpha \Rightarrow \beta) \models_B) \wedge ((\beta \neg) \models_B)) \models_B) \; / \; ((\neg \alpha) \models_B)) \models_B$$

We certainly have to believe the entire antecedents as they are composed (by the operators $\wedge$) and we have to believe the upper rules of modus tollens. Informational operator '/' replaces the usual operation of detachment. ■

[Transformation Rules]^{EX3}:
The next two examples of modus tollens we are going to examine concern knowledge and awareness. The traditional form of modus tollens of knowledge is, for instance,

$$\frac{\models_K (\alpha \Rightarrow \beta),\ \models_K (\neg \beta)}{\models_K (\neg \alpha)}$$

This formula has the following meaning: if it is known that α implies β and if it is known that β is negated, then it is known that α is negated. However, we can interpret the operator $\not\models_K$ as 'it_is_not_known' or 'it_does_not know'. Thus, the basic formula of modus tollens of knowledge can be rewritten into the form

$$\frac{\models_K (\alpha \Rightarrow \beta),\ \not\models_K \beta}{\not\models_K \alpha}$$

The meaning of this formula is the following: if it is known that α implies β and if β is not known, then α is also not known. As $\models_K$ and $\not\models_K$ can be particularized in a non-uniform way, the meaning of the operator variable can cover a broad informational realm, which might not have any relation to the opposition of a particular operator belonging to the type $\not\models_K$.

A similar reasoning is possible in case of the so-called awareness ($\models_A$) and unawareness ($\not\models_A$). The traditional form of modus tollens of awareness is

$$\frac{\models_A (\alpha \Rightarrow \beta),\ \models_A (\neg \beta)}{\models_A (\neg \alpha)}$$

Let us interpret the meaning of this formula: if 'it is aware' (= 'it is consciously evident') that α implies β and if 'it is aware' that β is negated, then 'it is aware' that α is negated. The awareness of $\neg \beta$ and $\neg \alpha$ can in a particular case be interpreted as unawareness of β and α, respectively. In this case, from the awareness that α implies β and that β is unaware follows that α is unaware. Thus, formula

$$\frac{\models_A (\alpha \Rightarrow \beta),\ \not\models_A \beta}{\not\models_A \alpha}$$

sounds quite reasonably. ■

In the following examples we shall examine the informational connectedness of truth, belief, knowledge, awareness, and their counterparts (for instance: falsity, doubt, illiteracy, unconsciousness).

[Transformation Rules]^{EX4}:
In the previous example we could recognize some semantic similarity existing among informational processes concerning truth, belief, knowledge, and awareness. For instance, in the case of the definition of information α,

$$(\text{'}\alpha \text{ is\_information'}) =_{Df} ((\alpha \models) \vee (\models \alpha) \vee (\Dashv \alpha) \vee (\alpha \Dashv))$$

it is possible, in a concrete case, to particularize this definition in a non-uniform manner into

$$(\alpha \models_T) \vee (\models_B \alpha) \vee (\Dashv_K \alpha) \vee (\alpha \Dashv_A)$$

or, for instance, expressing it in the form of a parallel metaphysical system

$$\mu \Vdash_T,\ \Vdash_B \mu,\ \dashV_K \mu,\ \mu \dashV_A$$

This could be a natural parallel metaphysical process in which informational cooperation of truth, belief, knowledge, and awareness is coming into existence. Certainly, this can occur not only in the cases in which transformations of modus tollens are taking part.

Within the domain of modus tollens it was possible to observe operational combinations (concatenations) concerning operators of Informing and non-Informing. We can explain the following examples:

$\models_T (\models_B \alpha)$	'it is informed true' that α is believed;
$\models_T (\not\models_B \alpha)$	'it is informed true' that α is not believed;
$\not\models_T (\models_B \alpha)$	'it is not informed true' that α is believed;
$\not\models_T (\not\models_B \alpha)$	'it is not informed true' that α is not believed;
$\models_B (\models_T \alpha)$	it is believed that α $\downarrow$is informed true';
$\models_B (\not\models_T \alpha)$	it is believed that α $\downarrow$is not informed true';
$\not\models_B (\models_T \alpha)$	it is not believed that α $\downarrow$is informed true';
$\not\models_B (\not\models_T \alpha)$	it is not believed that α $\downarrow$is not informed true'

Some operationally split cases can be of particular interest. For instance,

$(\models_T \alpha) \models_B$	α 'is informed true' informs believable;
$(\models_T \alpha) \not\models_B$	α 'is informed true' does not inform believable;
$(\not\models_T \alpha) \models_B$	α 'is not informed true' informs believable;
$(\not\models_T \alpha) \not\models_B$	α 'is not informed true' does not inform believable;
$(\alpha \models_B) \models_T$	α informs believable informs true;
$(\alpha \models_B) \not\models_T$	α informs believable does not inform true;
$(\alpha \not\models_B) \models_T$	α informs unbelievable informs true;

$(\alpha \not\models_B) \not\models_T$ α informs unbelievable does *not inform true*

Etc. We can see how particular cases can be operationally reduced. If information informs believable and true, then it can be reduced to inform simply true or simply believable. For instance,

$$\models_T (\models_B \alpha),\ \models_B (\models_T \alpha),\ (\models_T \alpha) \models_B,\ (\alpha \models_B) \models_T$$

could be reduced either into

$$\models_T \alpha \text{ and } \models_B \alpha \text{ or into } \alpha \models_T \text{ and } \alpha \models_B$$

As soon as we have an operator which informs in an untrue or unbelievable manner, a combination of "concatenated" or split operators can be reduced to inform simply untrue or simply unbelievable. For instance, formulae of the above cases

$$\models_T (\not\models_B \alpha),\ \not\models_T (\models_B \alpha),\ \models_B (\not\models_T \alpha),\ \not\models_B (\models_T \alpha),$$
$$(\models_T \alpha) \not\models_B,\ (\not\models_T \alpha) \models_B,\ (\alpha \models_B) \not\models_T,\ (\alpha \not\models_B) \models_T$$

could be reduced either into

$$\not\models_T \alpha \text{ and } \not\models_B \cdot\alpha \text{ or into } \alpha \not\models_T \text{ and } \alpha \not\models_B$$

In cases, where operators inform simultaneously untrue and unbelievable, i.e.,

$$\not\models_T (\not\models_B \alpha),\ \not\models_B (\not\models_T \alpha),\ (\not\models_T \alpha) \not\models_B,\ (\alpha \not\models_B) \not\models_T$$

it is not possible to get a senseful operational reduction. As we can understand, in some particular cases, rules for operational reduction can be constructed. ■

[Transformation Rules]^{EX5}:
We can show how sequences of informational operators can be reduced into a single operator. For instance, if information α is informed aware, known, believable, and true, it can be reduced in the following way:

$$(\models_A (\models_K (\models_B (\models_T \alpha)))) \Rightarrow$$
$$(((\models_A \alpha) \vee (\models_K \alpha) \vee (\models_B \alpha) \vee (\models_T \alpha))$$

The antecedent part of this formula is to be read as follows: it is informed aware that it is informed known that it is informed believable that α is informed true. The shorter meaning would be: α is informed aware, known, believable, and true. Within this example it is possible to recognize the common informational circularity of awareness, knowledge, belief, and truth.

A similar informational phenomenon appears also when such an operator sequence is split. For instance:

$\models_K (\models_T (\alpha \models_B))$	it is known that it is informed true that α informs believable;
$\models_T ((\alpha \models_K) \models_B)$	it is informed true that α informs known informs believable;

etc. Truth, belief, and knowledge are informational entities (processes) which in the realm of living belong to the awareness within a being's metaphysics. ■

[Transformation Rules]^{EX6}:
Now let us examine some contrary operations to truth, belief, knowledge, and awareness. informational entities which inform and are informed in this sense are

$(\alpha \models_T) \vee (\models_T \alpha)$	for truth,
$(\alpha \models_B) \vee (\models_B \alpha)$	for belief,
$(\alpha \models_K) \vee (\models_K \alpha)$	for knowledge, and
$(\alpha \models_A) \vee (\models_A \alpha)$	for awareness

In a similar way it is possible to introduce the contraries of these informational entities, denoting them as

$(\alpha \not\models_T) \vee (\not\models_T \alpha)$	for untruth,
$(\alpha \not\models_B) \vee (\not\models_B \alpha)$	for unbelief,
$(\alpha \not\models_K) \vee (\not\models_K \alpha)$	for ignorance, and
$(\alpha \not\models_A) \vee (\not\models_A \alpha)$	for unawareness

How is it possible to determine subclasses to these informational entities? Let us introduce falsity, doubt, illiteracy, and unconsciousness as particular contraries to truth, belief, knowledge, and awareness:

$(\beta \models_F) \vee (\models_F \beta)$	for falsity,
$(\beta \models_D) \vee (\models_D \beta)$	for doubt,
$(\beta \models_I) \vee (\models_I \beta)$	for illiteracy, and
$(\beta \models_U) \vee (\models_U \beta)$	for unconsciousness

It is probably possible to construct relation of the so-called subinformation (operator ⊂) between falsity and untruth, doubt and unbelief, illiteracy and ignorance, and unconsciousness and unawareness. Thus,

$$(\beta \models_F) \subset (\alpha \not\models_T),\ (\models_F \beta) \subset (\not\models_T \alpha),$$
$$(\beta \models_D) \subset (\alpha \not\models_B),\ (\models_D \beta) \subset (\not\models_B \alpha),$$
$$(\beta \models_I) \subset (\alpha \not\models_K),\ (\models_I \beta) \subset (\not\models_K \alpha),$$
$$(\beta \models_U) \subset (\alpha \not\models_A),\ (\models_U \beta) \subset (\not\models_A \alpha)$$

This example shows the informational power of operator $\not\models$, which can embrace quite a substantial realm of contrary information. ■

II.4.3.5. The Case of Informational Modus Rectus

I wish to examine the concept of a system whose behavior can be - at least sometimes - explained and predicted by relying on ascriptions to the system of beliefs and desires (and hopes, fears, intentions, hunches, ...). I will call such systems intentional systems, and such explanations and predictions intentional explanations and predictions, in virtue of the intentionality of the idioms of belief and desire (and hope, fear, intention, hunch, ...).

Daniel C. Dennett [14] 220

In Latin, rectus means something erect, right, proper, appropriate, suitable, intelligent, natural, etc. Informational modus rectus (IMR) will concern direct adjustment (setting, ruling, intentionality) of some experienced (occurred) informational subjectiveness and/or objectiveness. Informationally, IMR concerns informational forms and processes in the realm of belief, desire, intention, etc. being embedded into a living being's metaphysics and within it informationally impacting a living being's behavioral information. In short, IMR concerns belief, desire, intention, etc. and their informational transformation within metaphysical and especially behavioral information. Within these informational circumstances it seems to be worth to examine the nature of the so-called intentional information or intentionality which would be the central notion in connection with the nature of IMR.

Intention is a determination to act in a certain way. Intention is oriented information (i.e. acts in a certain direction). In this sense, information as phenomenology of the living is intentional in general and has its intentionality being impacted by the previous arising of information as information concerning information. Further, intention of information means that certain informational entities within information intend to be more important or significant for the arising of information than others and that they intend to have various impact on their own informational arising.

Informational intentionality means that some information about certain information is arising, thus, that this intentionality concerns the so-called aboutness of certain information. Such an informational aboutness can be a kind of observation, investigation, and comprehension as information of a certain information. Informational intentionality is a particular form or process of counter-information and counter-Informing, which arise within information.

Particular cases of informational intentionality can be clearly informationally distinguished. What are, for instance, beliefs, hopes, cares, hunches, plans, goals, suspicions, knowledge, truth, etc. other than intentional forms of information? Do they impact a being's metaphysics and its behavior? The answer to such questions is by itself a form of intentional information. This means simply that intention of information is its arising, changing, and vanishing during the life cycle of information. The informational modus rectus takes intention as an essential rule or ruling information, which concerns informational transformation not only on the level of living information, but in the case of informational logic also on the level of transformation of iwffs.

[Operands]$^{\mathrm{DF5}}$:
Let us have the following definition:

$$(\text{'}\alpha\ \text{is\_intentional\_information'}) =_{\mathrm{Df}} ((\alpha \models_{\mathfrak{I}}) \vee (\models_{\mathfrak{I}} \alpha))$$

where $\mathfrak{I}$ is intentional Informing of α (hidden in α), so that $\mathfrak{I} \subset \alpha$. As α is information, for which

$$(\text{'}\alpha\ \text{is\_information'}) =_{\mathrm{Df}} ((\alpha \models) \vee (\models \alpha))$$

there is, in the case of intentional information,

$$((\alpha \models_{\mathfrak{I}}) \subset (\alpha \models)) \vee ((\models_{\mathfrak{I}} \alpha) \subset (\models \alpha))$$

In these formulae $\mathfrak{I}$ is the informing (or informationally active) component of information α. ■

[Transformation Rules]$^{\mathrm{DF10}}$:
What could be the transformation formulae of modus rectus? One of the possible ways is to proceed from the notion of intention or intentionality. On this way we have to develop an initial philosophy.

Let α be intentional information which hides some intention $\mathfrak{I}$ as an informing part of information α. Intention $\mathfrak{I}$ is a part of α's Informing. Let intentional information α act (inform) upon information β, so, $\alpha \models_{\mathfrak{I}} \beta$. Now, modus rectus is the rule which separates (detaches, reveals) the intention $\mathfrak{I}$ as particular information which informs intentionally within α or is a form of Informing of intentionality α. Thus, the traditional modus rectus can be expressed as

$$\frac{\alpha,\ \alpha \models_{\mathfrak{I}} \beta}{\mathfrak{I}}$$

where α is intentional information and $\mathfrak{I}$ its intention (as information). This formula can be rewritten in a logical manner as

$$\frac{\models_{\mathrm{T}} ((\alpha \models_{\mathfrak{I}}) \vee (\models_{\mathfrak{I}} \alpha)),\ \models_{\mathrm{T}} (\alpha \models_{\mathfrak{I}} \beta)}{\models_{\mathrm{T}} (\mathfrak{I} \subset \alpha,\ \beta)}$$

or in a logically more complete form

$$\models_{\mathrm{T}} ((\models_{\mathrm{T}} ((\models_{\mathrm{T}} ((\alpha \models_{\mathfrak{I}}) \vee (\models_{\mathfrak{I}} \alpha))) \wedge (\models_{\mathrm{T}} (\alpha \models_{\mathfrak{I}} \beta)))) \ / \ (\models_{\mathrm{T}} (\mathfrak{I} \subset \alpha,\ \beta)))$$

where '/' is the detachment operation. There are traces of the intention $\mathfrak{I}$ in α, as well as in β. Information β arises as a consequence of intention $\mathfrak{I}$ within α, which intentionally informs β. The last two formulae enable understanding of the so-called detachment of $\mathfrak{I}$ (or extraction of $\mathfrak{I}$ from the antecedent of modus rectus) as a true informational entity within the informational realm of $\alpha \models_{\mathfrak{I}} \beta$. ■

II.4.3.6. The Case of Informational Modus Obliquus

In Latin, obliquus means slanting, sideways, oblique, indirect, covert, and also envious. Informational modus obliquus (IMO) will concern indirect adjustment (a peculiar or personal point of view, attitude, or opinion) of an absurdly (and individually) experienced informational subjectiveness and/or objectiveness. In this respect, within IMO also a line with a special (oblique) interest will be interpreted or presented. We can say that IMO as an informational transformation is applied from one (specific) side, also with disapproval or distrust. Informationally, IMO concerns informational forms and processes in the realm of unawareness, illiteracy, doubt, and falsity. If modus rectus was a transformation rule in the sense of directness or intentionality, then modus obliquus will be a transformation rule in the sense of indirectness or absurdity.

As a form of indirect rule, modus obliquus deviates from a direct or intentional line of discourse, performing roundabout or not going straight to the point. As an indirect proof, it involves proof of informational entities that negation leads to an absurdity or contradiction. In this manner IMO reveals information which is not openly shown or is to some degree secret.

[Transformation Rules]$^{\mathrm{DF11}}$:
Let α be an absurd or contradictory information defined as

$$(\text{'}\alpha\ \text{is\_absurd\_information'}) =_{\mathrm{Df}} ((\alpha \models_{\mathfrak{A}}) \vee (\models_{\mathfrak{A}} \alpha)).$$

where $\mathfrak{A}$ is absurdity as information or Informing of information as absurdity. Let τ be information for which it is believed that it informs true ($\models_{B} (\tau \models_{T})$). Then, the rough or traditional form of modus obliquus could be

$$\frac{\tau,\ (\neg\, \tau) \Rightarrow \alpha}{\neg\, \tau}$$

or, more precisely,

$$\frac{\models_{B} (\tau \models_{T}),\ (\neg\, \tau) \Rightarrow \alpha}{\tau \not\models_{T}}$$

We see how in this case it is meaningful to explicate the belief of the true Informing of τ at the beginning of the process of IMO. The last formula of IMO is read in the following way: if it is believed that information τ informs true and if the negation of information τ implies an absurd informational entity α, then τ does not inform true. In this case, the implication of absurdity by negation of τ causes an untrue Informing of τ. The last formula can be rewritten in a logically complete iwff:

$$\begin{gathered}\models_{T} (\models_{T} ((\models_{T} ((\models_{B} (((\tau \models) \vee (\models \tau)) \models_{T})))) \wedge \\ (\models_{T} ((\neg\, \tau) \Rightarrow ((\alpha \models_{\mathfrak{A}}) \vee (\models_{\mathfrak{A}} \alpha))))) \\ /\ (\models_{T} (((\tau \models) \vee (\models \tau)))))\end{gathered}$$

where '/'is the operator of detachment. ■

II.4.3.7. The Case of Informational Modus Procedendi

Informational modus procedendi is a mood of informational detachment by which a goal information is coming into the process of Informing. The Latin procedo has the meaning of to go forth or before, advance, make progress; to continue, remain; and to go on. When informationally proceeding, the process has to go forward by showing the goal in advance. As an informational process, modus procedendi runs on according to a goal information, where this goal information informs, for instance, a motor, behavioral, or simply an acting information and, finally, when the goal is exhausted, elapses.

There exist an infinite number of possibilities how to structure and organize goal-directed informational systems. The task of a modus procedendi could be, for instance, how to extract a goal structure and organization from a complex living or artificial informational system, to bring this goal informational structure and organization to the surface, for instance to the logical or conscious level. This could be a senseful informational process of hidden informational goals identification and their use in various life and technological strategies.

[Transformation Rules]$^{\mathrm{DF12}}$:
Let γ be a goal information, where $\models_{\mathfrak{C}}$ is its goal Informing. Now, let us have the following definition of a goal operand variable:

$$(\text{'}\gamma\ \text{is\_goal-expressing\_information'}) =_{\mathrm{Df}} ((\gamma \models_{\mathfrak{C}}) \vee (\models_{\mathfrak{C}} \gamma))$$

Let α be information (for instance, motor or behavioral operand variable) which must approach or at least consider the goal information, or, as we usually say, must be informed by γ. We can conclude that in some informational elements α has to become informationally similar to γ, thus, $\alpha \therefore_{\mathfrak{C}} \gamma$. This expression is read as follows: α becomes goal-similar to γ. Under this circumstances α is information approaching to the goal and γ is information which informs α. We can now express informational modus procendendi (IMPr) in the following, traditional form:

$$\frac{\mathfrak{C},\ \gamma \models_{\mathfrak{C}} \alpha}{\alpha \therefore_{\mathfrak{C}} \gamma}$$

Let us analyze this informational modus! The essential informational entity of the consequent is the operator $\therefore_{\mathfrak{C}}$. This operator has to answer the question, how much has α already approached γ. In this way, modus procedendi has extracted the relation of informational similarity between α and γ. In the antecedent, γ does not arbitrarily inform α, but it has to inform α particularly by the structure and organization of $\mathfrak{C}$. In this respect modus procedendi seems to be much more complex than the previous modi have been. It evidently concerns some parts of Informing of γ (the antecedent of IMPr) as well as of α (the consequent of IMPr).

II.4.3.8. The Case of Informational Modus Operandi

The reason such an internal selectivity is a major condition on semantic information is that a tokened information structure counts as semantic only if its shape and function in a system can be explained, under appropriate types of regularities, relative to some distal properties. The information structure must therefore be shaped inside the system, by its architecture and modus operandi, in ways which can be explained only by appeal to semantic considerations.

Radu J. Bogdan [13] 98

In Latin, modus operandi means a method of operating or proceeding. This meaning comes near to the concept of algorithm, which is a method of procedure. Evidently, the informational modus operandi (IMOp) has to answer the question what is the aim or essence of informational operation within an informational complex or what is the subject of operation. Thus, IMOp has to extract the operational information, and in regard to this it has to explicate the Informing of information which, in general, informs and is informed. Informational modus operandi reveals the nature of Informing of information. By this explication it becomes informationally known how a certain information informs and is informed. IMOp discovers the informing of information and, in this respect, it is an informational tool for the identification of Informing.

How does an information function? How does it produce informational effects on itself and on informationally involved information? How does it arise and how does it cause arising of other information? How are this informational effects particularized? Informational modus operandi delivers answers to this questions in the form of its consequent. The task of IMOp is, for instance, to discover the algorithm of data processing. However, information cannot be reduced to data, which are static informational entities, which are a collection of operative and informative data. The question is what puts and keeps information in its operation. What are operational operators as concerned their informational structure and organization?

[Transformation Rules]^{DF13}:
What is Informing $\mathfrak{I}$ (or $\mathfrak{I}_\alpha$) of information α? Informing $\mathfrak{I}$ is nothing else but an informational functionality $\mathfrak{F}$ of α, thus,

$$\mathfrak{I} = \mathfrak{F}(\alpha)$$

In this sense, Informing $\mathfrak{I}$ is an implicit informational operator of α which is a product of α and which as an active part of information produces α. In this respect, the basic definition of information α can be expressed also as

$$(\text{'}\alpha\text{ is\_information'}) =_{\mathrm{Df}} ((\alpha \models_{\mathfrak{F}(\alpha)}) \vee (\models_{\mathfrak{F}(\alpha)} \alpha))$$

This would have the meaning that α informs and is informed in virtue of its own functionality.

Informing $\mathfrak{I}$ of information α means that information α counter-informs itself and that it embeds the produced counter-information. This is the known principle of informational cyclicity. Within this philosophy, counter-Informing of α, denoted as $\mathfrak{C} = \mathfrak{C}(\alpha)$, and informational embedding $\mathfrak{E} = \mathfrak{E}(\alpha)$ of the counter-informed counter-information γ are sub-Informings of $\mathfrak{I}$, thus,

$$\mathfrak{C}, \mathfrak{E} \subset \mathfrak{I}$$

When discovering $\mathfrak{I}$, informational modus operandi has to reveal components $\mathfrak{C}$ and $\mathfrak{E}$ of Informing $\mathfrak{I}$ to answer the question about the nature of α's Informing. In this procedure IMOp asks for the cyclic structure and organization of information α. The most simple form of IMOp in the case of α's self-Informing is

$$\frac{\alpha,\ \alpha \models_{\mathfrak{I}} \alpha}{\mathfrak{C},\ \mathfrak{E} \subset \mathfrak{I}}$$

The cyclic complexity of α's cyclic parallel Informing $\mathfrak{I}$, considering its counter-Informing $\mathfrak{C}$ and informational embedding $\mathfrak{E}$, can be chosen as follows:

$$\alpha \Vdash_{\alpha} \mathfrak{I},\quad \alpha, \mathfrak{I} \Vdash_{\mathfrak{I}} \mathfrak{I}, \alpha,\quad \alpha, \mathfrak{I} \Vdash_{\mathfrak{I}} \mathfrak{C},$$
$$\alpha \Vdash_{\mathfrak{C}} \gamma,\quad \alpha, \mathfrak{I}, \gamma \Vdash_{\alpha} \mathfrak{E},\quad \gamma \Vdash_{\mathfrak{E}} \alpha$$

IMOp has to explore this cyclic informational domain since $\mathfrak{I}$ can be identified considering also its instantaneous components $\mathfrak{C}$ and $\mathfrak{E}$. ■

[Transformation Rules]^{DF14}:
The next cases, which are much more complex than the previous one, concern the question how does information α inform other information β, i.e., $\alpha \models \beta$ and, in general, $\alpha \models \xi, \eta, \ldots, \zeta$. If the previous rules concerned self-Informing, the subsequent ones will concern one-way inter-Informing of information. Let us introduce the following tokens:

$\mathfrak{I}_\alpha$ or $\mathfrak{I}(\alpha)$ will mark the Informing of information α;
$\mathfrak{C}_\alpha$ or $\mathfrak{C}(\alpha)$ will mark the counter-Informing of information α; and
$\mathfrak{E}_\alpha$ or $\mathfrak{E}(\alpha)$ will mark the informational embedding of information α

In the case of the one-way Informing of information α to information β, the following rule of informational modus operandi can be constructed:

$$\frac{\alpha, \beta;\ \alpha \models_{\mathfrak{I}(\alpha)} \beta}{\mathfrak{C}_\alpha, \mathfrak{E}_\alpha \subset \mathfrak{I}_\alpha;\ \mathfrak{C}_\beta, \mathfrak{E}_\beta \subset \mathfrak{I}_\beta}$$

This form of IMOp has to consider the following complexities: the cyclic complexity of α's cyclic parallel Informing $\mathfrak{I}_\alpha$ (coming into existence in virtue of $\alpha \models \alpha$), considering the counter-Informing $\mathfrak{C}_\alpha$ and informational embedding $\mathfrak{E}_\alpha$; the one-way complexity of α's (linear) parallel Informing of β (coming into existence in virtue of $\alpha \models \beta$); and the cyclic complexity of β's cyclic parallel Informing $\mathfrak{I}_\beta$ (coming into existence in virtue of $\beta \models \beta$), considering the counter-Informing $\mathfrak{C}_\beta$ and informational embedding $\mathfrak{E}_\beta$. This complexity can be expressed by the following parallel system:

$$\alpha \Vdash_{\alpha} \mathfrak{I}(\alpha);\ \alpha, \mathfrak{I}(\alpha) \Vdash_{\mathfrak{I}(\alpha)} \mathfrak{I}(\alpha), \alpha;$$
$$\alpha, \mathfrak{I}(\alpha) \Vdash_{\mathfrak{I}(\alpha)} \mathfrak{C}(\alpha);$$
$$\alpha \Vdash_{\mathfrak{C}(\alpha)} \gamma_\alpha;\ \alpha, \mathfrak{I}(\alpha), \gamma_\alpha \Vdash_{\alpha} \mathfrak{E}(\alpha);\ \gamma_\alpha \Vdash_{\mathfrak{E}(\alpha)} \alpha;$$

$$\alpha \VDash_{\alpha} \mathfrak{I}(\beta);\ \alpha, \mathfrak{I}(\alpha) \VDash_{\mathfrak{I}(\alpha)} \mathfrak{I}(\beta), \beta;$$
$$\alpha, \mathfrak{I}(\alpha) \VDash_{\mathfrak{I}(\alpha)} \mathfrak{C}(\beta);$$
$$\alpha \VDash_{\mathfrak{C}(\alpha)} \gamma_\beta;\ \alpha, \mathfrak{I}(\alpha), \gamma_\alpha \VDash_{\alpha} \mathfrak{E}(\beta);\ \gamma_\alpha \VDash_{\mathfrak{E}(\alpha)} \beta;$$

$$\beta \Vdash_{\beta} \mathfrak{I}(\beta);\ \beta, \mathfrak{I}(\beta) \Vdash_{\mathfrak{I}(\beta)} \mathfrak{I}(\beta), \beta;$$
$$\beta, \mathfrak{I}(\beta) \Vdash_{\mathfrak{I}(\beta)} \mathfrak{C}(\beta);$$
$$\beta \Vdash_{\mathfrak{C}(\beta)} \gamma_\beta;\ \beta, \mathfrak{I}(\beta), \gamma_\beta \Vdash_{\beta} \mathfrak{E}(\beta);\ \gamma_\beta \Vdash_{\mathfrak{E}(\beta)} \beta$$

In this system, two essentially different parallel informational operators appear, namely, $\Vdash$ for cyclic and $\VDash$ for general parallel

Informing. It is certainly possible that also inter-informational parallel Informing can become cyclic, when a circular Informing between two informational entities is introduced. In this case we can say that simultaneously $\alpha \models \beta$ and $\beta \models \alpha$ are taking place. This can be our next case of IMOp. ■

[Transformation Rules]$^{\mathrm{DF15}}$:
In the case of the two-way or intercyclic Informing between informational entities α and β, the following rule of IMOp can be constructed:

$$\frac{(\alpha, \beta), (\alpha \models_{\mathfrak{I}(\alpha)} \beta, \alpha \Dashv_{\mathfrak{I}(\beta)} \beta)}{\mathfrak{C}_\alpha, \mathfrak{E}_\alpha \subset \mathfrak{I}_\alpha; \mathfrak{C}_\beta, \mathfrak{E}_\beta \subset \mathfrak{I}_\beta}$$

This form of IMOp has to consider the following complexities: the cyclic complexity of α's cyclic parallel Informing $\mathfrak{I}_\alpha$ (coming into existence in virtue of $\alpha \models \alpha$), considering the counter-Informing $\mathfrak{C}_\alpha$ and informational embedding $\mathfrak{E}_\alpha$; the one-way complexity of α's (apparently linear, but in fact inter-informationally circular) parallel Informing of β (coming into existence in virtue of $\alpha \models \beta$); the one-way complexity of β's (apparently linear, but in fact circular) parallel Informing of α (coming into existence in virtue of $\beta \models \alpha$); and the cyclic complexity of β's cyclic parallel Informing $\mathfrak{I}_\beta$ (coming into existence in virtue of $\beta \models \beta$), considering the counter-Informing $\mathfrak{C}_\beta$ and informational embedding $\mathfrak{E}_\beta$. This complexity can be expressed by the following parallel system:

$$\alpha \Vdash_\alpha \mathfrak{I}(\alpha);\ \alpha, \mathfrak{I}(\alpha) \Vdash_{\mathfrak{I}(\alpha)} \mathfrak{I}(\alpha), \alpha;$$
$$\alpha, \mathfrak{I}(\alpha) \Vdash_{\mathfrak{I}(\alpha)} \mathfrak{C}(\alpha);$$
$$\alpha \Vdash_{\mathfrak{C}(\alpha)} \Upsilon_\alpha;\ \alpha, \mathfrak{I}(\alpha), \Upsilon_\alpha \Vdash_\alpha \mathfrak{E}(\alpha);\ \Upsilon_\alpha \Vdash_{\mathfrak{E}(\alpha)} \alpha;$$

$$\alpha \Vdash_\alpha \mathfrak{I}(\beta);\ \alpha, \mathfrak{I}(\alpha) \Vdash_{\mathfrak{I}(\alpha)} \mathfrak{I}(\beta), \beta;$$
$$\alpha, \mathfrak{I}(\alpha) \Vdash_{\mathfrak{I}(\alpha)} \mathfrak{C}(\beta);$$
$$\alpha \Vdash_{\mathfrak{C}(\alpha)} \Upsilon_\beta;\ \alpha, \mathfrak{I}(\alpha), \Upsilon_\alpha \Vdash_\alpha \mathfrak{E}(\beta);\ \Upsilon_\alpha \Vdash_{\mathfrak{E}(\alpha)} \beta;$$

$$\mathfrak{I}(\alpha) \Dashv_\beta \beta;\ \mathfrak{I}(\alpha), \alpha \Dashv_{\mathfrak{I}(\beta)} \beta, \mathfrak{I}(\beta);$$
$$\mathfrak{C}(\alpha) \Dashv_{\mathfrak{I}(\beta)} \beta, \mathfrak{I}(\beta);$$
$$\Upsilon_\alpha \Dashv_{\mathfrak{C}(\beta)} \beta;\ \mathfrak{E}(\alpha) \Dashv_\beta \beta, \mathfrak{I}(\beta), \Upsilon_\beta \perp \alpha \Dashv_{\mathfrak{E}(\beta)} \Upsilon_\beta;$$

$$\beta \Vdash_\beta \mathfrak{I}(\beta);\ \beta, \mathfrak{I}(\beta) \Vdash_{\mathfrak{I}(\beta)} \mathfrak{I}(\beta), \beta;$$
$$\beta, \mathfrak{I}(\beta) \Vdash_{\mathfrak{I}(\beta)} \mathfrak{C}(\beta);$$
$$\beta \Vdash_{\mathfrak{C}(\beta)} \Upsilon_\beta;\ \beta, \mathfrak{I}(\beta), \Upsilon_\beta \Vdash_\beta \mathfrak{E}(\beta);\ \Upsilon_\beta \Vdash_{\mathfrak{E}(\beta)} \beta$$

In this system parallel cyclic operators $\Vdash$ and $\Dashv$ can be introduced since α and β are cyclically interwoven in an informational manner. Maybe the last example seems clumsy, but it shows a rich complexity in the case of inter-informational activity. This kind of complexity must certainly be considered in a case of informational reality. ■

It is evident that informational complexity for a general case $\alpha, \beta, \ldots, \gamma \models \xi, \eta, \ldots, \zeta$ and $\alpha, \beta, \ldots, \gamma \Dashv \xi, \eta, \ldots, \zeta$ can enormously grow. Identification of appearing inter-informational forms of Informing calls for particular rules of informational modus operandi.

II.4.3.9. The Case of Informational Modus Vivendi

How could the vital goal of staying alive or that of enjoying oneself shape any sort of information? Vital goals are satisfied only when active, specific goals are.

Radu J. Bogdan [13] 92

Informational modus vivendi concerns information of life in environmental, individual, populational, and social circumstances. Several levels and sorts of life information can certainly be distinguished. The basic living information present everywhere where the living arises may be marked as autopoietic information α. This information may be compared to basic informational fuel of which any higher living informational forms and processes are composed and aggregated. This informational fuel includes the most elementary and primitive informational lumps, living informationally related and unrelated in their biological environment and out of which, during a life cycle, higher and more complex informational forms and processes would come into existence.

We can imagine, for instance, how in a living being its total information called metaphysics μ is permanently arising out of informational lumps within its autopoietic system, where α is coming into existence, changing, and vanishing. This metaphysics μ represents a life related informational form and process of autopoietic information α. In these circumstances, α together with stimulus or sensory information σ enables the coming of metaphysics μ into existence. Through life processes, α and σ structure and organize μ, thus, as we say, inform μ. In general,

$$\alpha, \sigma \models \mu$$

At first, this process could be seen as an initial process of metaphysical development of a living unit. As soon as μ begins to develop, it begins to impact a being's autopoietic system, i.e., its autopoietic information α, and it begins to filter and modulate metaphysically the sensory information σ. So, to the initial process, the process

$$\mu \models \alpha, \sigma$$

can be attributed.

Further, an essential part of metaphysics μ is the so-called behavioral or motor information β, by which a being performs its acting (intelligent deciding) within its autopoietic system and in its environment. In processes of life all informational occurrences of a living being interact, so, a general living system can be demonstrated informationally in the form

$$\alpha, \sigma, \mu, \beta \models \alpha, \sigma, \mu, \beta$$

This informational system can be decomposed into basic interacting parallel processes, for instance,

$$\begin{array}{l}\alpha \Vdash \alpha,\ \alpha \Vdash \sigma,\ \alpha \Vdash \mu,\ \alpha \Vdash \beta,\\ \sigma \Vdash \alpha,\ \sigma \Vdash \sigma,\ \sigma \Vdash \mu,\ \sigma \Vdash \beta,\\ \mu \Vdash \alpha,\ \mu \Vdash \sigma,\ \mu \Vdash \mu,\ \mu \Vdash \beta,\\ \beta \Vdash \alpha,\ \beta \Vdash \sigma,\ \beta \Vdash \mu,\ \beta \Vdash \beta\end{array}$$

This system says that not only informational entities α, σ, μ, and β interact, but that also their Informings $\Im_\alpha$, $\Im_\sigma$, $\Im_\mu$, and $\Im_\beta$ interact within the informational parallelism of the basic processes $\alpha \Vdash \alpha$, $\alpha \Vdash \sigma$, ... , $\beta \Vdash \mu$, $\beta \Vdash \beta$.

The basic system being described can be broadened into the form

$$\begin{array}{l}\alpha \Vdash \alpha,\ \Im_\alpha;\ \alpha \Vdash \sigma,\ \Im_\sigma;\ \alpha \Vdash \mu,\ \Im_\mu;\ \alpha \Vdash \beta,\ \Im_\beta;\\ \sigma \Vdash \alpha,\ \Im_\alpha;\ \sigma \Vdash \sigma,\ \Im_\sigma;\ \sigma \Vdash \mu,\ \Im_\mu;\ \sigma \Vdash \beta,\ \Im_\beta;\\ \mu \Vdash \alpha,\ \Im_\alpha;\ \mu \Vdash \sigma,\ \Im_\sigma;\ \mu \Vdash \mu,\ \Im_\mu;\ \mu \Vdash \beta,\ \Im_\beta;\\ \beta \Vdash \alpha,\ \Im_\alpha;\ \beta \Vdash \sigma,\ \Im_\sigma;\ \beta \Vdash \mu,\ \Im_\mu;\ \beta \Vdash \beta,\ \Im_\beta\end{array}$$

where $\Im_\xi$, $\xi \in \{\alpha, \sigma, \mu, \beta\}$ is Informing of information in question. This parallel informational system can further be decomposed (particularized) into more and more details.

It has to be stressed again that autopoietic information α is a kind of basic architectural, molecularly structured and by molecular processing organized information of a living being - also of a living cell. Information α is a matter of molecular processes within complex molecules of life, microtubules, cell lumps and generally subunits of the cell as entirety, etc.

In constructing various kinds of informational modi vivendi, living informational components α, σ, μ, and β can be considered as basic elements or a background of specialized and dedicated living informational entities. On higher levels of living structure and organization, e.g., on the level of higher cortical processes, modus vivendi embraces all of the imaginable modi informationis where each special modus can be a part or a function of modus vivendi. The information, which living beings are capable to produce, is in principle only autopoietic, thus its arising is under the impact of such or another modus vivendi in a particular time slice (step of development) and within a particular environment. What a living being thinks, hypothesizes, does, performs, informationally adopts, etc., can arise only within the realm of its autopoietically informational.

In this section we shall not discuss other specific modi informationis (ponens, tollens, rectus, etc.), but will concentrate to revealing some specific and elementary life processes, which originate, preserve, and destruct life, i.e., the real and essential forms of modus vivendi.

As a modus informationis we have determined an iwff which uses the so-called fractional (detachment) or extractive line operation. Now, cases of modus vivendi have to be constructed in this standard way.

[Transformation Rules]$^{\mathrm{DF16}}$:
Let α, σ, μ, and β be autopoietic, sensory (stimulus), metaphysical, and behavioral (motor) information, respectively. It is to understand that after its conception metaphysics μ is certainly being informationally embedded in autopoietic information α. In the very beginning of a being's conception, when only its autopoietic information arises, its beginning metaphysics is coming into existence. This fact can be expressed by the modus

(1) $$\frac{\alpha,\ \alpha \models \mu}{\mu}$$

The meaning of this modus is the following: if there exists autopoietic information α and if α informs metaphysics μ, then there exists μ. This modus has to be conjoined with the axiom

(1a) $$((\alpha \models) \lor (\models \alpha)) \Rrightarrow (\alpha \llcorner \mu)$$

which governs the conception of metaphysics μ and says the following: if α is autopoietic information (if it informs and is informed), then α causes the appearance of μ or, cuases μ to come into existence by Informing of α. This property of autopoietic information, to conceive its metaphysics, exists as its own intention and is a way of its Informing and informational development.

At the conception of a being's metaphysics μ also Informing of autopoietic information α and Informing of arising metaphysics μ can be considered in the operationally explicit way, by the following modus:

(2) $$\frac{\alpha \models,\ (\alpha \models \mu) \models}{\mu \models}$$

The meaning of this formula is as follows: if α informs (arises) and if the process $\alpha \models \mu$ informs (arises), then μ informs (arises) as well. This conclusion is important because $\mu \models$ does not follow explicitly from the antecedent of the last modus. In fact, to this modus the following axiom can be conjoined:

(2a) $$\begin{array}{c}(((\alpha \models) \models) \lor (\models (\alpha \models)) \lor \\ ((\models \alpha) \models) \lor (\models (\models \alpha))) \\ \Rrightarrow (((\alpha \models) \lor (\models \alpha)) \llcorner ((\mu \models) \lor (\models \mu)))\end{array}$$

Although the consequence of the last implication is not essentially different from the consequence of (1a), it can be read differently, namely, that complex Informing of α causes the appearance of complex Informing of μ. However, complex Informing of information is nothing else but information itself.

As in an environment autopoietic information α and metaphysics μ conceived by it are always arising, the impact of sensory information on metaphysics is taking place:

(3) $$\frac{(\alpha,\ \sigma),\ (\alpha,\ \sigma \models \mu)}{\mu}$$

If autopoietic information α and sensory information σ exist and if they inform metaphysics μ, then metaphysics μ exists or performs as information impacted by α and σ. This modus can be conjoined by the axiom

(3a) $$((\alpha \models) \vee (\models \alpha) \vee (\sigma \models) \vee (\models \sigma)) \Rightarrow (\alpha, \sigma \mathrel{L} \mu)$$

This axiom seems to be informationally more accurate than axiom (1a), however, we have to consider that axiom (1a) was the very beginning of the conception of metaphysics. In fact, axiom (3a) is an iwff describing the continuation of conceptional formation of metaphysics and has the following meaning: if α is autopoietic information which has already conceived its metaphysics μ, then in further process of conception, α together with sensory information σ informationally impacts metaphysics μ. In this respect (3a) is a sequel of (1a). On the other hand, (3a) can be seen as a particularization of (1) where α was substituted by α, σ. Thus, modus (3) has the semantic value because it shows the impact of α as well as σ on extraction of μ.

Similarly to modus (2) it is possible to broaden modus (3) to the Informings of impacting and impacted informational components. Thus, the following modus is obtained:

(4) $$\frac{(\alpha \models, \sigma \models), (\alpha, \sigma \models \mu) \models}{\mu \models}$$

This formula is a particularization of modus (2), where $\alpha \models$ was substituted by $\alpha \models, \sigma \models$ and α by α, σ. This modus has the semantic value in showing the impact of antecedent Informings on consequent Informing of μ. As in previous cases, this modus can be conjoined with the axiom

(4a) $$\begin{aligned} &(((\alpha \models) \models) \vee (\models (\alpha \models)) \vee \\ &((\models \alpha) \models) \vee (\models (\models \alpha)) \vee \\ &((\sigma \models) \models) \vee (\models (\sigma \models)) \vee \\ &((\models \sigma) \models) \vee (\models (\models \sigma))) \\ \Rightarrow &(((\alpha \models) \vee (\models \alpha) \vee (\sigma \models) \vee (\models \sigma)) \\ &\mathrel{L} ((\mu \models) \vee (\models \mu))) \end{aligned}$$

As the conception of μ is progressing it becomes more and more clear that particular modi vivendi must satisfy the basic expression of the living, namely,

$$\alpha, \sigma, \mu, \beta \models \alpha, \sigma, \mu, \beta$$ ■

[Transformation Rules]^{DF17}:
Let us list several other modi vivendi explicating Informings of vital informational components. For autopoietic information α there is the basic modus

$$\frac{\alpha}{\alpha \models \mathfrak{I}_\alpha}$$

It means that autopoietic information α informs (generates, observes, also autopoietically limits) its own Informing $\mathfrak{I}_\alpha$. This modus is informationally regular, for Informing $\mathfrak{I}_\alpha$ is an implicit component of living information. This is the known principle of Informing of information [4, 11]. The last modus is informationally regular, for Informing $\mathfrak{I}_\alpha$ is only an implicit operational component of α, hidden in α (in the antecedent of the last formula). Thus, this modus means explication or extraction of the component hidden in α. This concealment could be considered, for instance, by the formula

$$\alpha = \alpha(\mathfrak{I}_\alpha)$$

Similarly as in previous transformation rule, now it is possible to reveal modi concerning Informings of autopoietic and metaphysical information. Thus,

$$\frac{\alpha \models \mathfrak{I}_\alpha, \alpha \models \mu}{\mathfrak{I}_\alpha \models \mu}$$

In course of conception of metaphysics, by metaphysics, its Informing is coming into existence, and the following modus can be observed:

$$\frac{(\alpha, \sigma), (\alpha, \sigma \models \mu)}{\mu \mathrel{L} \mathfrak{I}_\mu}$$

Finally, within autopoietic information α, and metaphysics μ embedded in it, as well as a consequence of appearing sensory information σ, information of a being's behavior β is coming into existence as a reaction to all of these informational circumstances. We suppose that the conception of β begins by μ and we can adopt the following axiom:

$$(\alpha, \sigma, \mu \models \mu) \Rightarrow (\mu \mathrel{L} \beta)$$

Afterwards, we can introduce the following modus:

$$\frac{(\alpha, \sigma, \mu), (\alpha, \sigma, \mu \models \beta)}{\beta}$$

This modus assures the existence of behavioral information β. It becomes evident that after this discussion different informational axioms arise concerning information of the living. For instance,

$$\begin{gathered} (\alpha \models) \Rightarrow (\mathrel{L} \mu) \\ (\mu \models) \Rightarrow (\mathrel{L} \beta) \\ ((\alpha \models) \vee (\sigma \models)) \Rightarrow ((\mathrel{L} \mu) \vee (\mathrel{L} \beta)) \end{gathered}$$

etc. ■

Examples of different modi vivendi have shown that modus vivendi above all presents a case being semantically bound to living being. The exposed cases of modus vivendi retain their meaning also in a more general informational sense, for arising of information is a living as well as cosmic and artificial phenomenon.

In this section we have discussed only very general forms of modus vivendi. We did not examine concrete modi, concerning higher intellectual functions and higher forms of life. Modus vivendi concerns any realm of living activity and can certainly be concretized for any field, form, or process referring to a living being.

II.4.3.10. The Case of Informational Modus Possibilitatis

Possibility is a modal determination which opposes reality (essential, existential) and necessity. Modality by itself is a mood of revealing of Being, occurrence, or thinking; it is a mood of conditionality. In logic, modality of propositions means the degree of trustability of propositions in regard to possibility (e.g., a problematic proposition is $\alpha \models_\pi \beta$ or $\alpha =_\pi \beta$ with the meaning α can be β), existence (e.g., an asserting proposition is $\alpha = \beta$ with the meaning α is β), and to necessity (e.g., an apodictic proposition is $\alpha \models_{must\_be} \beta$ with the meaning α must_be β). According to Kant, categories of modality are possibility/impossibility, existence/non-existence, and necessity/chance. In psychology, modality is a common domain of quantitatively related sensations, conditioned by functions of certain organs, for instance, sensations of sight, hearing, etc.

In formal-informational sense, possibility means that something, which is informational, can always be informed or can come into existence as a new informational subject and object, irrespective to its particular nature of counter-Informing and informational embedding. A cognitive-theoretical or material-objective possibility is only a particular information which can arise irrespectively to a concrete metaphysical experience which might be or might not be a reference versus arising possibility. Metaphysics as a total information of a being certainly conditions the possibility of informational arising according to the autopoietic informational nature of a being. As information possibility is a potential dynamics of information to arise into a new, unforeseeable and foreseeable informational phenomenon.

[Transformation rules]^{DF18}:
The basic question of informational possibility is the following: if α is information, how can it arise. In fact, informational possibility and informational arising concern a common and essential question of information. Information can arise in various ways. The possibility of this arising is impacted by Informing of α by itself and by Informing of other information β which can possibly impact α. If $\alpha, \beta \models \alpha$, then α can arise into any particular information, information γ, for instance; thus

$$\alpha, \beta \models \alpha \models_\pi \gamma$$

The possibility of Informing of α is hidden within the informational process $\alpha \models \gamma_\pi$, where γ is the possible information. Informational modus possibilitatis has to detach possible information. If γ unites possible information, then it marks an informational set of possible informational entities, for instance, $\gamma_1, \gamma_2, \ldots, \gamma_n$, which are samples of possible entities. These entities can be informationally conjoined or interwoven. Thus, possibility γ hides possibilities $\gamma_1, \gamma_2, \ldots, \gamma_n$.

An example of informational modus possibilitatis can be the following:

$$\frac{\alpha, \beta; \alpha, \beta \models \alpha \models_\pi \gamma}{\gamma \models_\pi \gamma_1, \gamma_2, \ldots, \gamma_n}$$

Informational modus possibilitatis answers the question what is informationally possible. According to philosophy of informational logic, any (also unforeseeable) informational arising is possible, it means, any arising, which leads to the appearance of counter-informational, contrary, absurd information, etc. Further, possible information is probable as well as improbable information, whose informational framework can be sensed or felt or can stand outside of a metaphysical imagination, which may happen and which is unforeseeable, informationally not yet revealed. ■

II.4.3.11. The Case of Informational Modus Necessitatis

The 'must' is compelled by necessity. Necessity as information is a pressure of informational circumstances, is informational impossibility (also incapability) of a contrary information. It is an urgent informational need and desire, in such a way, that it cannot be otherwise. Necessity can be comprehended as an inevitable informational consequence.

Informationally, necessity is a mood of revealing and functioning of reality, Being, essentialness, etc. It is a mood of principles and informationally constructed legality of various informational systems imagined as they function orderly in reality. Necessity can be, for instance, a form of fatalism, a deterministic conception, by which existence and arising of information are understood as a necessary (predetermined) phenomenology which informs as such in the past, present and future. These concepts of necessity can reject chance and also possibility as categories of objectiveness. Dialectically, necessity is comprehended to relate chance and possibility according to reality of life where a living being can estimate various possibilities and can make determined decisions. Further, necessity can also be conceived as a form of informational repression.

[Transformation Rules]^{DF19}:
Let us introduce the following definition:

$$(\text{'}\nu \text{ is\_information\_of\_necessity'}) =_{Df} ((\nu \models_N) \vee (\models_N \nu)$$

where N is the Informing of necessity as information within ν. Now, let information α be informed by ν, thus, $\nu \models_N \alpha$. We say that in this case α is N-deterministic (necessity-deterministic). Certainly, it is possible to construct various rules of modus necessitatis. Let us take the following example:

$$\frac{\nu, \nu \models_N \alpha}{\alpha \therefore_N \nu}$$

If ν is information of necessity (necessary information) and if this information informs, by necessity N, another information α, then information α is informationally compatible in regard to information ν. In this case, it is said that ν and α support informationally the so-called informational kernel of necessity. ■

Within a formal theory, it is necessary to follow its axioms and rules of transformation, otherwise the theory can expose inadmissible contradictions. Within an ideology, it is necessary to develop only ideologically permissible information which is informed by virtue of ideological kernel.

II.4.3.12. Cases of Informational Modus Informationis

Cases of different elementary modi informationis have been presented, from the informational modus ponens to the informational modus necessitatis. From these elementary modi it is possible to construct mixed modi informationis (for instance, modus ponendo ponens, modus ponendo tollens, modus tollendo tollens, modus tollendo ponens, modus ponendo rectus, etc.) and compose them to more and more sophisticated informational rules for iwff transformation.

On the other side, we have to keep in mind that information by itself is a transformational entity which, besides of the previously explicated cases of modus informationis, develops itself and is developing other and developed by other informational entities. Modus informationis is just another constructive look at the same problem, namely, at informational arising. However, in a concrete case, modus informationis is a transforming system in Informing of iwffs in the framework of the concrete case. Certainly, there exist an indefinite number of cases of modus informationis where their antecedent and consequent parts are thrown into spontaneous and circular Informing with an instantaneous intentionality of living or artificial information. Besides of transformational arising of informational parts within a certain information, the general and particular informational entities can always be additionally particularized and universalized in a new way. This principle of particularization and universalization may come close to the intention of designing living and particularly artificial information.

II.4.3.13. Conclusion Concerning Modus Informationis

Through the transformation principle of modus informationis we have opened the real abyss of developmentally extracting possibilities, where information is extracted from information by the way of informational arising. In this regard modus informationis is another essential principle of informational arising for it seizes into the elementary philosophy of informational phenomenology. In fact, it helps to reveal concepts of a discrete Informing of informational entities which informationally develop in themselves. In this manner, modus informationis contributes also to explanation of the concept of informational arising. It is possible to say that modus informationis is a part of explanation which concerns the most general informational metaoperator $\models$.

II.4.4. Rules of Informing

Rules of Informing as introduced in the form of informational axioms and transformation rules of IL concern a specific, theoretically and symbolically logical nature of Informing of information. These rules remain open to the processes of their further informational development (arising) and, certainly, of their particularization and universalization. In fact, any process of Informing is performed as an informational rule upon informational units within an informational domain. Thus, Informing by itself can be understood as an instantaneous rule applied spontaneously and circularly upon instantaneous informational entities.

II.4.4.1. Openness of Introducing New Transformation Rules

The transformation rules determined in the previous paragraphs show the possibilities of their indefinite continuation of development. Beside of the existing cases of transformational detachability new and much more complex informational transformations are possible. It is, of course, also possible to add new transformational types to the detaching ones. Modus informationis as a general principle of Informing can embrace variously imaginable rules for transformational arising of information. The consequence of these possibilities is that a transformational system remains open for new transformational determinations. On this basis it is possible to conclude that informational transformation, as presented in the previous cases, irrespective of the informational system (theory, mind, behavior, etc.) involved, remains open in the informational sense. This informational phenomenon enables to open new principled questions concerning the nature of possible informational transformation.

The basis of informational transformation of IL remains developmentally open. Principles of informational particularization and universalization concern informational transformation rules in the same sense as they concern information and its Informing in general. They are a constructively senseful component of keeping the transformation basis open. Thus, the exposed informational transformation rules perform as regular information. They are informational.

II.4.4.2. Transformation Rules and Metaphysical Beliefs

Certainly, the listed transformational cases arise from a particular metaphysical disposition from which they are thrown as cases of modus informationis into a broader scientific, professional, and philosophical discourse. Where are the limits of informational arising of discussed transformation rules? The answer is that only in the metaphysics which dwells and develops on its autopoietic foundation. Beliefs, intentions, and desires cause their creation. Some principles of their creation seem to be evident, at least some very primitive ones. In this sense, informational transformation rules can preserve their developmental and arising power, of course, being impacted by their cultural environment.

We have recognized how informational modi can be developed from some ancient and also modern, for instance, mathematical principles of inference, proof theory, and common sense. In each case, these modi have been informationally (conceptually) broadened, and did not stay only on their traditionally philosophical and mathematical foundations. The traditional meaning of these modi was preserved in their most primitive forms, but they could be developed in a more general way and preserving not only traditional logical relations. It is relevant to stress that modi informationis became a regular arising of information.

II.5. A SURVEYING CONCLUSION CONCERNING THE FORMAL INFORMATIONAL LOGIC

The informational logic presented in this essay has not always been placed inside of the strict traditional rationalism and has not built any protective ditch against the possibilities of its further development, arising, and theoretical improvement. In this respect it was in no way closed as a sulky routine of logical positivism being characteristic for some Western posts of the angry common sense. This means that IL stands on a broadened theoretical ground of a sound reasoning. The most relevant theoretical origin of IL was the construction of its axiomatizational and transformational possibilities of informational entities which root in the phenomenology of the entire information of a being, in the so-called being's metaphysics. By such a way of formalization, to IL was given the semantic nature of informational arising on the level of informational operands as well as on the level of informational operators. The most general operational variable of informational arising was the operator $\models$ which became also an inward property of operand information α as an arising entity. To stress clearly, this was the most relevant innovation to the formalistic conception of information and Informing of information, where $\models$ as operational variable obtained the possibility to be particularized and universalized.

References of IL I, II, III, and IV

[1] H. L. Dreyfus and S. E. Dreyfus: Mind over Machine. The Free Press, Macmillan, New York 1986.

[2] A. P. Železnikar: On the Way to Information. Informatica 11 (1987), No. 1, 4-11.

[3] A. P. Železnikar: Information Determinations I. Informatica 11 (1987), No. 2, 3-17. [Pyblished also in Cybernetica 31 (1988), 181-213.]

[4] A. P. Železnikar: Principles of Information. Informatica 11 (1987), No. 3, 9-17. [Published also in Cybernetica 31 (1988), 99-122.]

[5] A. P. Železnikar: Information Determinations II. Informatica 11 (1987), No. 4, 8-25.

[6] A. P. Železnikar: Problems of Rational Understanding of Information. Informatica 12 (1988), No. 2, 31-46.

[7] L. A. Steen (Editor): Mathematics Today. Twelve Informal Essays. Springer-Verlag. New York (Third Printing, 1984).

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

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

[10] A.P. Železnikar: Principles of Information. Cybernetica 31 (1988) 99-122.

[11] S. Schiffer: Symposium on Remnants of Meaning: 1. Overview of the Book. Mind and Language 3 (1988), No. 1, 1-8 (Basil Blackwell).

[12] T. Winograd: On Understanding Computers and Cognition: A New Foundation for Design (A response to the reviews). Artificial Intelligence 31 (1987) 250-261.

[13] R.J. Bogdan: Information and Semantic Cognition: An Ontological Account. Mind & Language 3 (1988), No. 2, 81-122.

[14] D.C. Dennett: Intentional Systems. In J. Haugeland, Ed.: Mind Design (Philosophy, Psychology, Artificial Intelligence). The MIT Press, Cambridge, Ma. (1985).

[15] A.P. Železnikar: Informational Logic I. Informatica 12 (1988), No. 3, 26-38.

[16] A.P. Železnikar: Informational Logic II. Informatica 12 (1988), No. 4, 3-20.

[17] A.P. Železnikar: Informational Logic III. Informatica 13 (1989), No. 1, 25-42.

INTERCONNECTION NETWORK ANALYSIS AND LOGIC DESIGN

INFORMATICA 2/89

Keywords: parallel processing, interconnection networks, asynchronous vs. synchronous timing, logic design

Andrej Žagar, Peter Brajak
Iskra Delta, Ljubljana

This paper shows the development efforts in designing a interconnection network for 64 processor tightly coupled PARSYS parallel machine. In the first two chapters a network analysis is described based on the discrete statistical simulation model. From the results of the discrete statistical simulation model a high performance asynchronous Routing Node logic design is defined and implemented for the n-cube based interconnection network.

1. INTRODUCTION

We are entering the era of fundamental changes in the field of computer architecture. In particular, a large part of these changes have been represented by a transition from serial to parallel processing. This transition is stimulated by at least four reasons:

- the cost of digital hardware has dropped to the point that processors need not be considered the scared resources,
- improvement of performance of monoprocessors is technologically limited,
- the scope of problems whose algorithmic complexity exceeds the capability of todays most powerful computers is very large,
- the formulation of the problems naturally suggest parallel realization.

Market Research Group, a consulting company that follows the trends on computer market, foretells that by the year 1990, 48% of all large scale computers will have one of the forms of parallel processing. Most of the developed countries support parallel system projects (Japan, German Federative Republic, USA, etc.). These projects differ from technological to philosophical concepts. However, one is common to all of them: the proposed computers are order of magnitude faster than conventional computers, they are successful, and there is a fully new open market for the application software.

1.1. RATIONALE AND CONCEPTS

PARSYS[12,13] is a tightly-coupled MIMD (multiple instruction, multiple data) research and development project on parallel processing at Iskra Delta Computers. Specifically, the project is involved with:

- development of the first prototype hardware and system software,
- development of specific program development environments,
- development of application software.

The rationale for the decision to build a MIMD tightly-coupled parallel system rather than a super fast SIMD (single instruction, multiple data) vector machine is multifold:

- the market for MIMD multiprocessors is just beginning to evolve, whereas for vector machines it is highly competitive,
- the higher flexibility of MIMD machines allows for a broader application spectrum,
- the development cost of the MIMD based multiprocessor system is lower than of the vector machine, since it is possible to use of-the-shelf standard components,
- the whole spectrum of new software products could benefit from general purpose MIMD machines (vector processors are too much specialized).

The PARSYS project is product-oriented. This means that in contrast to pure research and prototype development, an additional market-oriented requirements must be satisfied such as:

- the desired absolute performance must be obtained at the competitive cost-effectiveness,
- the machine must be manufacturable, testable and maintainable,
- there must exist a time schedule during which the research and development must lead to a production model.

In order to meet the time schedule, the architectural design of the PARSYS machine has been based not only on the innovative concepts but also on the solutions that have incorporated already proven technologies, methodologies, standards and products.

The recent research and development work on the PARSYS system has been focused on:

1. top level architectural specification and implementation a 64 processor machine based on multistage interconnection topology,
2. implementation of minimal kernel for fast process communication,
3. development of software tools for automatic parallelization and debugging.

In this paper we show only the hardware realization of the PARSYS n-cube interconnection network. The work was done at Iskra Delta Computers as partial fulfillment for Bs.EE. degree[11] from June 1987 to September 1988.

2. INTERCONNECTION NETWORK ANALYSIS

Interconnection networks for concurrent computers have been studied intensively, and many different topologies have been proposed: from non-blocking, but impractical crossbar switches, Benes and Batcher networks to more practical blocking networks such as omega, flip, delta, indirect binary n-cube.

PARSYS original network topology used flip multistage interconnection network[1], directly transformed into the binary n-cube. The benefit of the transformation was the simplicity of the n-cube implementation and the versatility of the multistage interconnection network.

PARSYS machine can be functionally represented as in figure 2.1. Programs and data can be either stored in processor's local (LM) or global memory (GM). CPU-s can access global memory only through the interconnection network. The problem is that the time delay required to access global memory is much larger than the time to access local memory. In order to decrease traffic in the network and to increase overall program execution it is required that both program code segments and parts of data segments (local stack) are placed in the local memory.

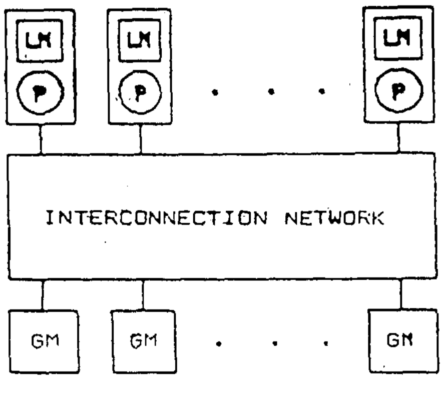


Figure 2.1. The parallel computer with distributed global and local memory

The interconnection network analysis must take advantage of the fact that the traffic flow is greatly reduced by introducing the local memories in the system and that a simple network design can sufficiently absorb traffic flow and bring global memory request in the fastest possible way.

2.1. THE PURPOSE OF INTERCONNECTION NETWORK

The interconnection network statistical analysis and the interconnection network simulation introduced in this paper are valid for any multiprocessor interconnection network topology which is based on global/local architecture. The goal of these two chapter is to show transport characteristics in the network and to make a ground for the optimal multiprocessor network logic design based on the results of the analysis .

CPU access global memory by creating a read or write packet. Packets travel through the network using self-routing mechanism described in [1].

The read packet contains an address and a routing tag when it travels from the processor to the destination global memory module (route: P -> GM) and it consists of data and the same routing tag when it returns back to the same processor (route: GM -> P).

The write packet consists of an memory address, a routing tag and data. It travels only in one direction: from the processor to the destination global memory module (route: P -> GM).

2.2. INTERCONNECTION NETWORK STATISTICAL ANALYSIS

It is very hard to determine statistically the percentage of global read and global write memory accesses. However, an average and a maximum number of packets in the network are very valuable information for the optimization process of the overall network design.

When statistical features and relative time delays have been included in the model, the basic network's features can be restored.

Fortunately, we can use some statistics gathered from the uniprocessor systems and tune them for the analysis of the parallel interconnection design. The results of statistical analysis of computer programs for IBM mainframe machines[6] have shown the following characteristics:

1. a typical program may be represented by the finite automata,
2. instructions may be divided in three characteristic groups, each group representing a node in the automata,
   - in the first group are instructions like INA, ASLA and CLRA. These instructions are register only instructions do not require memory access, PC := PC+1,

- memory jump instructions (JMP, JSR,..), PC := <new address>
- read and write instructions (LDA, STA,..), Reg := Memory[Address]; PC := PC+1;

3. there exists a state transition among the groups: (figure 2.2.1)

   - probability, that an instruction from the first group is followed by an instruction in the same group is 11/20,
   - probability that an instruction from the first group is followed by an instruction from the second group is 1/4,
   - probability that an instruction from the first group is followed by an instruction from the third group is 1/5.
   - the similar relations are for the second and for the third group, as well.

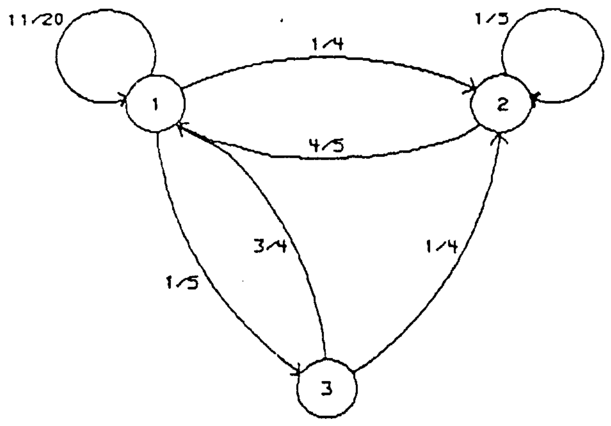


Figure 2.2.1. Probability scheme of the instruction cycle

4. when the memory space is divided to the local and the global address space, then only every tenth instruction will require global memory request for its execution.

Since we are interested only at the memory requests that require global memory, we can compress state 1 and state 3 into only one state, called local state, and state 2 called global state.

A probability that machine will turn from the local state to the global state is 1/10.

The following assumptions are necessary to construct a statistical model of the multiprocessor network based on previously defined probabilities:

- programs are all loaded into local memories,
- there is an equal number of read and write instructions,
- a probability scheme of the CPU machine cycle is equal to the probability scheme of the instruction cycle,
- the CPU's machine cycle is equal to the processor's machine cycle,
- the processor which has just sent a read packet to the network has to wait until this packet returns,
- the processor has finished a write cycle when the write packet had been sent in the network,
- PreRouting and PostRouting delays are treated as a part of the network delay,
- the processors are treated as a two state machines,
- the processors are statistically independent,
- network is deadlock free,
- there is no combining in the network necessary[7].

From the statistical analysis and the assumptions previously stated, it can be concluded that the processor in 20 machine cycles generates one read and one write packet.

The following symbols should be defined:

- N is a number of processors,
- Tcpu is the CPU'S machine cycle,
- Tn is an average packet delay in the network,
- Tm is a global memory module's access time,
- Tw is time which is needed to write data in the destination global memory module,
- Tr is time which is required to read data from the global memory module and
- Np is a number of packets in the network.

Therefore:

$$Tr = Tn + Tm + Tn. \qquad 2.2.1.$$

$$Tw = Tn + Tm. \qquad 2.2.2.$$

2.3. INTERCONNECTION NETWORK SIMULATION

A discrete simulation model (figure 2.3.1.) is constructed for the three functional parts:

- (CPU(Td)) delay units represent the processors,
- (WPGM(Ta), RPGM(Tb) and RPCPU(Tc)) delay units represent the packets in the multiprocessor network and
- (GM(Te)) delay units represent the global memory.

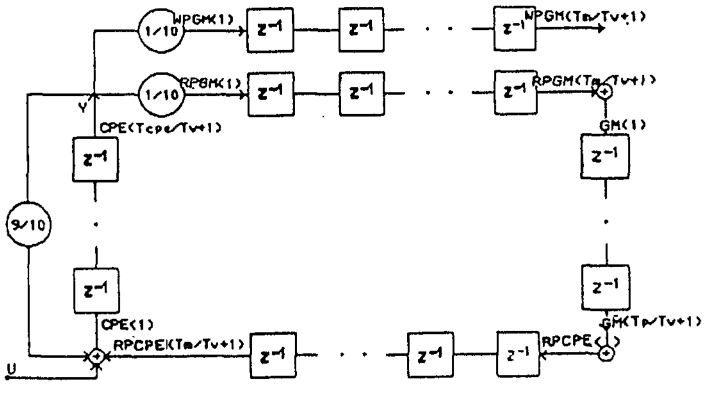


Figure 2.3.1.

The number of packets which are sent to the network at the moment T, is depended on the number of processors which finished their last machine cycle in the moment (T-Tcpu). The read packets (RPGM(Tb) delay units) and the write packets (WPGM(Ta) delay units) travel through the network to the global memory modules (route: P -> GM). When the write packets come to the global memory modules, write data in the memory and their route is finished there. On the other side, the read packets read data from the global memory modules (GM(Te) delay units) and after that, they return to the processors with required data (RPCPU(Tc) delay units, route: GM -> P).
Exact definitions are:

N * WPGM(Ta)	- a number of write packets which are Ta time units in the network,
N * RPGM(Tb)	- a number of read packets which are Tb time units in the network (route: P -> GM),
N * RPCPU(Tc)	- a number of read packets which are Tc time units in the network (route: GM -> P),
N * CPU(Td)	- a number of processors which are going to start new machine cycle for (Tcpu/Ts + 1 - Td) time units,
N * GM(Te)	- a number of global memory modules which are going to return read packets in the network for (Tm/Ts + 1 - Te) time units and
Ts	- is a sampling time.

The number of all packets in the network (Np) is a sum of the read and the write packets which are in the network at the particular moment:

$$Np = N * \sum_{i=1}^{Tn/Ts+1} (WPGM(i) + RPGM(i) + RPCPU(i)). \qquad 2.3.1.$$

The initial condition is a delta impulse on one of the processors' delay units CPU(Ti). The simulation will start after (Tcpu/Ts + 1 - Ti) time units.

The probability scheme is valid for the first machine cycle too.

All processors could send packets to the network simultaneously. However, this possibility is one of the initial conditions too. If this condition could happen, the network would congest mostly. The maximum number of packets in the network is:

- N, if 0 < Tn < Tcpu and

- k * N, if (k - 1) * Tcpu < Tn < k * Tcpu.

2.3.2.

System function is:

$$\frac{Y(Z)}{U(Z)} = \frac{Z^{-Tcpu/Ts}}{1 - 19/20 * Z^{-Tcpu/Ts} - 1/20 * Z^{-(Tcpu+Tn+Tm+Tn)/Ts}} \qquad 2.3.3.$$

The sampling time is bounded by the upper and the lower limit. The upper limit is determined by:

$$Tcpu = a * Ts,$$
$$Tn = b * Ts \text{ in}$$
$$Tm = c * Ts, \qquad 2.3.4.$$

where a, b and c are integer. If the condition is not accomplish, the extended Z-transformation must be used and the model can not be simulated on digital computers.

The lower limit is defined by the number of the delay units which represent the processors. The maximum number of CPU(Ti) delay units is N. This is the highest asynchronization which can be hardly achieved with N processors.

The sampling time is determined by the requested precision and the observing range.

y(k) is defined by the inversive Z-transformation:

$$y(k) = Z^{-1} (Y(Z)). \qquad 2.3.5.$$

The number of all packets in the network (Np) is:

$$Np(k) = N * (1/20 * \sum_{i=k-b}^{k} y(i) + 1/20 * \sum_{i=k-b}^{k} y(i) + 1/20 * \sum_{i=k-b-c-b}^{k-b-c} y(i)) \qquad 2.3.6.$$

for T = k * Ts.

The analytic analysis is very complicated so the discrete simulation is used.

The number of all packets in the network (Np) after the transitional phenomenon is found out by the simulation algorithm given in appendix I.

The simulation results are shown in the figure 2.3.2..

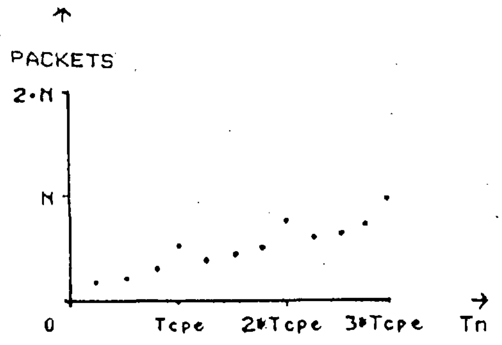


Figure 2.3.2. The number of all packets in the network (Np) after the transitional phenomenon (Tcpu = 125 ns, Tn = 25 - 300 ns, Tm = 100 ns, Ts = 5 ns)

The number of packets increases with Tn. However, the non-regularity shows up at the condition:

$$Tn = k * Tcpu \; ; \; k = 1,2,.. \qquad 2.3.7.$$

This non-regularity must be bypassed. This can be assured by enforcing asynchronous processor operations.

The maximum number of all packets in the network (Npm) is found out by the simulation algorithm given in appendix II.

The simulation results are shown in the figure 2.3.3..

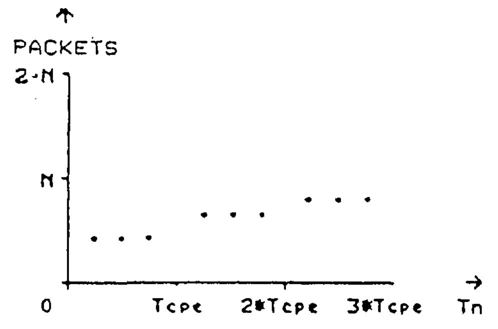


Figure 2.3.3. The maximum number of all packets in the network (Tcpu = 125 ns, Tn = 25 - 300 ns, Tm = 100 ns, Ts = 5 ns)

A comparison between the equation 2.3.2. and the figure 2.3.3. shows that the maximum number of all packets in the network depends on the initial condition of the simulation. Furthermore, the maximum number of packets is in the network when processors work synchroniously.

Therefore, the simulations show that hardware asynchronization of the whole system (processors among each other and network itself) must be assured.

An efficient network can be built if only Tm is slightly lesser than k * Tcpu.

The results of the interconnection network statistical analysis based on the discrete simulation model get together with the results of other analysis which is built on a interconnection network as an arbitration system. The advantage of the described analysis is also that it is independent from the system architecture.

The main disadvantages of this analysis is that Tm must be presumed to find out the exact number of packets in the network. However, the exact value of Tm parameter has no essential influence on the network optimization and logic design.

2.4. MULTISTAGE N-CUBE OPTIMIZATION

Multistage networks like OMEGA, DELTA, BUTTERFLY, FLIP and BINARY N-CUBE are well known interconnection networks[14]. The main advantage of these networks is that a processor can access every memory module in Log_N steps. The main disadvantages of these networks is the number of crossing links which increase with every new level and the processor inability to access local memories in less than Log_N steps.

The FLIP network (figure 2.4.1.) has an isomorphic characteristic which enables a transformation from FLIP's multistage structure into Log_N dimensional n-cube structure[1].

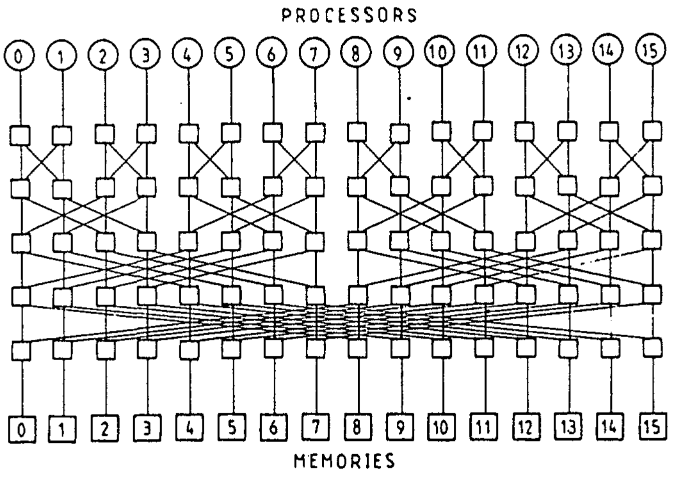


Figure 2.4.1.

A n-cube vertex consists of a processor, the arbiters under that processor and a global memory module under that arbiters.

A packet, which arrives to the arbiter, is stored in its memory block. After that a packet's route is determined by its routing tag. If the route is ready for the transfer, the packet is copied in the neighboring arbiter, processor or global memory module. Only one packet can be carried from the source (arbiter's memory block) to the sink (neighboring memory block) in the one arbitration cycle. The arbitration cycle's length is named an arbitration time (Ta) which is proportional with the average packet delay in the network (Tn). The relation between these characteristic parameters follows from a complex statistical analysis of arbitration systems. But only the fact that the arbitration time has to be as short as possible is important for the network logic design.

Some calculations are given in this chapter to be used as the direction for the final optimization. All calculations are made for 64 multiprocessor system.

A physical and logical ampleness, which is an optimization subject, is mostly depended on:

- one or bidirectional packets buses,
- packet buses width,
- the number of the arbiters and memory blocks,
- the kind of memory blocks.

Every vertex is connected with Log_2N neighboring vertices. The number of all one-directional packet buses is:

$$N * Log_2N = 384 \text{ (buses)} \qquad 2.4.1.$$

The bidirectional bus needs the arbiter to apportion the bus to one of the sources. The number of all arbiters can increase by:

$$N * Log_2N \;/\; 2 = 192 \text{ (arbiters)} \qquad 2.4.2.$$

The one-directional bus requires more links in the bus, but the logic ampleness will decrease because the additional arbiters are not required.

Therefore, the read packet should include:

- 32-bit address or data,
- routing tag ($Log_2N = 6$ bits),
- direction indicator (1 bit) and
- read or write mark (1 bit).

The write packet consists of:

- 32-bit address,
- data (32 bits),
- routing tag ($Log_2N = 6$ bits),
- direction indicator (1 bit) and
- read or write mark (1 bit).

The maximum packet's width is 72 bits and it has to be equal the packet buses width. But the buses width is halved to achive a propitious rate between the network efficiency, logic ampleness and number of links. Therefore, the write packet is divided into two words:

The first word contains:

- routing tag,
- direction indicator and
- read or write mark,
- address.

After that, the second word is sent to the memory block and it consists only:

- data part.

The number of arbiters in the vertex is very high, so the reduction is required, too. The next architectures has been analyzed:

- Log_2N stage n-cube (figure 2.4.2.),
- $Log_2N / 3$ stage n-cube (figure 2.4.3.) and
- singlestage n-cube (figure 2.4.4.).

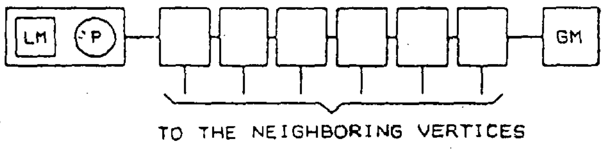


Figure 2.4.2.

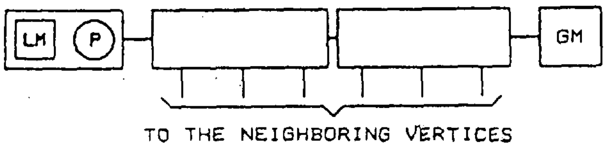


Figure 2.4.3.

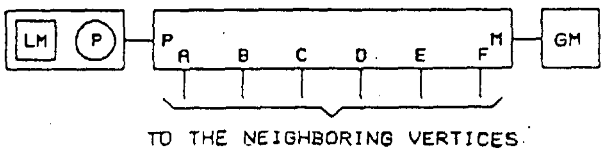


Figure 2.4.4.

The number of arbiters is:

- $N * Log_2N$ (arbiters)

  for Log_2N stage n-cube,

- $N * Log_2N / 3$ (arbiters)

  for $Log_2N /3$ stage n-cube and

- $N * 1$ (arbiters)

  for singlestage n-cube.

2.4.3.

The number of memory blocks is equal to the number of all arbiter sources or sinks and it is:

- $3 * Log_2N * N = 1152$ (memory blocks)

  for Log_2N stage n-cube,

- $5 * Log_2N / 3 * N = 640$ (memory blocks)

  for $Log_2N /3$ stage n-cube and

- $8 * N = 512$ (memory blocks)

  for singlestage n-cube.

2.4.4.

A following estimation could be given between the average packet delay, the arbitration time and the architecture:

- $Tn = Log_2N * Ta$

  for Log_2N stage n-cube and

- $Tn = Ta * Q$

  for singlestage n-cube.

  where is $Q > 1$. 2.4.5.

The estimation is based on the multistage networks without dead locks and races. If almost all interested data for processor i are placed in the memory module i, the Q is kept low.

The number of memory blocks is much higher than the average or maximum number of packets in the network (figure 2.3.2., figure 2.3.3. and equation 2.4.4.). Also the physical and logical ampleness decreases with the architection reduction (equation 2.4.3. and equation 2.4.4.). The third indicator is equation 2.4.5. which shows that Ta could be higher in the singlestage n-cube.

It is evident that singlestage n-cube is the most appropriate architecture and its efficiency is depended on:

- data disposition in the global memory modules,
- packets' conflict in the network.

Programs' adaptations to the architecture could mostly sooth this problems, so the single-stage n-cube is used for the network logic design.

Registers are used as memory blocks, because the number of all memory blocks is much higher then the maximum or average number of packets (figure 2.3.2., figure 2.3.3. and equation 2.4.4.). The register is formed of two blocks, because it has to store all packet and not only one word. If other way had been used, the logic ampleness would enormously increase. A solution is given in the figure 2.4.5..

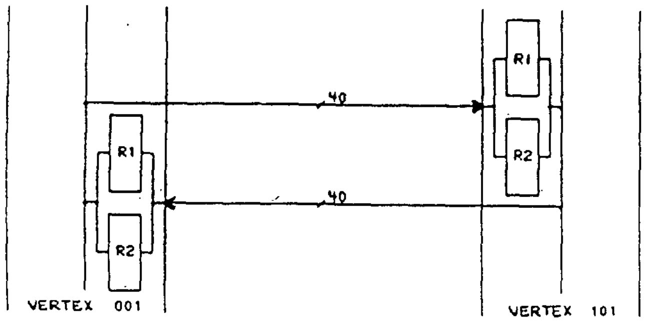


Figure 2.4.5.

At the beginning of the arbitration cycle the first word is carried in the block R1. After then the words are changed on the packet bus and after all the second word is transferred in the block R2.

Let's resume the optimization process and define the routing node' features:

- the network burden is lower if the asynchronous mode is used (equation 2.3.2. and figure 2.3.3.),
- Tn is proportional with Ta (follows from the arbitration theory)
- one-directional buses should be used,
- the buses width is 40 bits, therefore the write packets are divided into two words,
- singlestage n-cube is applied and
- registers are used as memory blocks.

3. ROUTING NODE LOGIC DESIGN

The routing node contains one arbiter which has eight sources and sinks. It is formed of:

- eight memory blocks,
- a synchronization logic,
- an arbitration logic and
- a termination logic.

A packet which comes to the routing node is stored in the register. After then the packet route is determined by the routing tag. If the next memory block on the packet route is empty, the bus request is generated. All bus requests from the sources are indicated in the synchronization logic. The arbitration logic chooses one of the bus requests and enables the packet transfer. The termination logic ends the arbitration cycle.

The routing node is built by 74F and 74AS logic families which are often used in application to achive high system performance. But on the other hand they are easy for designing because many equipment and software tools have been developed for these integrated circuits.

3.1. MEMORY MODULE

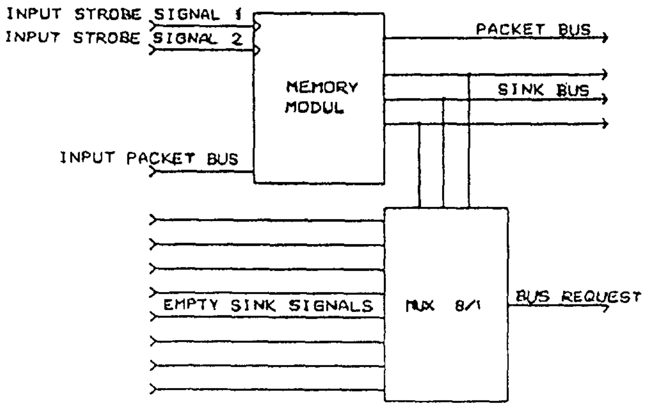


Figure 3.1.1.

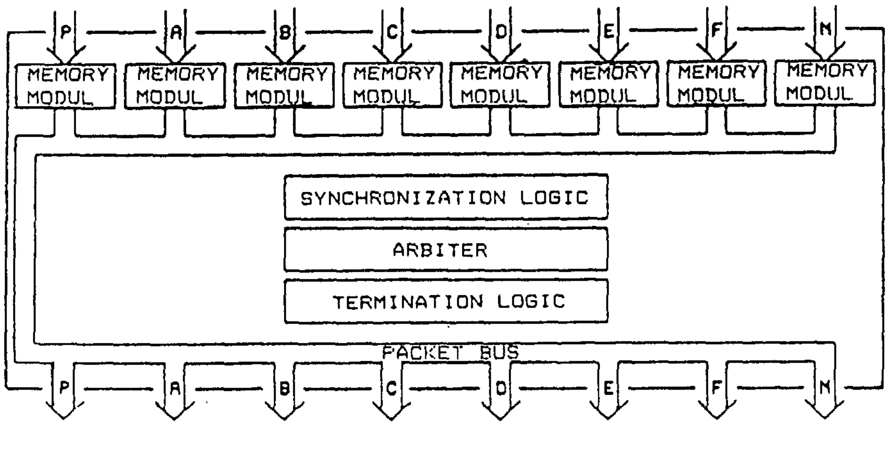


Figure 3.1. Routing node

The functions of the memory block are:

- to store packet in the registers,
- to determine the packet's length and directions,
- to define the sink,
- to generate a bus request.

A packet is written in the register by the input strobe signals. The sink is defined by the routing tag and a code of that memory block (P, A, B,..).

The arbiter on the level i is shown in the figure 3.1.2..

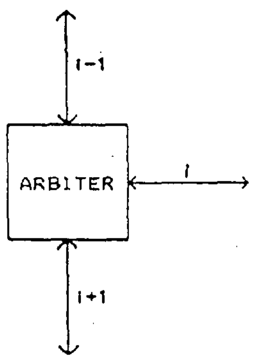


Figure 3.1.2.

When the packets travel from processors to memory blocks, the source can be the arbiter on the level i-1 or the neighboring arbiter on the level i.

For the source i-1 the sink is defined by the r_i bit (r_5 r_4 r_3 r_2 r_1 r_0 is the routing tag):

r_i = 0; the sink is the arbiter on the level i+1 and

r_i = 1; the sink is the arbiter on the level i.

For the source i packets are transferred to the level i+1.

When the read packets return to processors, they reach the arbiter on the level i through the source i-1 and i.

For the source i+1 the sink is determined by the r_i bit:

r_i = 0; the sink is the arbiter on the level i-1 and

r_i = 1; the sink is the arbiter on the level i.

For the source i packets are carried to the level i-1.

The routing tag is defined in the processor and it is:

routing tag =

= processor num. XOR global memory module num.

But the singlestage n-cube requires an extensive analysis of the routing tag.

Let's look the following example for the Log_N stage n-cube:

A packet from the processor eleven travels to the global memory module three. The routing tag is :

1011 XOR 0011 = 1000.

A packet route is shown on the figure 3.1.3..

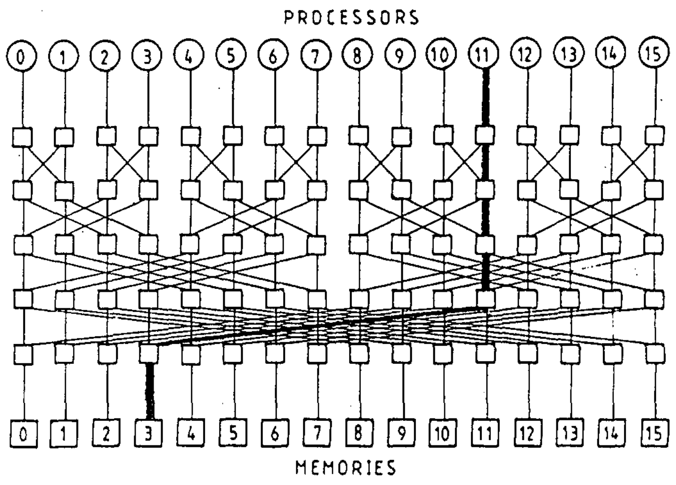


Figure 3.1.3.

The packet travels straight down through the arbiters, and it turns on the last level arbiter. However, in the singlestage n-cube, with only one arbiter between a processor and memory module, the packet is directly transferred to the last level - on the sink D, where it is received by the neighboring routing node (Figure 2.4.4. for 16 P - there is no sinks E in F).

Therefore, for the source P, the pattern of successive zeros from the r_0 inclusively in the routing tag has meaning for further route inside the node (the route P -> GM). The source A can send packets in the both directions. For the route P -> GM is important the pattern of successive zeros ahead the r_0 and for the route GM -> P all packets are carried to the sink P. For the source B is important the pattern of successive zeros ahead the r_1 for the route P -> GM and the content of the r_0 for the route GM -> P. Let's explain the routing tag in the source D for the complete notion. The source can also send the packet in two directions. For the direction P -> GM is important the pattern of successive zeros ahead r_3 and the same pattern from r_2 backwards (the route GM -> P).

The function:

SINK = f(SOURCE, DIRECTION, ROUTING TAG)   3.1.1.

is given in the [11].

Functions like that are usually calculated by the state machine and the algorithm for it is shown in the figure 3.1.4..

Because state machines are very slow, the function is calculated by the decoder. The decoder has also to code the sink and to define the packet length. The sink code is separated from the sink bus by three-state gates. The sink has to be calculated very fast, because the bus request needs this information for its activation. The bus request is generated by the MUX. The MUX is addressed by the sink code and its inputs are signals which define if the sink is empty or not.

```
begin
     i:=f(level);
     case direction of
          "F":begin
                   sink:=source+1;
                   a:=0;
                   while r(i+a)=0 do
                        begin
                             sink:=sink+1;
                             a:=a+1
                        end
              end;
          "B":begin
                   sink:=source-1;
                   a:=1;
                   while r(i-a)=0 do
                        begin
                             sink:=sink-1;
                             a:=a+1
                        end
              end
     end
end.
```

Figure 3.1.4.

To achive high activation speed the bus request is generated on two different ways:

M1 = input strobe signal *

* f(SOURCE, DIRECTION, ROUTING TAG) *

* empty sink and

M2 = full memory block *

* f(SOURCE, DIRECTION, ROUTING TAG) *

* empty sink.

Therefore:

bus request = M1 V M1.

M1 is active when the packet is being writing in the memory block while M2 conforms and keeps the bus request active.

3.2. SYNCHRONIZATION LOGIC

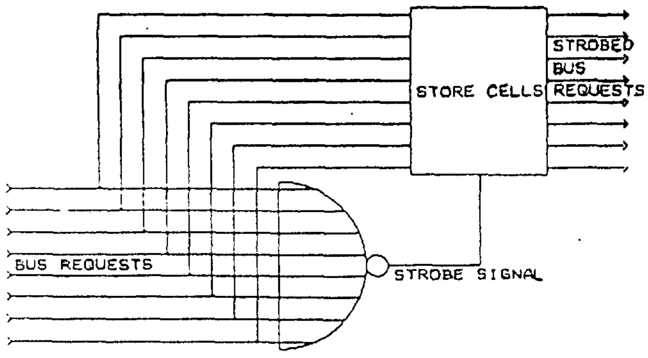


Figure 3.2.1. Synchronization logic

The synchronization logic is used to assure one decision in one arbitration cycle. To achive this purpose the logic has locked after the first bus request had come.

New request may come between locking but it is not important if they go through the logic or not. All bus requests which come later are waiting for the next arbitration cycle. If the lock does not exist, more than one decision can be possible in one arbitration period. This is undesirable in any case.

There are two possibilities how to make the lock:

- synchronous lock (the sampling is periodical) and
- asynchronous lock (the sampling is caused by the becaming signals).

The asynchronous lock is much faster and its application in the synchronization logic is shown on the figure 3.2.2.).

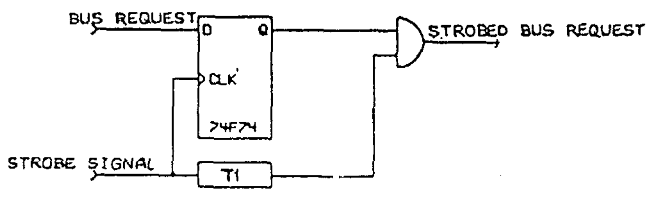


Figure 3.2.2.

A sampling signal is a disjunction of all bus requests. The next sampling is prohibited until only one bus request helds up in the active state. Because of that all bus requests can only be deleted by the termination logic.

The OR-gate has also a task to make a delay between the sampling signal and the bus requests, so this way the T-set is assured.

All sampling bus requests are conformed by the control signal, which is delayed for T1 seconds. The control signal delay has to prevent only the oscillations of sampling bus requests. Because packets which are late can wait for the next decision cycle or can be considered in this arbitration. Because of this reason the delay is only a bit longer than the maximum propagation delay through the cell.

3.3. ARBITRATION LOGIC

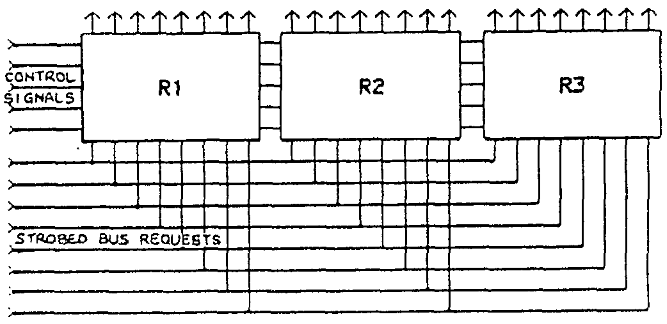


Figure 3.3.1.

An arbitration logic chooses one of the sampling bus requests and allows the packet transfer. A full source which has been processed in the last arbitration cycle has no priority ahead the other full sources. Information about the last processed source is stored until the next arbitration cycle. If more then one bus requests have occurred, the nearest right neighbor of the last processed source in a chain will be chosen. The chain is shown in the figure 3.3.2..

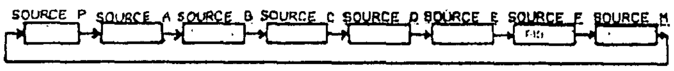


Figure 3.3.2.

If the source B is empty, the source C follows the source A.

The fast arbitration is achieved by three parallel decision logic. This is necessary because the logics are activated in different time.

They are:

- R1 enables the first word transfer with opening the three-state gates on the packet bus,
- R2 enables the second word transfer with opening the three-state gates on the packet bus and
- R3 is active across all arbitration cycle and it enables the sink code to occupate the sink bus.

3.4. T E R M I N A T I O N L O G I C

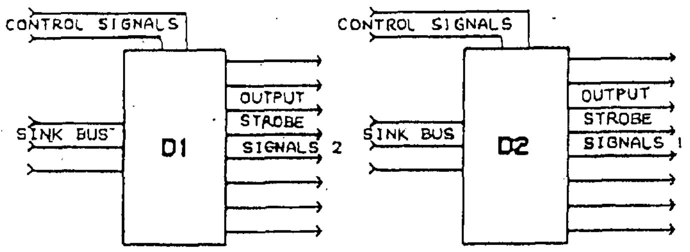


Figure 3.4.1.

A termination logic generates output strobe signals and end signals.

The first part is made of decoders which transform the sink code in output strobe signals:

- D1 generates output strobe signals for the first word and
- D2 generates output strobe signals for the second word.

The output strobe signal returns to the termination logic after it write the packet into the sink. This signal starts an output strobe signal·for the second word or ends the cycle with termination signals. This part is made of three MUXes:

- MUX1 generates control signals for activation output strobe signals or for termination the cycle,
- MUX2 terminates the arbitration cycle and
- MUX3 defines how long the packet is.

If the output strobe signal does not return, the node alarms the processor.

The termination signals are used to:

- reset the synchronization logic,
- reset the arbiter,
- deactivate output strobe signals and
- delete a treated bus request.

3.5. T I M I N G

The exact logic and electrical design and the detail timing description is given in [11]. For the further experties and understanding the following is only important:

- the arbitration cycle is composed of the following sequences:
  - a packet defination,
  - a synchronization sequence,
  - an arbitration,
  - a first word transfer,
  - a changing the words on the packet bus,
  - a second word transfer and
  - a termination.
- each sequence is executed in approx. 20 ns, so one worded packets are transmitted in approx. 80 ns and two worded packets are copied in approx. 140 ns.

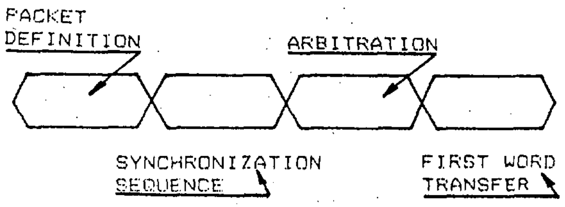


Figure 3.5.1. The first word transfer

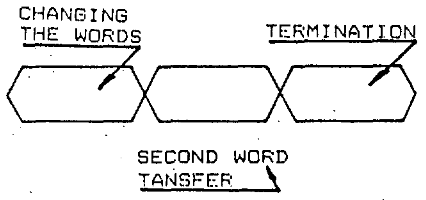


Figure 3.5.2. The second word transfer

4. C O N C L U S I O N

This article explains that network's features mostly depend on the choosen architecture, used technology, internal functional organization and packet bus constructions.

The way to more efficient multiprocessor network leads to high parallel structure like cross-bar switch and fully connected networks. On the internal functional organization and bus construction field new principles must be developed to achive pipelining and interleaving in the arbiter. Faster technology brings many electronic problems which can be solved by twisted-pair cables, coaxial cables, termination circuits, on-chip connectors and a precise design. Semi or full-custom design integrited circuits will be applied to increase reliability,to protect invested knowledge and money.

Further experties on the project will give the answer how to unite this suggestions to achive high performance interconnection network.

A C K N O W L E D G M E N T

It is not just my merit that the paper has gotten this content and form. I wish to thank Peter Brajak, who has introduced me to parallel processing, has led me through the work and who has been ready to give a right advice at every time. Furthermore, I wish to thank Lojze Vogel and Marko Kovačevič for useful discussions on hardware design, as well as Dr. Sasa Prešeren, dr. A. P. Železnikar and others members of the PARSYS group.

R E F E R E N C E S

[1] Brajak P., Designing a Reconfigurable Intelligent Memory Module for performance enhancement to large scale, general purpose parallel processor, Informarika 1, Jan. 1987, page.:19-53.

[2] Hitij P., Simulator paralelnega računalnika PARSYS, Diplomsko delo, Dec. 1987, page.:12.

[3] Mackenna C., Metastabilty and Synchronization, Dec. 1987, D1-D10.

[4] Howell D., IC master, 1987, page.:2/3701.

[5] Motorola, Schottky TTL data book, 1984/85, page.:6-21.

[6] Bantz D. F., Computer system design notes, 1980, page.:I.12.

[7] Lee G., Some issues in general-purpose shared memory multiprocessing: parallelism exploitation and memory access combining, Jun. 1986, page.:82-106.

[8] Data Delay Devices, Catalog 87, 1987, page.:7, 14, 17, 36, 51.

[9] Advanced Micro Devices, Publication #08601, 1986, page.:4-99.

[10] Monolithic Memories, LSI Databook, 1987, page.:2-29, 2-30, 2-33, 2-36.

[11] Žagar A., Analiza in sinteza multiprocesorske mreže, Diplomsko delo, Sep 1988.

[12] Prešeren S., Brajak P., Vogel L., Železnikar A., A Massively Parallel Computer System Project in Yugoslavia, ISMM Inter. Conf. on Mini and Microcomputers, Florida, Dec. 1988.

[13] Brajak P., Prešeren S., Vogel L., Scheduling and Synchronization Concepts of the Parsys Parallel Processor, ISMM Inter. Conf. on Mini and Microcomputers, Florida, Dec. 1988.

[14] Siegel H.J., Interconnection Networks for Large-Scale Parallel Processing, Lexington Books, 1985

A P P E N D I X I

Simulation program for determining the number of all packets in the network (Np) after the transitional phenomenon.

```
      DEFINE THE INITIAL CONDITION
      CHOOSE A,B,C
      DO 10 I=1,10*(A+B+C+B)
      CALL SIMULATION
10    CONTINUE
      EQUATION 2.3.1.
      WRITE Np
      END
```

The core of the simulation program:

```
C..
C..  SIMULATION SUBROUTINE
C..
      SUBROUTINE SIMULATION
      INTEGER A,B,C
      COMMON A,B,C,WPGM(121),RPGM(121),
     *RPCPU(121),CPU(121),GM(121)
C..
C..   SUMMATION AND SPLITTING POINTS
C..
      GM(1)=RPGM(B+1)
      RPCPU(1)=GM(C+1)
      CPU(1)=RPCPU(B+1)+(19./20.)*CPU(A+1)
      RPGM(1)=(1./20.)*CPU(A+1)
      WPGM(1)=(1./20.)*CPU(A+1)
C..
C..   DELAYS
C..
      DO 10 K=B,1,-1
      WPGM(K+1)=WPGM(K)
      RPGM(K+1)=RPGM(K)
      RPCPU(K+1)=RPCPU(K)
10    CONTINUE
      DO 20 K=C,1,-1
      GM(K+1)=GM(K)
20    CONTINUE
      DO 30 K=A,1,-1
      CPU(K+1)=CPU(K)
30    CONTINUE
      RETURN
      END
```

A P P E N D I X I I

Program for calculating the maximum number of all packets in the network (Npm):

```
      DEFINE INITIAL CONDITION
      CHOOSE A,B,C
      Npm=0.
      DO 10 I=1,10*(A+B+C+B)
      CALL SIMULATION
      EQUATION 2.3.1.
      IF (Np.LT.Npm) GOTO 10
      Npm=Np
10    CONTINUE
      WRITE Npm
      END
```

Analiza in sinteza multiprocesorske mreže

Članek obsega pregled razvoja multiprocesorske mreže paralelnega računalnika PARSYS. Analiza mreže je podana v prvih dveh poglavjih. Kot osnova služi statistično-diskretni simulacijski model, ki se pokaže kot zelo dober pokazatelj razmer v mreži. Na teh temeljih je v tretjem poglavju zasnovana krmilna enota.

INFORMATICA 2/89

AN ADAPTABLE PARALLEL SEARCH OF KNOWLEDGE BASES WITH BEAM SEARCH*

Keywords: expert system, knowledge base, beam search, parallel search

S. Prešern, P. Brajak, L. Vogel and
A. P. Železnikar
Iskra Delta, Ljubljana

This paper describes an on-line computer supported expert system, which is designed for searching complex knowledge bases. Searching is performed in parallel with beam search, so that only a limited number of specific features are examined on each level. Parameters for beam search are modified during application of the software package, so that self adapting capability of the beam search through an expert system is incorporated in the design of a package. The proposed approach is especially effective for image processing and speech recognition. The difference between making a sequential program parallel and using natural parallelism is shown. The proposed method on a mesh network of Transputers or on a MIMD parallel computer PARSYS has properties of a neural network.

1. INTRODUCTION

The need for more powerful computers is growing faster than technology. Parallel processing is a very powerful solution to many processing-intensive applications as for example image processing, numerical problems, artificial intelligence, speech recognition and others. Even though many parallel computers are already on the market, most software approaches are based on von Neuman logic. Many sequential algorithms are simply parallelised for a parallel environment, which means not using all the capabilities and approaches of parallelism. Other different solutions have also been proposed, as for example a neural network for visual pattern recognition (3), which has also a self-organizating capability of the network, the neural network processor for speech recognition (5), and others.

Instead of rewriting sequential programs to a parallel form, we suggest approaching the problem in a different way. The natural parallelism of the problem has to be explored and used directly in a parallel computer system. In natural parallelism, a problem is specified by a set of objects and relations between those objects. The pool of processing elements performs the task in a self organized fashion so that available processors perform the task. Using an expert system in the background and enabling adaptability of the system to the environment, we propose a new approach to complex application software approach.

An expert system is assumed as a support in our parallel computer system applications for a parallel search of a knowledge base. Most complex applications should interact with the changing environment, which means a need or capability to react and adapt to those changes. The role of an expert system is to enable easy man-machine interfacing and to enable adaptability to the changing environment. Such adaptability is called genetic adaptability.

(*) This paper was originally presented at the 22-nd Annual Hawaii International Conference on System Sciences, Hawaii, Jan. 1989

2. KNOWLEDGE BASE

The knowledge base is the fundamental framework for the description of the domain of the problem. It represents the core of the system and contains all relevant domain-specific information, permitting the system to behave as an intelligent specialist. A domain might be a set of industrial parts or objects (9) (typically for automatic recognition of industrial objects in robotics), analysis of different properties by testing (printed board testing, chemical testing) or socio-economic properties of interacting subjects (in behavior simulation, macroeconomic model simulation) etc. The knowledge base changes its content and connectivity as new information is added to the system. Therefore we have a dynamic system for performing cognitive tasks. Information processing in dynamic systems is studied by Harmony theory (11).

2.1. Elements of a Knowledge Base

In order to represent a knowledge base we must have a symbolic description of the properties of the objects which are connected by relations.

A domain oriented network consists of n objects. Each object o(i) forms a knowledge atom in a knowledge network and is represented by a vector of m properties P(i,j)

$$O(i) == P(i,j) \qquad 1 >= i >= n, \quad 1 >= j >= m, \quad m =< n$$

All objects form a P matrix of properties

$$P = \begin{array}{cccc} P(1,1) & P(1,2) & \ldots & P(1,m) \\ P(2,1) & P(2,2) & \ldots & P(2,m) \\ . & . & \ldots & \\ . & . & \ldots & \\ P(n,1) & P(n,2) & \ldots & P(n,m) \end{array}$$

Recognition includes performing a set of measurements and feature extractions on an unknown object in order to analyze some properties of the object. The identification process consists of a set of feature extractions F. A set of feature extractions F consists of h feature extractions:

$$F = (f(1), f(2), \ldots, f(j), f(j+1), \ldots f(h))$$

The distinction between levels of description must be clear: a microlevel involving knowledge atoms, and a macrolevel involving individual features by themselves. The relationship between those two levels is very important. The knowledge atoms are determined by the nature of the objects and form the problem domain. On the other hand, the automatic cognition process always detects only a single representational feature.

knowledge atoms P(1,1)P(1,2)...P(1,m) P(n,1)P(n,2)...P(n,m)

PROBLEM DOMAIN

representational features V(1,1)V(1,2)... V(i,j),...V(k,n-1)V(k,n)

COGNITION PROCESS

Fig.1. Knowledge atoms form a problem domain and representational features are detected in cognition process

The result we get after performing a single feature extraction f(j), is one of the previously stored values. This is the representational feature V(j,i) of the knowledge atom. We say that a feature extraction f(j) consists of k elements V(j,i)

$$f(j) = (V(j,1), V(j,2), \ldots V(j,i), \ldots V(j,k))$$

The probability of obtaining the value V(j,i) by feature extraction f(j) equals p(j,i). All probabilities p(j,i) (1=<i=<n) for realization of an event A(i) by feature extraction f(j) form a set P(j)

$$P(j) = (p(j,1), p(j,2), \ldots p(j,i), \ldots p(j,k))$$

with a property

$$\sum p(j,i) = 1 \qquad 1 >= i >= n$$

A different feature extraction f(j+1) consists of g different elements V(j+1,i)

$$f(j+1) = (V(j+1,1), \ldots V(j+1,i), \ldots V(j+1,g))$$

with probabilities p(j+1,i) for realization of a representational feature V(j+1,i)

$$P(j+1) = (p(j+1,1), \ldots p(j+1,i), \ldots p(j+1,g))$$

All objects o(i) (1=<i=<n) which are treated by a computer supported sensing system form a set N

$$N = (o(1), o(2), \ldots, o(n))$$

The representational features of all objects o(i) are represented in a matrix form

	f(1)... f(h)
o(1)	V(1,1)...V(h,1)
o(2)	V(1,2)...V(h,2)
.	
.	
o(n)	V(1,n)...V(h,n)

where each row represents a symbolic description of an object o(i) and each column represents possible outcomes of each feature extraction f(i).

The problem of object recognition is solved when an unknown object o(i) is identified as one element of the set N.

All representational features V(i,1), V(i,2),... represent the cognitive system's representation of possible states of the environment with which it deals. In the environment of object recognition, these features are for example individual pixels, edges, syntactic description of an image, holes and identification of objects. In medical diagnoses features are symptoms, outcomes of tests, diseases, prognosis and treatments. In circuit analysis the features are high current through some resistor, high voltage on certain pin, or low voltage at a transistor.

A knowledge network also requires connections between knowledge atoms and representational features. These connections can form a single a or multilevel network.

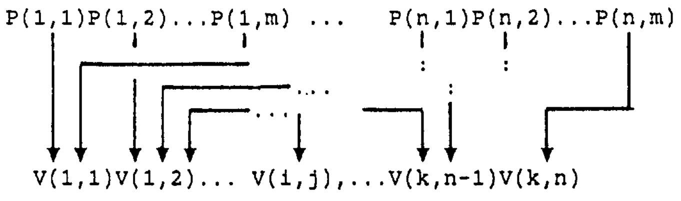

P(1,1)P(1,2)...P(1,m) ... P(n,1)P(n,2)...P(n,m)
V(1,1)V(1,2)... V(i,j),...V(k,n-1)V(k,n)

Fig.2. Configuration of a single level network

Symbolic representation of each knowledge atom is simply a value vector V of 1 and 0 values which denote on and off connection between the corresponding nodes.

```
V(O(1)) => (0  0  1        1  0        0  0)
V(O(2)) => (0  1  0        1  0        0  0)
   .
   .
V(O(n)) => (0  0  0        1  0        1  0)
```

The first index in a representational feature means the type of feature extraction and the second index is a value of a particular feature extraction.

2.2. Construction of the Knowledge Network

The upper level of a knowledge base is a set of production rules determining connections between atoms and representational features together with atom definitions. Each rule consists of a precondition and an action. The precondition contains Boolean combinations of clauses, each of which applies a predicate to an object in the system and tests its value. The action of a rule gives a new description of a representational state that can be drawn if the precondition is satisfied.

Knowledge source:

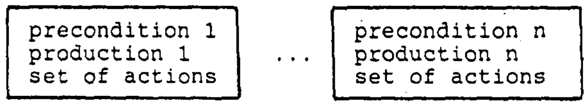


Fig.3. Knowledge source consisting of preconditions, productions and actions

A representational feature has the role of a pattern, which is a precondition to the production (Fig.3). Each production is a selection of appropriate routing, based on present and past representational states. The set of actions includes pruning of a search tree and update of the representational state.

The use of pattern based knowledge lends itself to production rules. Whenever a pattern is matched, a corresponding action is taken which enables the sensing-based recognition system to act on the information obtained by matching the pattern. Each pattern becomes the condition of a production rule and is associated with a certain action. These patterns have to be matched in parallel. A pattern is data which are obtained by feature extraction. Therefore, a set of patterns for each object is actually a symbolic description of the object's features.

The general structure of the knowledge source which is used by the planner to identify an object, is that sensor data are obtained and that the pattern is used to choose the production rule. Therefore a planner consists of a pattern and a production rule:

PRODUCTION
- get a pattern

PRODUCTION RULE
- update the set of possible objects
- update the set of possible productions
- choose the next cheapest action
- perform the choosen action.

Let us form a network for knowledge representation. On the highest level there is the whole set of possible representational features V. After the first feature extraction is performed, the possible representational features are:

1st feature extraction:
$V(1,1), V(2,1), \ldots V(s,1)$

Some of those representational features have the same value and most knowledge atoms can not be distinguished one form another. Therefore a sequence of feature extraction has to be performed in order to find a knowledge atom and identify an object.

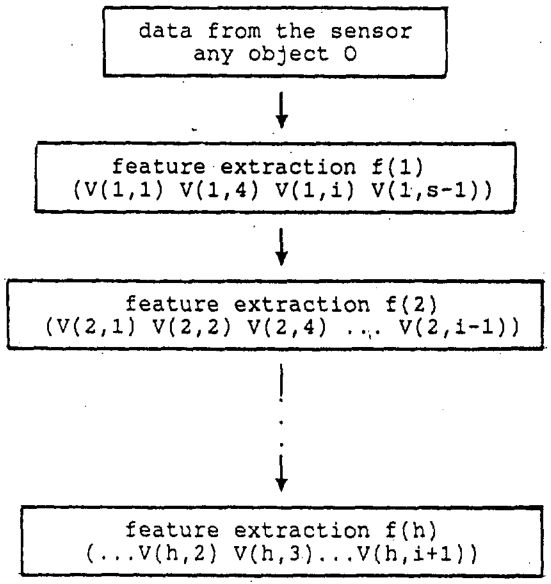


Fig.4: A sequence of feature extractions in a sequential program

The parentheses below each measurement state the possible representational features. In a sequential activation of a particular feature extractor this sequence of feature extractions forms a knowledge representation which is the basis for a search tree.

The spanning of the knowledge tree on this level is equivalent to the number of different values of representational features (Fig.5).

The knowledge atoms are placed on the lowest level of the knowledge network.

This organization is appropriate for a nonparallel approach. The recognition is performed by comparison between the vector of properties P(i) defining the knowledge atom and the representational feature vector V(O(i)). In our task of cognition we perform pruning in stages. The goal is reached when a particular representational feature has an active connection to only one knowledge atom. This is a part of the decision-making process in Harmony theory.

The knowledge network is organized as a collection of hierarchical descriptions where each level in the hierarchy represents a

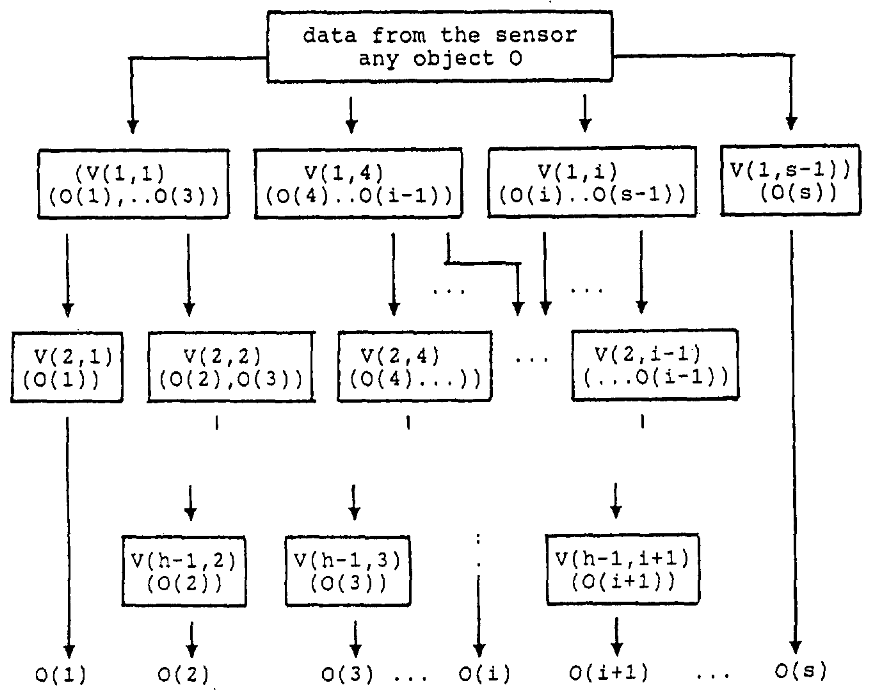


Fig.5. Knowledge representation for object recognition

different conceptual abstraction of the object symbolic representation. Spanning of the knowledge network is not a problem if parallel processing on a parallel computer with shared memory is proposed. If a mesh architecture with message passing is used, then a mapping of the knowledge network to the mesh architecture has to be done.

Analysis of different properties and relations in the knowledge network requires a different complexity and therefore a different amount of computer time. The strategy or sequence of operations is very important. The strategy is determined by the knowledge network organization. Therefore, the final step in knowledge network construction is optimization, so that the shortest average time is required for object recognition.

3. NATURAL PARTITIONING AND MAPPING TO THE PARALLEL COMPUTER

The process of object recognition requires an adaptable algorithm which has to be mapped to a mesh network of Transputers. The mapping is required when the communication graph of a parallel algorithm differs from the interconnection architecture of the physical parallel machine. Several solutions have been proposed for mapping algorithms to CHiP machine (2) array processors (6), Star computers (7) and others.

For a given harmony model, every node in the network is mapped to one processor, and every link in the network becomes a communication link between two processors. The processors each have two possible messages with values for the representational feature processors; 1 = active and 0 = inactive. The completion is denoted by the code identity number at any processor.

The knowledge network consists of features which are describing objects. Each feature extractor is, in terms of parallel search, a software process. A knowledge network is designed as an interconnected set of processes. Each process is an independent software unit and a feature, forming the world representation. Depending on the result of the present state, a new set of actions is performed. This means that the result on each level guides the search and therefore a self routing is performed in real time. This means that each process communicates with other processes along four channels, limited by the physical design. A logical design is hidden and is specified by the messages between processors.

3.1. Parallelization

In general a knowledge network permits any number of links between different objects which is acceptable for a shared memory parallel computer system. But the Transputer system is a typical message passing parallel computer system limited to four neighboring links and forming a mesh architecture. Therefore a mapping has to be performed in order to:

- limit the number of links to four
- avoid backward links
- reduce the unnecessary links.

In the case of object recognition, processes running in parallel on different Transputers represent feature extraction. Communication is achieved by message passing to communication channels. Massages have to be sent to only those processors with the correct identity number. The code in each Transputer has four sections:

- a feature extractor
- the pruning of a search tree
- a decision which feature to extract next
- the sending of a message to the neighbor to perform the next feature extraction.

In our model a general knowledge is stored in a long term memory. Each level of long term memory contains symbolic descriptions of the physical properties of objects in a representational feature vector. Primitive object elements are organized as levels of different classes. Each feature extractor f(i) is a process on one processor. Depending on a result of feature extraction, the correct branch in a network is selected (Fig.6).

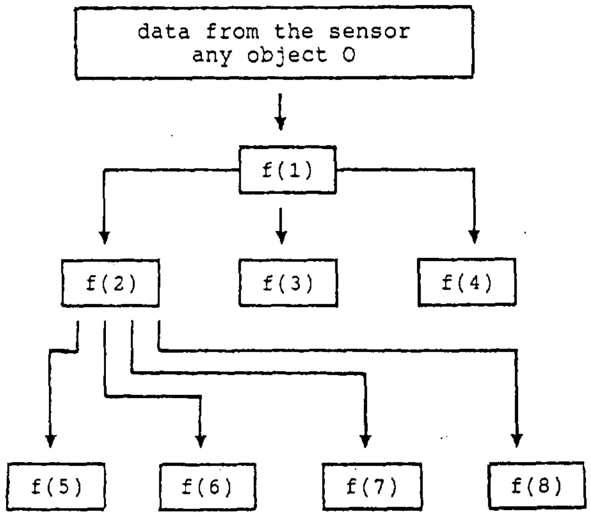


Fig.6. The original tree of feature extractors or processors

The symbolic representation, formed in the knowledge network, is therefore mapped onto the physical network of Transputers. All network interconnections are limited to four physical connecting links. A grammer-based description for generating embeddings of large binary trees in square processor array was suggested by Bailey (1).

In our mapping procedure some feature extractors have to be multiplicated for two reasons (fig.7):

(1) in order to enable enough connections, and
(2) in order to enable connections to dislocated feature extractors.

This transformation process can be done automatically in stages. Fig. 7 shows mapping of the original tree of processes from fig. 6 to the mesh architecture.

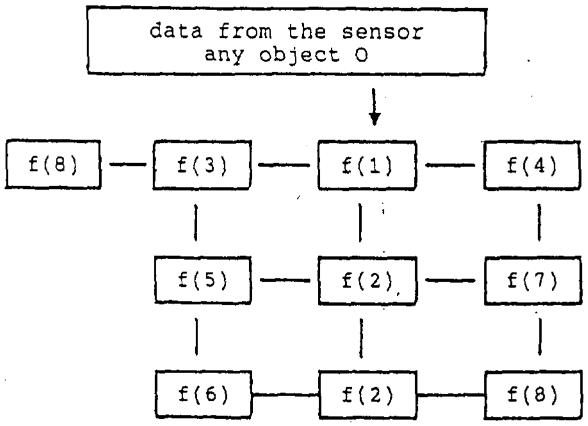


Fig.7. Mapping to the mesh architecture

The algorithms for computing tree functions for systems where a search tree is represented by a list of edges, so that each undirected edge is represented by two directed edges in this list, have been proposed by Gopalakrishnan (4).

The programming approach on Transputers is based on OCCAM language and the problem has to be divided into large grain parallelism. Within a process running on one Transputer even more fine grain parallelism can be achieved by using the opportunity for fast context switch provided by the hardware.

After every stage of feature extraction, the unknown object is better defined than on the previous stage. The process of identification is considered as a multistage recognition, where each feature is processed on a separate Transputer. The object recognition is in that sense similar to a bubble movement in a bubble sort. Some proposals for a bubble sort on an array of Transputers have been already proposed (8).

We have seen a parallelization approach where each processor is dedicated to one process. This approach is not very effective because Transputers are not loaded in balance and because a network can not have genetic properties. Therefore a pool of waiting processes has to be available in each Transputer. An internal processor utilization has to be measured for a large number of runs in order to get an average load balance. This is done with monitor processes or simulation processes to represent parts of the application program.

3.2. Natural Parallelism

Using the natural structure of a problem and the natural parallelization we partition the knowledge network, the feature extraction and search algorithm.

Natural partitioning simplifies the task of system design and programming. The knowledge network is transformed to a hierarchical logical structure, realized on a variable number of processors, depending on the size of a problem. By using natural partitioning, there is no contention for the communication mechanism, regardless of the number of Transputers in the system. The mapping of the problem to the computer hardware is not limited to the number of Transputers, because arbitrary size and topology can be constructed.

Using natural parallelism in a Transputer system, the input data, as for example the image, are broadcasted to the first column of Transputers. The whole first row is performing feature extraction in parallel and pruning is not necessary after each feature extraction. The required condition is that we have as many Transputers as we have processes for feature extraction (Fig.8).

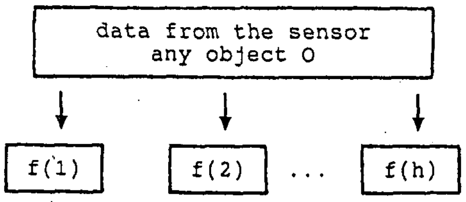


Fig.8. Parallel feature extraction

Each processor performs the feature extraction procedure and gets a vector describing which knowledge atoms have the same representational feature value as in the input. This representational vector is for example on processor 1 (0,1,1,0,0,0,0) which means that knowledge atoms 2 and 3 have corresponding value for the feature 1. The same procedure is performed on other processors (Fig.9).

knowledge atoms:	1 2 3 4 5 6 7	
processor 1:	(0,1,1,0,0,0,0)	F.E.1
processor 2:	(0,1,1,1,1,0,1)	F.E.2
processor 3:	(1,0,1,0,1,0,0)	F.E.3
processor 4:	(0,0,1,0,0,0,0)	F.E.4
AND	(0,0,1,0,0,0,0)	

F.E.# is the feature extraction #

Fig.9.: The representational vectors on 4 processors performing 4 feature extractions.

The unknown object is identified simply by making an AND operation on all representational vectors. A corresponding knowledge atom where an AND operation gives "1" is the identified object.

If the number of processors is smaller than the number of required feature extractors, then each processor has two or more feature extractors. One is in the running state, others are waiting. Comparison is performed after each cycle of feature extraction. Let us look at an example with 2 processors and four feature extractors (Fig. 10).

knowledge atoms:	1 2 3 4 5 6 7	
processor 1:	(0,1,1,0,0,0,0)	F.E.1
processor 2:	(0,1,1,1,1,0,1)	F.E.2
AND	(0,1,1,0,0,0,0)	

PRUNE THE KNOWLEDGE ATOMS

knowledge atoms:	2,3	
processor 1:	(1,0,1,0,1,0,0)	F.E.3
processor 2:	(0,0,1,0,0,0,0)	F.E.4
	(0,1,1,0,0,0,0)	AND P.S.
AND	(0,0,1,0,0,0,0)	

P.S. denotes previous step

Fig.10.: The representational vector on 2 processors performing 4 feature extractions.

The representational state of the cognitive system is determined by an AND logical operation of values for all the representational variables (V(i)), a so called representation vector.

If the number of processors is smaller than the number of processes, then a combination of tree pruning and parallel feature extraction has to be performed.

We see that in the case when the number of processes is bigger than the number of processors, the sequence of activating processors is important. If the sequence is different we might identify the object sooner. The upper example gives the answer already after the first run if features 3 and 4 are executed before the feature extractors 1 and 2. The arrival of particular type of objects is the criteria for determining the optimal sequence of processes execution. The criteria is to require minimal average time for object recognition. Therefore the history of identified objects has to be collected and ocasionaly the sequence of feature extractors has to be adapted if the changing environment requires so. This is a task for the expert system.

The proposed approach is very convenient also because most speed is gained simply by parallel execution of the whole feature extractor procedures and no recoding of algorithms is required.

3.3. Neural Network

Let us sum up the computer environment. We have a parallel machine with distributed processing capability, distributed memory and distributed knowledge. Our problem is a problem of fuzzy reasoning on a symbolic world structure. The system has properties of neural network as for example:

- Patterns (representational features) are not locally memorized but are memorized over the whole system in a sense that properties of an object are stored in different memory modules.

- Object recognition is performed through feature extraction, that means symbolic representation rather than memorizing the whole image. That is exactly the way that living systems work.

- The network is fault tolerant. In that sense, the system behaves like a neural network because if a part of the system is damaged or fails, the system still works, although the probability for wrong identification is higher.

- Recognition and learning are performed by local adaptation in the sense that a set of knowledge atoms is updated, a set of representational features is updated, and a set of distribution probabilities is locally updated. A new strategy is selected in real time.

- The dynamic of the system is parallel and asynchronous.

- The knowledge network contains elements which have a high percentage of fuzziness.

The nature of our expert system is parallelism. The system is composed of a number of modules, executed on different processors. The calling of the module depends on the data environment. The knowledge network including all steps of symbolic representation, normalization and optimization gives the framework for parallelism.

Future applications of parallel processing will not be based on static expert systems but rather on knowledge engineering. This is a shift from mere data processing to an intelligent processing of knowledge.

4. GENETIC PARALLEL SEARCH

The interesting activity in sixth generation computer projects is development of genetic algorithms which adapt to the changing environment.

In the second paragraph we have seen that strategy depends on knowledge network organization. Knowledge network organization depends on the optimization process which is described before. But the optimization process and therefore strategy strongly depends on the test pattern.

Let us see an example: we have a domain for object recognition of 100 industrial objects. Some objects do require very complex analysis. But in the recognition process only a subset of 5 objects appear and they are easily recognized by one typical feature. A conventional static object recognition system would perform always the same search designed for 100 types of knowledge atoms. But more flexible systems for future generation parallel computers will have the capability to learn and adapt the search algorithm according to the environment. Therefore the system has to keep track of the recent history of objects and accordingly adapt the search strategy.

In order to perform algorithm adaptability, each data has to have a weighting function describing its importance for problem solving, as for example object recognition. A method is proposed to acquire this global information by calculating probability factors that any unknown object is recognized fast.

A set of n objects has the distribution a(i) where a(i) is the percentage of objects a(i) in a set of n objects.

$$\sum_i a(i) = 1 \qquad 1 >= i >= n$$

The number of objects i in a set of expected domain equals N, so that

$$a(i) = \frac{N(i)}{\sum_i N(i)}$$

where N(i) is the number of objects i in the whole set of objects. At the same time this means that probability for an unknown object to be object i equals a(i).

For the whole set of n objects we get a probability vector

$$A = (a(1) \;\; a(2), \;\; \ldots \;\; a(n))$$

An adaptable parallel search means a capability of the system to change the search strategy dynamicaly according to the present state and history of the search. Strategy for further search depends on conditions which allow specifications of different plans for different data from the environment. The advantage is that the adaptable search can immediately try the correct plan instead of searching an appropriate plan and backtracking to correct itself.

A genetic computer must have the capability to modify itself by learning. The learning phase is denoted by three steps:

- assign values to representational features of a knowledge atom,
- activate connections to knowledge atoms that are consistent with the representation and
- confirm unique representation of a new knowledge atom

else

- suggest a new feature extraction and enlarge a dimension of representational state.

Present technology gives us an opportunity to perform an adaptable parallel search of knowledge bases on an array of Transputers.

The Transputer is programmed to perform a specialized function and is regarded as a black box thereafter. This specialized function is feature extraction and search information, depending on the present result of the feature extraction. The adaptable parallel search is performed by a network of programmable components which have the mesh topology, limited to four links to each Transputer.

5. CONCLUSION

We have discussed an adaptable parallel search supported by an expert system. This subject is a goal of the sixth generation computers which are based on a man-behavior intelligence or neural intelligence (12). Present research projects in parallel processing design have shown very good results in hardware design using a massively parallel structure with a high interconnectivity between large number of processors.

By speculation and having in mind living organisms we can predict future trends in genetic computing. The parallel system would perform a self balancing statistics so that processor utility is measured, bottlenecks are identified and rescheduling of strategy is performed.

Nowadays pattern directed systems are proposed as future software organization on parallel machines. We feel that even on parallel machine we need a deterministic fixed calling system but upgraded by a genetic capability to adapt, reformulate, reorganize according to performance measurements and to changing environment.

Not many new concepts in problem approach have been proposed with parallel processing. We feel that parallelization of sequential programs is only the first step and the bridge between sequential problem approach and parallel machines. But real parallelism on parallel machines has different problem formulation, partitioning and adaptability. The principles which were proposed in this paper show how to combine the sixth-generation computer project and artificial intelligence and are under development on a PARSYS parallel computer system (10). The approach is general enough even to be performed on any MIMD computer or on a Transputer based parallel machine which was demonstrated by examples.

Having this in mind we have designed a massively parallel computer system PARSYS with up to 64 powerful 32 bit processing units and up to 64 memory modules.

6. REFERENCES

(1) D.A. Bailey and J.E. Cuny, "An Efficient Embedding of Large Trees in Processor Grids", Proc. of the 1986 Int. Conf. on Parallel Processing, IEEE Competer Society, p. 819, 1986.

(2) F. Berman, "PREP-P: A Mapping Preprocessor for CHiP Computers", Proc. of the 1985 Int. Conf. on Parallel Processing, IEEE Computer society, p. 731, 1985.

(3) K. Fukushima, "A Neural Network for Visual Pattern Recognition", IEEE Computer, p. 65, 1988.

(4) P.S. Gopalakrishnan et al., "Computing Tree Functions on Mesh-Connected Computers", Proc. of the 1985 Int. Conf. on Parallel Processing, IEEE Computer society, p. 703, 1985.

(5) T. Kohonen, "The 'Neural' Phonetic Typewriter", Helsinki University of Technology, IEEE Computer, March 1988, p.11, 1988.

(6) T. Lin and D.I. Moldovan, "Tradeoffs in Mapping Algorithms to Array Processors", Proc. of the 1985 Int. Conf. on Parallel Processing, IEEE Computer society, p. 719, 1985.

(7) W. Lin and C. Wu, "Design of Configuration Algorithms for Commonly-used Topologies for a Multiprocessor - STAR", Proc. of the 1985 Int. Conf. on Parallel Processing, IEEE Computer society, p. 734, 1985.

(8) J. Modi and R. Prager, "Implementation of Bubble Sort and the Odd-even Transpozition Sort on a Rack of Transputers", Parallel Computing, Vol.4, No.3, June 1987.

(9) S. Prešern and L. Gyergyek, "An Intelligent Tactile Sensor - An On-line Hierarchical Object and Seam Analyzer", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.2, March 1983.

(10) S. Prešern et. al., "PARSYS - A Massively Parallel Computer System Project in Yugoslavia", ISMM Int. Conf. on Mini and Microcomputers, Florida, 1988.

(11) D. E. Rumelhart et al., "Parallel Distributed Processing", MIT Press, Cambridge, Massachusetts, 1986.

(12) B. Souček, "Neural and Massively Parallel Computers: The Sixth Generation", John Wiley & Sons, Inc., 1988.

SURVEY OF THE MAP PROJECT

INFORMATICA 2/89

Matjaž Colnarič, Ivan Rozman
Faculty of Engineering, Maribor

Keywords: MAP project, real-time extension, survey

This paper is intended to give a brief introduction to the MAP and a description of its major functions to the readers that are not familiar with it and the estimation of the state of the project to those readers who already know it. Layer functions are described and commented. Special emphasis is given to the real-time extensions and to the integrated architecture. At the end some comments on MAP 3.0 specification are enumerated.

1. INTRODUCTION

Information is becoming the most important factor in manufactoring industry. It is the fact that the information controls all the levels of production. Following the information flow in the factory, we pass all the crucial points, from the management, planning, finance to the process control management. Optimization of all these points helps us to economize the production which is the final goal of the effort. The technologies enabling this are computer science and microelectronics.

To make the information useful it has to be brought to the places where it is needed. In other words, the providers and the users of information have to be integrated in a system. Studying these problems a new discipline named CIM (Computer Integrated Manufacturing) arose and with right it became very fashionable. CIM became a must in the modern industry.

That is where the problem appears - missing a means for the communication. Computer companies have developed their own methods of transferring data between their products. As a result, different makes of computers are more and more difficult to integrate into the same system and devices need different interfaces to match different types of the data flow. Integration can thus easily mean changing and adopting interfaces to every system, which again means studying all the different systems in the network.

The only way to change this praxis seems to be the making of universaly accepted communication systems to fit the computers and to become a standard.

Persuading computer firms to redesign their products could never succeed. The only thing to do is to make the communication systems to fit the computers, in other words, to work towards an universally accepted communication system to which any make of programmable device could be easily standardized.

2. HISTORY OF THE MAP

The MAP resulted from the operating difficulties of a very large company - General Motors. By the end of 1970s, in the GM, they had 20.000 programmable controllers, 2000 robots and over 4000 intelligent devices in use and only good 10% of them were capable to communicate with each other beyond the limits of their own "island of automation". Making their plans for 1980s they realised that soon the sum of different inteligent stations to be interconnected would reach the number of 300.000.

To solve the problem, in 1980 a group of scientists led by Mike Kaminsky was organised and by 1983 they defined a communication protocol concept named MAP (Manufacturing Automation Protocol). In the beginning, some independed equipment manufacturers, later on a lot of mighty potential users, like Allen-Bradley, Gould, Motorola, IBM, Hewlett-Packard joined them, finding their future problems very much alike to them of GM. The first great success and public appreciation was achieved in 1984 at the National Computing Conference (NCC).

After that the project was supported by some new big companies (Ford, General Electric etc..), and the concept found interest outside USA, in Europe, Canada and Japan. In connection to the MAP, several other protocols were developed, like TOP (Technical Office Protocols) under the leadership of Boeing, real time extensions etc..

The next great event in the history of MAP was the AUTOFACT fair in 1985 in Detroit, where a 10mbit model network connecting several different stations was presented under the standard MAP protocol version 2.1.

The current version 3.0 was published in April 1987, including internetwork communication, TOP, real-time extension, carrier-band version etc.. The version 3.0 is to be left unchanged for the next six years. This should give users the oportunity to get used to it and to show its good and bad sides.

The final proof of the functionality of the MAP was ENE (Enterprise Networking Event), in June '88 in Baltimore, where a network of over 100 different active stations was successfully presented. We could see a little subset of this at the MAP-Sonderschau at SYSTEC'88 in Munich.

3. LAYERS OF THE ISO-OSI MODEL

In 1978, International Standard Organization (ISO) issued its famous seven-layer Reference Model for Open Systems Interconnection (OSI). It suggests a phylosophy of peer-to-peer communication between partners. It is not intended to be a standard, it only offers a way to solve communication problems. It also dues not have to be used in its entirety, the layers that are considered not to be necessary in a particular application can be omitted and their function can be distributed among other layers.

There are two good reasons why to use the layering: first, because of the complexity of the functions required in a comfortable transmission of data and second, it gives a possibility to make functions transparent and independent so it could be modified without disturbing others as long as interfaces are intact.

The idea of the model is to define functions of particular layers and interfaces between them. Portions of data (layer headers) are being added to the information to be transferred from one system in the network to another while travelling from the upper layer of the sender to the lowest - physical layer. The same headers are being unpacked and used while traveling the other way around in the receiver (see Fig.1).

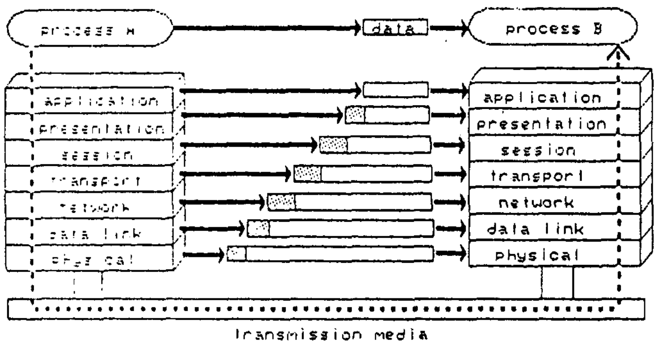


Fig.1. Data transmission between the processes in different nodes

Each node on the network is equipped with this layer mechanism, and each layer plays its part in data transfer, communicating with its alter ego in the partner node. The only direct link between nodes is the network cable, connecting all the nodes on the level 1.

The OSI architecture offers a possibility of connecting non-compatible nodes or networks, providing they support OSI reference model. It uses special nodes to adopt data up to the layer that is still common. Depending on this layer, the nodes are named repeaters (connecting distant equivalent networks or nodes), bridges, routers or gateways (universal coupling node). The principle is shown in the Fig.2.

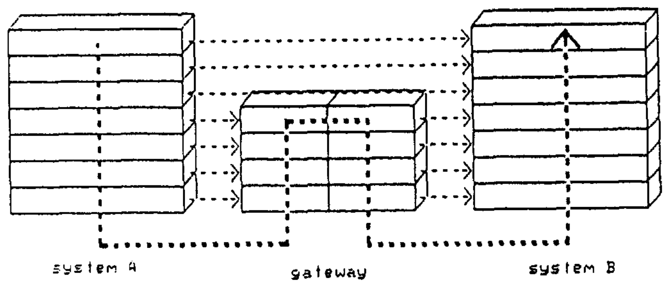


Fig.2. Connecting up to the layer i incompatible nodes

4. CHARACTERISTICS OF THE MAP

In the following chapter we shall describe major characteristics of the MAP protocol going through all the layers of the reference OSI model which is in its entirety implemented in the MAP.

<u>The Physical Layer</u>

provides the physical medium for transmitting data between two nodes. It provides the communication hardware, connects and disconnects, modulates and demodulates and drives data on the network.

MAP specifies two alternatives for the transmission medium, both conforming the IEEE 802.4, according to the application:

- for backbone networks and other multi-channel applications it recommends 10 Mbit/s broadband technology on co-axial cable,

- to suit less sophisticated applications it allows the use of carrierband 5 Mbit/s technology on co-axial cable.

The broadband means multi-channel, unidirectional, high-rate transmission. It allows multiplexing with other communication signals in the factory, data, voice and video transmission simultaneously on the same carrier. The carrierband means single-channel, bidirectional, low-cost, potentially more reliable transmission. It avoids the problems of broadband (head-end remodulators, precise tunning to the channels, no delay in remodulating between transmission and reception frequencies etc.), but it is slower in transmission rate and restricted to the cable length of 700m and a maximum of 30 nodes per segment.

The next version of MAP will probably add the third option - fibre optics in the near future. It has some very convenient properties: light-weight, suitable for long distance connections, very low attenuation, very high data rates, immune to electrical noise, safe in hazardous environments, unaffected by lightning, secure from unautorised tapping, electrically totally insulated connection - no grounding problems. But, in the field of fibre optics technology there is still not enough experience and no stable standards to lean on. The MAP mentions fibre optics in an addendum and will include it into the standard, as soon as conditions will be met.

The Data-Link Layer

The data-link layer means a communication link between the nodes, being responsible for assembling data to the frames, giving them adresses, gaining access to the network, checking the data and the medium for errors etc. IEEE802 divides the layer 2 into two independent sub-layers: the Logical Link Control (LLC) sub-layer and the Media Access Control (MAC) sub-layer. LLC does transfer, sequencing, error checking, and addressing and thus enabling the error free path between nodes. MAC manages the access to the physical medium.

MAC in MAP supports IEEE 802.4 token-bus configuration. It guarantees access to the network within a definite time (as opposite to e.g. the CSMA/CD).

LLC, using ISO 8802.2, offers three options to control the traffic: connectionless (type 1), connection oriented (type 2) or acknowledged connectionless (type 3). The connection oriented option organizes a virtual connection between communicating nodes. The connectionless one means sending data without any feedback information, and acknowledging one means no connection, but a possibility to confirm a succesfull receipt of a message. The connectionless transfer is very rapid, but without confirmation, error recovery and flow control. When using connection oriented option, one first has to establish a connection. After that any data can be sent with options of flow control, error recovery etc.. Finally, the connection has to be broken. The connectionless acknowledged type of LLC is used in real time extension of MAP (the MiniMAP), where connections can not be built up because of the time, but data have to be confirmed or an immediate answer is needed.

The Network Layer

Sometimes, for different reasons we have to connect two or more sub-networks into one system. The network layer has a task of routing messages between two nodes in the network, regardless whether they are in the same or an interconnected sub-network. This task includes finding the best communication route between the two nodes, and determining the physical address in the new environment.

The MAP uses CLNS (Connectionless Network Service). The internal organisation of the network layer is devided into three parts, the Sub-Network-Independent Convergence Protocol (SNICP), the Sub-Network-Dependent Convergence Protocol (SNDCP), and the Sub-Network Access Protocol (SNAcP). The SNICP encounters the required standard and service to the service above and to the user, the SNAcP is an interface to the Data-Link Layer, and the SNDCP converts the SNAcP services to and from the form (if) required for the SNICP.

The task of this layer is also naming. That is one of the crucial points in the network design. Variables and other names in the network that are local to a node should not be transparent to the others, and vice versa. This layer also defines a routing model and standard address formats which is one of the greatest contributions of the MAP.

The Transport Layer

Using the services given by lower layers, the transport layer enables to higher layers a reliable end-to-end data transfer. It controls data exchange, organizes access to the network, segments messages to a restricted length, and reconstructs coming segmented data and corrects errors occuring during transmission, if possible and necessary.

The MAP uses a very wide option of the suggested layer 4 specification to compensate rather limited choice (connectionless service) in network layer. It includes the basic functions - connection, segmentation, reassembling and disconnection - and adds some more sophysticated service - for example multiplexing of the connections (sharing one link to different connections in multi-network use), flow control, error detection etc..

The Session Layer

The session layer is the first of the three more sophisticated layers, which are application oriented. Its function is to set up and manage a dialogue between application processess. It ensures that exchange of data on Transport layer is structured and synchronised. It allows establishing and releasing session connections and transfers synchronisation signals.

In the MAP, the layer is broken into three parts; the Kernel Functional Unit, supporting the mandatory session services required to establish a session connection, transfering normal data and releasing the connection, the Duplex Functional Unit, supporting a two-way simultaneous transfer service, and the Resynchronise Functional Unit, allowing the session users to synchronise the transmission of data between asynchronuous nodes.

The Presentation Layer

The Presentation Layer is responsible for the presentation of the data among the nodes taking part in the session. There are two types of the notation in the OSI model. Abstract syntax is a notation for the formal description of data types and values associated with the representational formats used by computer systems. This form is then ready to be encoded into a bit level transmission format using transfer syntax. On the receiver side, the data must be translated from the transfer format to the abstract format.

The presentation layer makes transparent the existance of the transfer syntax, and provides for the conversion to and from it.

In previous versions of the MAP, the Presentation Layer was null. MAP 3.0 defines translation services and protocols and thus supports services of the application layer.

The Application Layer

The layers below the Application Layer were designed to be as much transparent to the network users as possible. The Application Layer prowides an interface between the user and the network. Because of the diversity of applications to support, the Application Layer can not be a universally valid standard. It is a set of procedures for each significant type of application. Information, the application layer is dealing with, concerns data, text, graphics etc.. Other layers deal with anonimous frames of data.

Standards provided by the Application layer are divided into two groups. The first, ACSE (Association Control Service Elements, former CASE), provides services for the establishment and termination of Application Associations. It is a common service for the communication, specifying service primitives like Associate request/indication/response/confirm; Release request/indication/response/confirm; Abort request/indication etc.

The second, called ASE (Application Service Elements) is there to provide the type of service required by the particular application. It includes four elements, Directory Services, File Transfer, Access and Management (FTAM), Network Management and Manufacturing Message Standard (MMS).

Directory Services:

To gain the access to data in remote nodes, all the nodes in network need information about all directories. This is enabled by the Directory Service. It has three major participants: the Directory User Agent (DUA), the Directory Service Agent (DSA) and the Directory Information Base (DIB). In the DIB the information about the directories is stored. The user gets this information from his Directory Service Agent that communicates with Directory Service Agent. Only the DSA has the access to data in DIB. The Directory Information Base can be stand-alone or distributed.

File Trasfer, Access and Management:

The FTAM includes services for information transfer between processes and file stores. It supports binary and text files. The main services available are creating and deleting files, transfering entire files, reading and modifying file attributes, erasing and searcing for the file contents etc.. It enables every node the use of all the files in the network as if they were its own.

Network Management

The network Management has three main roles: gathering information on the usage of the network media by the network devices, ensuring the correct operation of the network and providing reports. It manages configuration, name, performance and faults

Manufacturing Message Format

The Manufacturing Message Format standardises the format and semantics of a list of mnemonic instructions and messages covering all the operating information required by manufacturing devices. It is a simple, limited vocabulary designed to control the widest range of all kinds of the intelligent machines like NC devices, robot controllers, programmable controllers, PC systems etc. Messaging services for factory floor devices requiring device-specific information are defined in four companion specifications covering numerical control devices (EIA specification), programmable controllers (NEMA specification), robot controllers (RIA specification) and process-control specification (ISA specification).

4. MAP AND THE HARD REAL TIME

The MAP with its seven layer topology can not assure data transfer quick enough for time-critical applications like hard real time. On the other hand, this kind of communications does not need time-costly comfortable services.

For that reason there are two extensions in MAP to cope with time-critical data transfer, one is MAP-EPA, the other Mini-MAP.

The MAP-EPA (Enhanced Performance Architecture) is a dual-architecture node, incorporating a full seven-layers MAP node at the one side, and a reduced version, consisting only of the layers 1,2 and 7, on the other (see fig.3). The node can use a full architecture to communicate with other full-MAP nodes or a reduced side architecture for rapid access to the same nodes, provided they are on the same network segment.

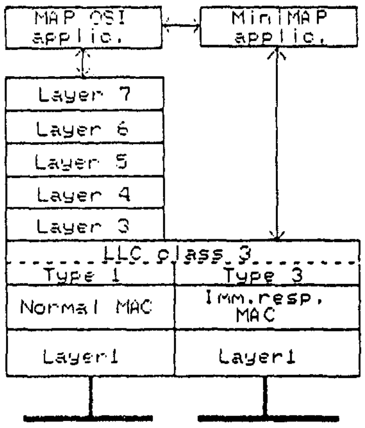


Fig.3: MAP-EPA architecture

The MiniMAP has the same architecture as the reduced side of MAP EPA architecture. It enables a rapid data transfer, but it is not an OSI- compatible device and hence can not communicate with full 7 layer nodes, only with nodes with the same architecture. That comprises its use in so called production cell networks. They have access to the backbone network over a MAP-EPA node, being in both networks and serving as a bridge.

The real-time communication needs to be very quick, but also very reliable. But, reliability takes time, so the two demands exclude each other. Reliability in communication is improved by acknowledging a successfull receipt of a message, but it takes time to gain control over the media. Reduced architecture resolves this problem introducing immediate responding. The transmitting node, possessing the token, passes it to the receiving one. If the message was received uncorrupted, receiver acknowledges it or provides the answer immediately if required. After that, the token is returned to the original transmitter.

5. INTEGRATED COMMUNICATION SYSTEM

In Fig.4 the communication system for Computer Integrated Manufacturing is shown. For the backbone, the full MAP factory-wide broadband network is proposed, connecting subnetworks and bigger systems and possibly serving for other communication needs in the factory (e.g. video and voice and other data transmission).

Subnetworks are tied to the backbone with some coupling devices, routers, bridges or gateways, depending on similarity of design. These are office, business and technical design dedicated subsystems, like TOP or other CSMA/CD

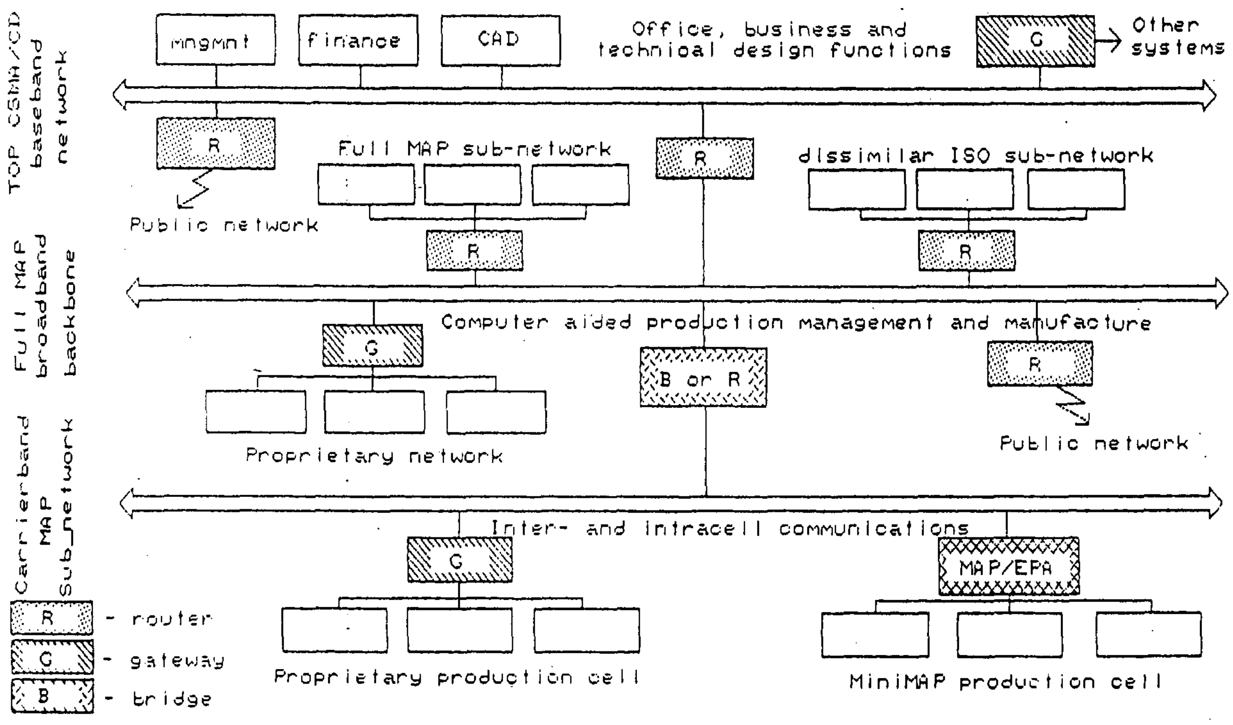

TOP CSMA/CD baseband network
mngmnt
finance
CAD
Office, business and technical design functions
G
Other systems
R
Public network
Full MAP sub-network
R
dissimilar ISO sub-network
R
R
Full MAP broadband backbone
Computer aided production management and manufacture
G
B or R
R
Proprietary network
Public network
Carrierband MAP Sub_network
Inter- and intracell communications
G
MAP/EPA
R - router
G - gateway
B - bridge
Proprietary production cell
MiniMAP production cell

Fig.4. MAP integrated communication system
(example from /6/)

networks on the one side, and production cells in real-time networks on the other. These networks can be proprietary, existing process control distributed systems or MiniMAP-like systems.

6. PRESENT STATE OF DEVELOPMENT

This chapter is meant to present the state of the MAP project as it was at the time of SYSTEC'88 at the end of October 1988 in Munich.

There were about 20 MAP vendors and users taking part in MAP Sonderschau. Information about Enterprise Networking Event in Baltimore in June '88 was also given where more than 100 different participants were connected to a MAP backbone network.

The MAP Sonderschau was initiated by the European MAP Users Group (EMUG). Its aim was to prove that MAP is operable and ready to integrate a lot of different products and even net-

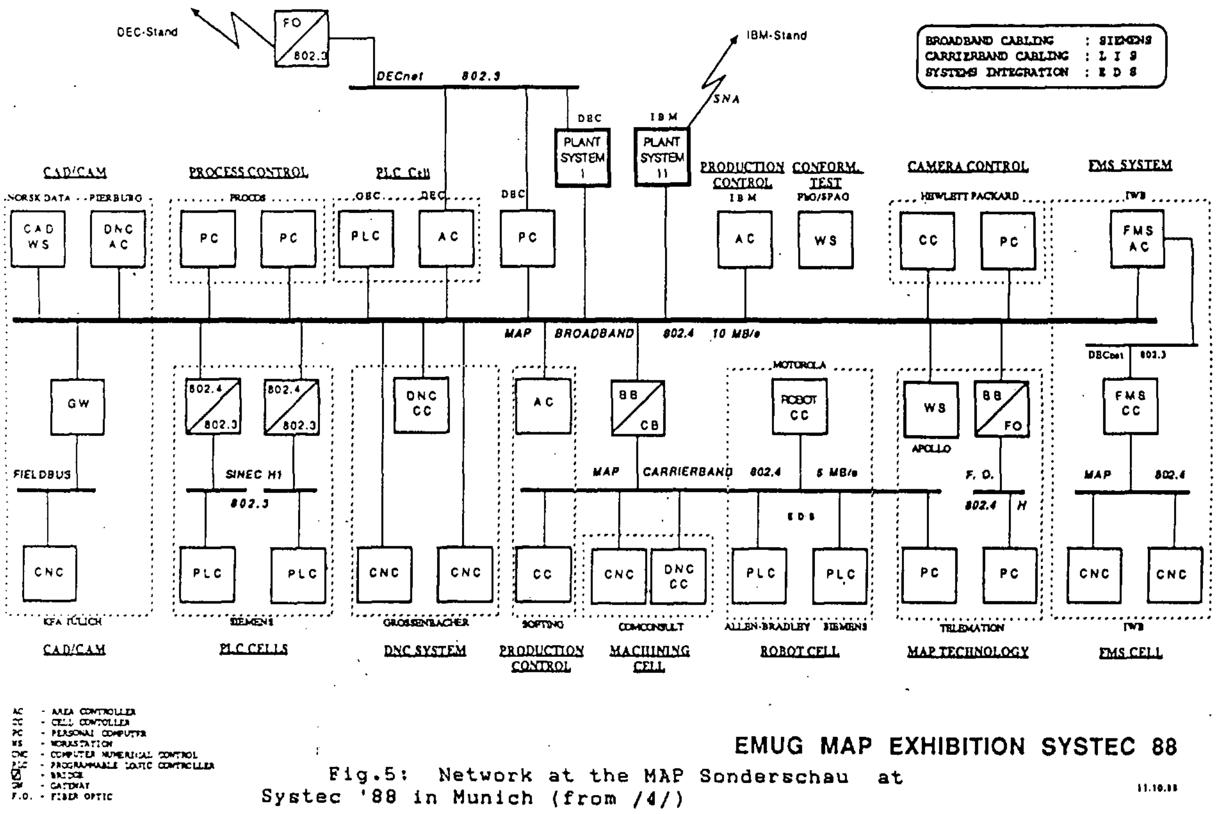


Fig.5: Network at the MAP Sonderschau at
Systec '88 in Munich (from /4/)

works into an interoperable system. On the MAP Broadband 802.4 10 MB/s backbone different nodes and sub-networks were connected, including TOP, MAP Carrierband 802.4, fiber optic carrierband of Telemation, DECnet 802.3, multi-vendor production cells and robots, CAD systems and even a separate video camera and a monitor system on its own channel. On the same broadband cable different versions of MAP, the 3.0 and the 2.1 are possible to show the upgrade possibility in the future. (see Fig.5)

Looking over the MAP products on ENE we must state that by Summer '88 only a few vendors could present them. The fairs in spring '89 will show if the promise that the number of MAP producers is increasing will be fulfilled. MAP board-level products were shown by Concord (PC-bus and Multibus products) and Motorola (VME-bus products). Some other announced new products. MMS software producers are also very rare: Motorola for its VME boards and SISCO for DOS versions. Support for other operating systems is announced. Knowing this, the fact that more than 100 nodes from different vendors at the ENE in Baltimore were interoperable is not so astonishing, as most of the vendors used the same Concord boards and Sisco software.

Another fact shown at ENE are the prices. They felt to the range of 2000 USD for one node, what is very near to the proposed price.

One of the less encouraging facts is that the biggest computer manufacturers, like DEC and IBM, still stand aside of the business as observers or producing their own OSI-compatible networks with an option to connect to MAP. It is certainly not that, what other dedicated MAP members would like to see. The development in early 1989 will show their intentions, which will be crucial for further progress of MAP. The latest news from DEC is not very encouraging.

7. CONCLUSION

Devotees are delighted: ENE was a triumph of MAP. It showed that numerous vendors were capable to communicate with each other. It is no doubt that MAP represents efforts of the largest group of computer manufacturers working together on one goal - to be able to transmit data over a network. The result of this collaboration is the widest all-embracing communication standard and everybody is waiting how it will prove.

Looking at the MAP critically, we can state some facts. First of all, some crucial topics are still not standardised, like network management. There exist several versions of some topics or they are not accepted by whole group. On the other hand, there are some parts of standards, that proved inconsistent or less suitable in the first applications, like immediate response option in real-time version.

To give the vendors and users enough time to implement their solutions they have frozen MAP 3.0 specification for the next 6 years. This period is very long in such a dinamic field like distributed computing, especially if the standard is not validated long enough. Deficiencies will have to wait too long to be included into a standard that would have to be a living structure.

The major disadvantage of MAP architecture are its real-time extensions. Introducing immediate response solves problems on the lowest level, it acknowledges the integrity of received data, not that of the message. There are troubles with error management since there is no higher-level mechanism. It allows no broadcasting and there is no clock synchronisation provided.

One of the greatest problems of the real-time MAP extensions is that it has no fault tolerant concept, what is crucial in modern process-control networks. Malicious node can not be located and excluded by the protocol.

In spite of all there are some very good and universally applicable standards in MAP that will survive regardless of long-term destiny of the whole project. One of the most important is doubtless Manufacturing Message Standard in the Application layer that defines semantics of communication language.

Some most important fairs in the near future will show the success of the proposed standard. The greatest computer manufacturers that are not convinced of the effectiveness of it are still standing by, leaving their door to the MAP open, but using mostly their own networks.

8. REFERENCES

/1/ SHERMAN, K. *Data Communications*, Reston, Reston Publishing Company, Prentice Hall, 1985

/2/ SUPPAN-BOROWKA, J. et SIMON, T. *MAP Datenkomunikation in der automatisierten Fertigung*, Pulheim, DATACOM Buchverlag, 1986

/3/ KOPETZ, H. *Distributed Real Time Systems and Fault Tolerance*, 1988, Courses in Advanced Technology, CEI-Europe Elsevier, Amsterdam

/4/ EMUG. *EMUG Sponsored MAP Exhibition Systec'88*, Munich, Publikationsgeselschaft Moderne Industrie, 1988

/5/ *MAP Report*, Eine Sonderpublikation zum Europaeischen MAP-Forum, Muenchen, Publikationsgeselschaft Moderne Industrie, 1988

/6/ *Through MAP/TOP to CIM*, Towards Integration Initiative, Leicester, Department of Trade and Industry, 1988

/7/ ROZMAN, I. et al., *Procesne komunikacije*, projekt Domače naprave in oprema, poročilo za PORS 21, Maribor, Univerza v Mariboru, Tehniška fakulteta, 1988

/8/ COLNARIČ, M. *Povezovanje računalnikov v tehniški informacijski sistem*, posvetovanje Informacijski sistemi, Maribor, Tehnička fakulteta, 1988

/9/ MAP Users Group, materials from the Duesseldorf, CeBIT Hannover, SYSTEC in SYSTEMS Muenchen 1987 in 1988 (EMUG, Motorola, DEC, Siemens, AEG, Hewlett-Packard, Telemation, EDS, etc.)

RELIABILITY PREDICTION OF PARSYS HYPERCUBE ARCHITECTURE

INFORMATICA 2/89

Keywords: parallel architecture, routing reliability, tours, hypercube

Rihard Piskar
Iskra Delta, Ljubljana

Abstract

The paper presents the combinatorial reliability model of a parallel computer PARSYS. The architectural design is based on 64 processor system with 64 memory modules, that are connected via network of routing nodes. The network is called 6-cube because 2^6 is the number of processors. At first there is derived the combinatorial reliability model of 6-cube architecture for 64 processors and memories connected in the hypercube. Then the reliability of the rerouting procedure is evaluated as the probability of the successful packet transfer from a processor to a memory. Further research showed that torus architecture is the enhancement of the packet transfer reliability because of redundant routing capabilities.

Introduction

PARSYS is MIMD parallel processor research project of Iskra Delta Computers Company. The architectural design is based upon 64 processor system with 64 memory modules, that are connected via network of routing units. Each memory module has the capacity from 2 Mbyte to 8 Mbyte and is divided to the local memory and global memory. Each routing unit supports fast routing and incorporates functions and logic that avoid or minimize degradations mostly encountered in the multiprocessor environments [1].

The routing network architecture design has passed several mayor development stages, the most important of them being hypercube and torus architecture. Theoretical studies of network topology had raised some doubts regarding high dimensional networks which require more and longer wires than medium dimensional networks. For that reason, binary N-cube was chosen with 2^N nodes in hypercube architecture with N=6. This is a special case of the family of m-arry N-cubes with N dimensions and m nodes in each dimension [2]. Latter on the transformation was done on torus architecture preserving hypercube functional characteristics and obtaining constant wire density. Torus is 8-arry 2-cube in comparision to hypercube 2-arry 6-cube with the same number of nodes, n=64 [3]. Separately to routing performance analysis the reliability analysis was done bringing some interesting cognitions .

The reliability analysis started first with hypercube architecture. As the evolution of the project proceeded, latter on torus reliability analysis showed even better reliability performances.

Modeling

Hypercube. The hypercube architecture of the parallel processor PARSYS is the essential actor of accounting the global memory to the processors. All communications between processors and the memory are passing at least one routing-node. If a routing-node fails, then the packet passing it, will not reach the destination.

To estimate the probability of that random event, there is necessary to have a probability model of the architecture, to make mathematical expression and to find out the relation between architecture and reliability. The probability that the packet transferes predefined number of routing-nodes, is combinatorial expressed and depends on the architecture. The architecture enables the rerouting in a case of failure in a routing-node. In that case the packet is sent via another path again. The model is expressing that possibility, but it was never realized because of great changements of the routing hardware.

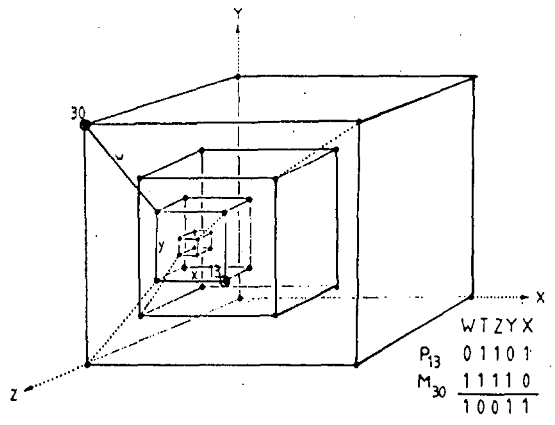


Fig.1.
The principle of hypercube architecture, the example of 6-cube.

From Fig.1. there are seen two addresses which differ in three bits, so that destination is three edges distant from the origin.

Let be in the system $n = 2^N$ processors and memories connected in N-cube, where N is the number of system levels. Let be the probability, that a processor want to get data from a memory, or to send changed data to the memory, uniformly distributed. Let this operation be called the packet transfer. Suppose, that connecting units fail with a failure rate π due to the Poisson law.

In the system N-cube each processor is connected to n memories of the appropriate level. The number of levels is N, so that the number of processors and memories is $n = 2^N$.

From each node there are N possible connections, in our case six. The addresses of possible connections are different in Hamming distance 1. Each node is containing the processor, the memory and the routing-node. We are interested in the probability, that the processor and the memory are distant in Hamming distance H. Suppose that the address consists of N bits and that k pairs of bits is different. Then H = N - k. In fact we are looking for the probability, that the addresses are distant for distance H or, that their *addresses have k different pairs of bits.*

This probability is:

$$p(k) = \frac{\binom{N}{k}}{\sum_{k=0}^{N} \binom{N}{k}} . \qquad (1)$$

The combinatorial probability of the successful packet transfer from the processor to the memory in other words the reliability of packet transfer is:

$$R(t) = \frac{1}{N} \sum_{k=0}^{N-1} [\ 1-p(k)q(k)] \qquad (2)$$

where p(k) is the probability of k different pairs of bits *in the processor and the memory* addresses, q(k) is the unreliability of k routing nodes:

$$q(k) = 1 - \mathrm{Exp}[-k\pi t] \quad \text{for hypercube without,} \qquad (3)$$

$$q(k) = [1 - \mathrm{Exp}(-k\pi t)]^2 \qquad (4)$$

for hypercube with rerouting, where π is the failure rate of single routing-node.

Equ.(3) is straight forward solution of the probability, that k routing nodes does not function correctly for time t. Equ.(4) envolves some redundancy of packet transfer routs. Let us suppose the possibility of the rerouting in a case of a fault in the packet *transfer* because *of a physical failure in* a unit, where the packet is transferred over. The packet is unable to reach the destination because of the failure in a routing-node. The rerouting is realized by the selection of another path.

Torus. The evolution of the parallel system proceeded to another architecture with better characteristics (less and shorter wires per board). This is 2-dimensional 8-arry torus with the same number of processors 64. The communication between processors and memories located at each node start from a node in four possible directions; north,south,west and east. In a case of a failure in one of nodes the alternative path is established. In the communication protocol two *directions* are provided, each time *the one with more* available routs to avoid the saturation.

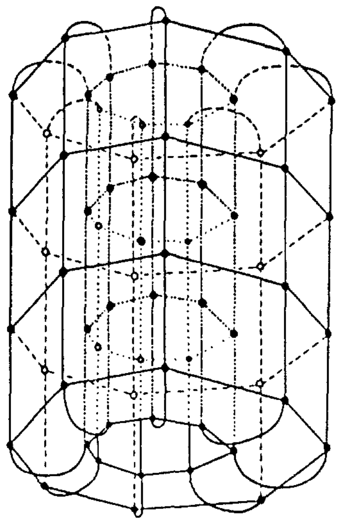


Fig.2.
Torus *archytecture of* 64 routing-nodes *connecting* 64 processors and memories.

There is a relation between N-dimensional m-arries of torus and hypercube. In 6-dimensional 2-arry hypercube there is 6 different possibilities of combining the number of nodes in one packet transaction and 64 in total. In 2-dimensional 8-arry torus there exist 8 different possibilities and 256 in total. The sum of probabilities of an established route with its reliability leads to greater reliability of torus than hypercube architecture. The reliability of randomly chosen single communication route for a packet transfer equals to Equ.(2) using (3) and of redundant route which is normal in torus architecture, equals to Equ.(2) using (4).

Reliability evaluation results

From the evolution point of view the analysis of different dimensions of hypercube and torus architecture is obvious. There must be the evidence of reliability versus dimension N and the number of processors $n=2^N$. Upon the combinatorial model the comparision of the reliability showed the following results that are summarized on the table 1.

Table 1.

The unreliabilities of packet transfer for hypercube and torus architecture for constant product of node failure rate and time $\pi t=0.1$

	TORUS	HYPERCUBE
Dimension	2-dim, 4-arry	4-dim, 2-arry
No. of processors	16	16
Max. no. of nodes	4	4
Single unreliability	0.031	0.031
Redundant unreliability	0.0010	0.0010
Dimension	2-dim, 8-arry	6-dim, 2-arry
No. of processors	64	64
Max. no. of nodes	8	6
Single unreliability	0.041	0.043
Redundant unreliability	0.0017	0.0018
Dimension	2-dim, 16-arry	8-dim, 2-arry
No. of processors	265	265
Max. no. of nodes	16	8
Single unreliability	0.034	0.041
Redundant unreliability	0.0011	0.0017

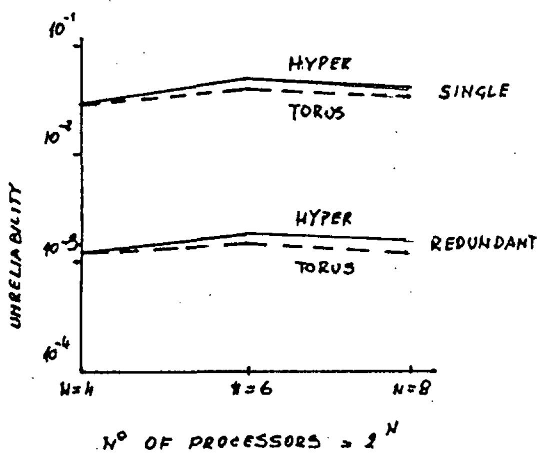


Fig.3.

Unreliability of hypercube and torus architecture for constant product of failure rate and time $\pi t=0.1$.

The differences between hypercube and torus architecture are not essential, if we look only single routing path. At N=6 there is not more than 5% difference in the unreliability of hypercube and torus. More significant is the difference between single hypercube and redundant torus. Hypercube is 36 times more unreliable than torus. It is true that the redundant hypercube is only 6% less reliable than torus, but to achieve the redundant routing algorithm in hypercube it is demanded to make huge changements in hardware. At the contrary, torus architecture contains the alternative algorithm to decide which direction may be taken. The rerouting is accomplished, when the route fails because of a failure in a routing node. Pragmatical meaning of better torus reliability in failures per thousant hours is that instead of $4\cdot10^{-6}$ failures there occure only 10^{-7} failures per randomly chosen packet transfer operation.

System availability

Another aspect of system performance is the availability as the function of failure and repair rate of routing-nodes. The failures of nodes are consequences of mutually independant random events. The natural result of this fact is, the architecture has no influence to the availability.

The technique used to evaluate system availability is the method of multidimensional Markov system, decribed in Ref.[4]. Briefly explained, the method consists of the allocation of system failure states in a multidimensional space in such a way, that the relation between the number of states S and the dimension d is $S=2^d$. Vice versa relation is $d=\log_2 S$.

From that relation with S=64 and d=6 the system named d-cube represents a failure state in each cube vertex. The numerical availability evaluation of the system was done by VMS program tool written in Fortran language. The inputs are failure and repair rates of nodes and dimension parameters. The outputs are system availability and MTBF. With 44.5 failures per milion hours and 2 repairs per hour the availability is 0.9989 and MTBF is 468 hours.

Final facts of the parallel system

Communication cost. The system PARSYS consists of 64 processor/memory boards folded into torus configuration. Communication cost is the function of the ratio between processor speed and the speed of routing-node as well as the function of dimension. It was found [3], that torus communication cost is quickly encreasing over dimension N=6.

Hardware design. The processor board is VME/386 processor with communication node and dinamic 2M to 8M Byt RAM. The packet transfer logic is the most important part of the node. Each packet routine logic consists of mechanism for independent selection of a packet in the direction of more paths to desired node.

Software design. Software task partitioning is based on explicite parallel constructs for two level paralellism; macro or user specified and micro with iterative structures embeded into macro paralellism. A very fast micro task has been developed based on the fact that a processor can interrupt any processor. The synchronization is a simple get-and-link swap mechanism [3].

Conclusion

The reliability evaluation showed some advantages of hypercube architecture. One of them is relatively low average number of routing-nodes in randomly chosen packet transfer operation. There is also a low level of the unreliability of packet transfer. The reliability of the hypercube architecture can be even encreased with possible rerouting, but with essential changement of deterministic routing algorithm. Better solution due to general better characteristics of torus architecture is to implement redundant capabilities of torus architecture. The combinatorial model can be used in further reliability analysis of parallel systems to find out the optimum number of processors to trade-off the communication cost versus the reliability.

References

[1]S. Prešern, "A Selected Survey of Parallel Computer Systems", Journal of Computing and Informatics, Vol.11, No.4,pp.40-53, YU ISSN 0350-5596, 1987.
[2]P. Brajak, "Designing Reconfigurable Intelligent Memory Module Performance Enchancement to Large Scale General Purpose Parallel Processor",Journal of Computing and Informatics, Vol.11, No.1, pp.19-53, YU ISSN 0350-5596, 1987., 1987.
[3]P. Brajak, S. Prešern, L. Vogel, A.P. Železnikar, P. Hitij, "Sheduling and Synchronization Concepts on the PARSYS Parallel Processor", Proc. on ISMM International Conference, Florida, 1988.
[4]R. Piskar, J. Virant, "Hardware-Software Availability of a System", Proc. of 6th int. conf. on reliab. and maint. Strasbourg 1988.

INFORMATICA 2/89

KOMUNIKACIJA ČLOVEK – RAČUNALNIK V REGULACIJSKI TEHNIKI

Keywords: man-machine interaction, control engineering, system design

Matjaž Debevc, Rajko Svečko, Dali Đonlagić
Tehniška fakulteta Maribor

Zaradi vedno večjega razvoja strojne opreme, programske opreme, se vedno bolj uveljavlja področje komunikacije človek - računalnik. Razvoj sistema človek - stroj se je razvil iz treh področij, ergonomičnega načrtovanja, sistemskega načrtovanja in programskega načrtovanja. Regulacijska tehnika ter komunikacija človek - računalnik so del tega sistema. Za učinkovito in uspešno komunikacijo moramo poznati primarne in sekundarne kriterije za kvaliteto ter pogoje za dobro komunikacijo. Pri regulacijskih sistemih moramo poznati model človeka ter model računalniškega procesa. Pri načrtovanju programov se moramo ozirati na funkcije procesa in funkcije uporabnika. Posebej je pomembno vprašanje, kaj je potrebno poznati pri izvedbi vnosa. Delo pri tem podaja določene rešitve in nekatera ergonomična navodila.

Computer - human interaction has become important due to the incessant development of hardware and software. The system man-machine has developed from the following three fields: ergonomic design, system design and software design. Control engineering and computer-human interaction are a part of this system. In order to enable an efficient and a succesful communication, primary and secondary quality criteria should be known, as well as the conditions that assure a reliable communication. In control systems the model of man and the model of the computer process should be known. Process functions and user functions should be taken into account in software design. Special attention should be paid to the requirements of data entry. Certain solutions are given together with some ergonomic instructions.

1. UVOD

Na področju računalništva in informatike je viden nezadržen razvoj strojne opreme. Zaradi vedno večjih hitrosti in zmogljivosti pridemo do situacij, ko lahko omogočimo čim bolj naravno komunikacijo človek - računalnik. Ker pa je razvoj strojne opreme hitrejši kakor razvoj programske opreme se dogaja, da so programi narejeni površno in nepopolno. To je posledica ciljev proizvajalcev, da naredijo program, še preden jih prehiti konkurenca ali zaradi slabe povezanosti tima.

Največ časa in veliko vloženega napora je potrebno za načrtovanje in programiranje tistih delov programa, ki se ukvarjajo s komunikacijo. Ti običajno zavzamejo tudi največ pomnilniškega prostora in programske kode (od 60% do 80% celotnega programa).

Pri snovanju učinkovitega uporabniškega vmesnika se poraja vprašanje, kakšna je optimalna komunikacija človek - računalnik. Poznati moramo razvoj sistema človek - stroj in opisati model človeka v povezavi s tehničnimi sistemi. V tem primeru z regulacijskimi sistemi. Ta model je osnova za razvoj učinkovitega uporabniškega vmesnika. Regulacijske sisteme delimo na vodenje procesa in na regulacije (za odpravo motenj). Glede na ta dva področja ustrezno načrtujemo strukturo za izdelavo komunikacije. Pri snovanju programa se držimo ergonomičnih navodil za pisanje programa.

2. SISTEM ČLOVEK - STROJ

Sistem človek - stroj se je razvil iz treh področij: ergonomičnega načrtovanja, sistemskega načrtovanja in programskega načrtovanja.

ergonomično načrtovanje • analiza nalog • obvestila • elementi uporabe • govorni vhodi/izhodi	sistemsko načrtovanje • postopki načrtovanja • simulacije in modeliran. • regulacijska tehnika • organizacija in komunik.
SISTEM ČLOVEK - STROJ	
programsko načrtovanje • obdelava podatkov • ekspertni sistemi • grafični prikaz • pripomočki dela	

Slika 1: Povezanost s sistemom človek-stroj

Skupne točke, stičišče posameznega področja so v sistemu človek - stroj. Ta sistem vsebuje eden ali več delov iz vsakega posameznega področja in jih združuje v skladno celoto. Del sistemskega načrtovanja je tudi regulacijska tehnika, ki nam je osnova v tem članku.

3. KRITERIJI USPEŠNE KOMUNIKACIJE

KRITERIJI ZA KVALITETO

Kadar ocenjujemo kvaliteto se ravnamo po kriterijih zaradi lažje primerljivosti in pravilne izbire tehnike za razvoj komunikacije. Kriterije delimo na primarne in sekundarne.

PRIMARNI:

- čas, ki ga uporabnik potrebuje, da opravi neko delo,
- natančnost, s katero uporabnik opravi delo,
- zadovoljstvo pri delu.

V zelo kreativnih okoljih je potrebno čim bolj optimizirati zadnji kriterij, medtem ko je v normalnih razmerah potrebno optimizirati čas, hitrost dela. Ti kriteriji so predvsem odvisni od sestave projekta, od znanja in sposobnosti uporabnika in od fizičnih karakteristik naprave.

SEKUNDARNI

- čas spoznavanja,
- čas učenja,
- čas obnavljanja naučenega,
- pomnenje,
- občutljivost na napake in utrujenost,
- omejitve motoričnega gibanja.

Vsi časi naj bodo čim krajši, kajti obsežne knjige ali navodila ne vodijo k učinkovitosti in produktivnosti.

O pomnenju govorimo, kadar si mora uporabnik zapomniti za kratek ali daljši čas simbole in tehnike, ki jih uporablja pri svojem delu.

Utrujenost nastane običajno zaradi slabega načrtovanja potez, ki jih mora uporabnik napraviti pri svojem delu (predolga uporaba svetlobnega peresa). Utrujenost vpiva na število napak in s tem tudi na zadovoljstvo pri delu.

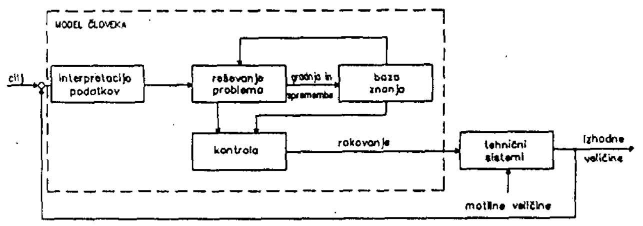
Omejitve se izražajo v velikosti delovnega prostora (doseg uporabnikove roke). Poleg delovnega prostora so tudi omejitve v vidnem polju (opazovanje več oken hkrati, opazovanje manjših slik v večjih ...).

POGOJI ZA DOBRO KOMUNIKACIJO

- ENOSTAVNOST - glavni del sistema ni obširen ali pa je organiziran hierarhično
- JASNOST - vsak del sistema je jasno viden in razumljiv
- SPOZNAVNOST - deli sistema spomnijo uporabnika na stvari, ki jih pozna
- POPOLNOST - sistem je urejena vsota njegovih delov
- UREJENOST - kar uporabnik pozna v enem delu, mu pomaga tudi v ostalih delih
- ZANESLJIVOST - sistem odgovarja uporabniku v zanesljivem dialogu
- DOVZETNOST - interaktivni odgovori so hitri, vljudni in informirajoči.

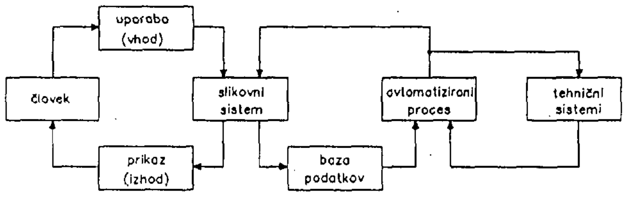
4. KOMUNIKACIJA V REGULACIJSKIH SISTEMIH

Če želimo narediti uspešno komunikacijo človek - računalnik v povezavi z regulacijskimi sistemi, se v prvi vrsti srečamo z modelom človeka, na katerega se oziramo pri načrtovanju in pisanju programa.



Slika 2: Model človeka v povezavi s tehničnimi sistemi

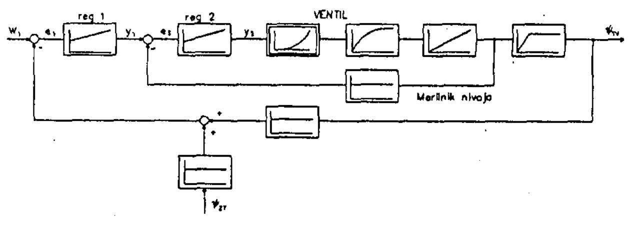
Naloga operaterja pri opazovanju in vodenju tehničnega procesa je sestavljena iz opazovanja trenutne situacije, primerjanje z zastavljenimi cilji, iskanje rešitev problema (na osnovi osvojenega znanja), priprava podatkov in poseganje v proces (menjanje stanje sistema). Za olajšanje dela in za dosego večje natančnosti in učinkovitosti se uporabljajo računalniško podprti vizualni sistemi. V glavnem so to zmogljive grafične naprave z velikimi hitrosti prikazovanja, izvajanja operacij in z veliko kapaciteto pomnilnika. Operater sedaj samo vstavi podatke in opazuje proces in ukrepa samo v določenih situacijah. Namesto z ročnim poseganjem v proces lahko to sedaj opravimo z računalniškim algoritmom.



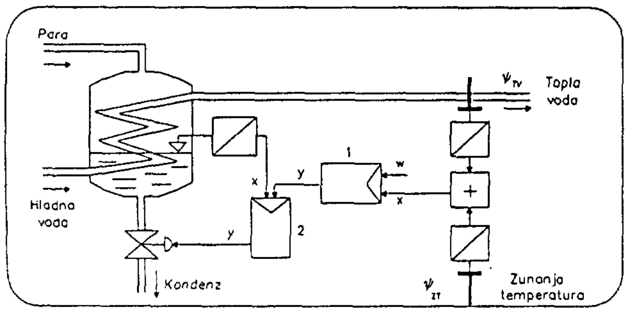
Slika 3: Model računalniškega procesa

RAZDELITEV FUNKCIJ ZA PROCES

Regulacijsko tehniko razdelimo na področje vodenja procesa in regulacije (za odpravo motenj). Najprej se ustvari slika procesne sheme iz katere nato dobimo model procesne zanke, v katero vpišemo na določena mesta ustrezne procesne parametre. Te slike kreiramo s pomočjo simbolov, črt in z risanjem teksta.

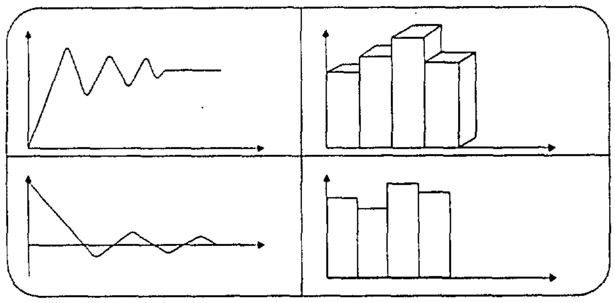


Slika 4: Model procesa



Slika 5: Procesna shema

Definirajmo si področja, kjer bo prikaz podatkov (digitalnih vrednosti, tabele, diagrami, krivulje in sporočila).

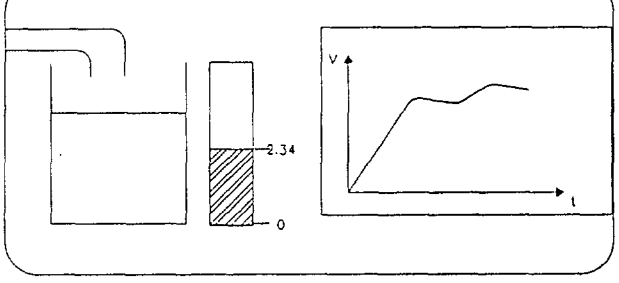

Slika 6: Prikaz

Nato imamo možnost analize ali sinteze regulacijske zanke.

Pri analizi je sistem podan in mu določimo stabilnost, vodljivost in spoznavnost. Vsi rezultati se prikažejo v ustrezni obliki na zaslonu.

Sinteza pa pomeni načrtovanje regulatorja v regulacijski zanke (izbira parametrov , izbira strukture). S pomočjo ukaznih menujev direktno v shemi regulacijske proge določamo parametre ali bloke in simuliramo progo. odzivi se zopet prikažejo v utrezni obliki na zaslonu. Načrtujemo lahko klasične kot tudi samonastavljive regulatorje.

Po načrtovanju ponovno pokličemo procesno shemo in dinamično opazujemo potek simulacije vodenja procesa. Po simulaciji lahko priključimo računalniški sistem na ustrezen dejanski proces. Pri vodenju procesa se nam na ekranu dinamično prikazujejo trenutni odzivi in parametri procesne sheme.



Slika 7: Dinamično opazovanje elementov v procesu

RAZDELITEV GLAVNIH FUNKCIJ UPORABNIKA

Za celotno simulacijo in vodenje procesa je značilno, da vedno nastopajo naslednje funkcije uporabnika:

- razpoznavanje situacije
- primerjava situacije z določenimi cilji
- iskanje rešitve problema
- priprava vhodnih podatkov
- izvedba vnosa

Prve štiri funkcije smo na kratko že opisali, ena pomembnih točk pri komunikaciji človeka z računalnikom pa je izvedba vnosa.

VNOS

Pri načrtovanju vnosa se moramo ozirati na tri pomembna vprašanja:

- katere interaktivne naprave bomo uporabili?
- kako kreirati dialog?
- kakšne naj bodo sistemske funkcije?

INTERAKTIVNE NAPRAVE

Izhodne	Vhodne
• alfanumerični zaslon	• kontrolne tipke
• grafični zaslon	• miška
• tiskalnik	• grafična plošča
• svetlobna plošča	• svetlobno pero
• risalnik	• plošča na dotik
• zvočnik	• igralna palica
• faksimile	• sledilna žoga
	• programabilne tipke
	• razpoznavalnik glasu

DIALOG

Pri pisanju programov za dialog se omejimo na ukaze in vnos podatkov. Izbira ukazov naj bo sestavljena iz ukaznih menujev, imenovanih "okna" (windows). Za manipulacijo in izbiro ukazov pa najbolj pogost uporabimo miško. Če pa miške nimamo na voljo pa uporabimo funkcijske tipke za premik kurzorja.

Običajno za selekcijo ukazov izberemo naslednje naprave:

- direktne, kot so svetlobno pero in plošča na dotik,
- indirektne, kot so miška ali grafična plošča, Te naprave so najbolj priljubljene in razširjene,
- simulacijske, kot so alfanumerična tipkovnica ali fizični lokator.

KREIRANJE VIDNEGA PROSTORA ZA UKAZE IN VNOS PODATKOV

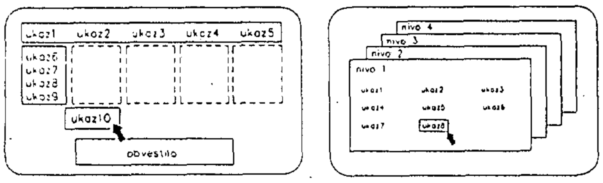
Vidni prostor v zadnjem času kreiramo prav z okni. Pri tem moramo vedeti naslednje:

- okna naj ne bodo prevelika ali premajhna,
- informacije v njih morajo biti kratke, jedrnate in jasne,
- pravilno izbirajmo barvo za ozadje in barvo črk,
- okna naj se pojavljajo hierarhično. Če je informacij, ukazov malo, naj bodo okna enonivojska,
- obvestila naj bodo v spodnjem delu ekrana, vendar

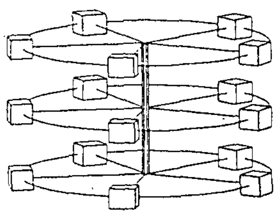
dovolj dobro vidna in opremljena z zvočnim signalom,

- izogibajmo se utripajočim ukazom, ki so škodljiva za oči, bolje je, da jih poudarimo.

Ukaze lahko izbiramo s padajočimi menuji ali pa v 3D tehniki.

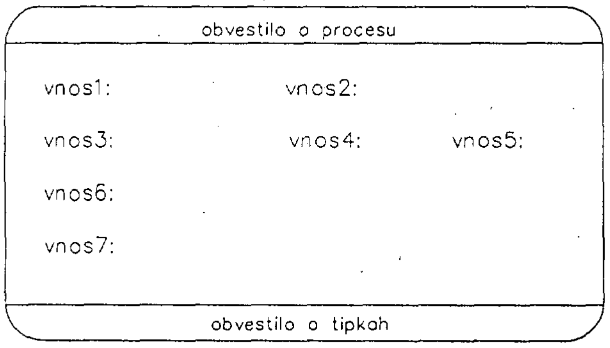


Slika 8: Padajoči menu in 3D tehnika

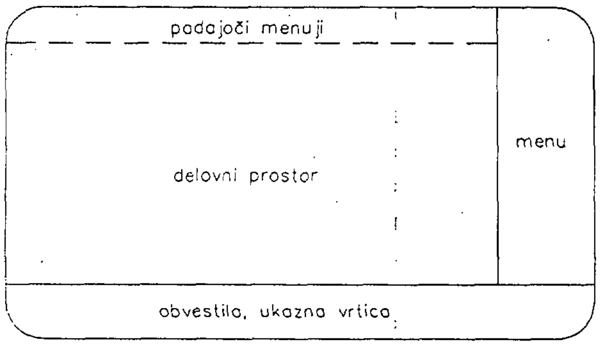


Slika 9: Valjna tehnika

Kadar imamo vnos večjega števila podatkov, naj bodo vnosi enakomerno razporejeni po ekranu.



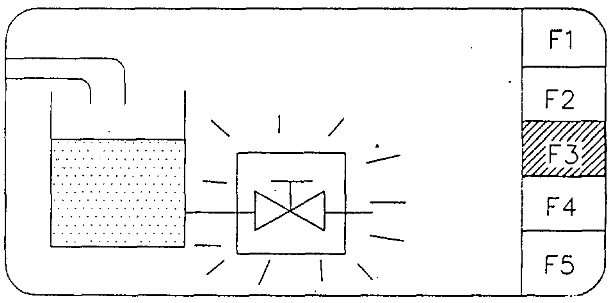
Slika 10: Alfanumerični vnos podatkov



Slika 11: Grafični vnos podatkov

SISTEMSKE REAKCIJE

Sistemske reakcije naj bodo v procesu takoj vidne (utripanje objekta, funkcijske tipke na zaslonu, zvočni signal). Zaželjeno je, da prikažemo direktno ali indirektno tudi odzive na vhode, ki smo jih dobili pri simulaciji zaradi primerjave.



Slika 12: Prikaz sistemskih reakcij

5. ZAKLJUČEK

Vedno hitrejši, zmogljivejši, prostorsko skorajda neomejeni računalniki danes omogočajo vedno boljšo in lažjo komunikacijo človeka z računalnikom. Cilj programerja ne sme biti, da naredi program po meri strojne opreme, ampak po meri človeka. Človekovo delo z računalnikom mora biti čim bolj naravno in blizu okolja, v katerem dela. Za načrtovanje takšnih programov pa je potrebno veliko napora in dodatnega dela, ki se pozneje zelo obrestuje pri uspešnosti programa. Uporabniško prijazni programi povečajo produktivnost, natančnost ter veselje človeka do dela z računalnikom.

Pri načrtovanju programov se moramo držati ergonomičnih navodil za delo računalnika s človekom. Ne nazadnje, če imamo možnost, izberimo ustrezne barve za dodatno informacijo.

Vedno moramo imeti pred očmi uporabnika programa in njegovo delo, saj bomo le tako naredili uspešen program.

6. LITERATURA

1. James Martin, Design of Man-Computer Dialogues, Prentice-Hall, New York, 1973,
2. Več avtorjev, Enajsta šola računalništva, Knjižnica Sigma, Ljubljana, 1988,
3. N. Guid, Računalniška grafika, TF Maribor, 1988,
4. R. Grimm, Zu ergonomischen Gestaltung der Mensch - Prozess-Schnittstelle: Prozessbeobachtung und Prozessbedienung, Regelungstechnische Praxis, 1984, št.4, str. 153-160,
5. G. Johannsen, Kassel, Neue Entwicklungen bei Mensch - Mashine - Systemen, Automatisierungstechnik, 1987, št.10, str. 385-395,
6. J. Foley, V. Wallace, P. Chan, The Human Factors of Computer Graphics and applications, november 1984, str. 13 - 48.

Keywords: neutral networks, brain, artificial intelligence, survey

Igor Kononenko
Fakulteta za elektrotehniko
in računalništvo, Ljubljana

V zadnjem času se vse več raziskovalcev ukvarja z načrtovanjem in uporabo nevronskih mrež. Nevronska mreža je sestavljena iz velikega števila preprostih elementov 'nevronov', ki delujejo paralelno in asinhrono in komunicirajo preko vezi, ki jim pravimo sinapse. Vsaka sinapsa ima pridruženo realno vrednost (utež), kar predstavlja porazdeljen pomnilnik nevronske mreže. Vsak nevron generira svoj izhod iz obteženega vhoda, ki je lokalno dosegljiv. Lepe lastnosti nevronskih mrež so (za vsako lastnost lahko najdemo izjemo in kolikor je avtorju znano ni nobene nevronske mreže, ki bi imela vse te lastnosti): podobnost biološkim sistemom, visok paralelizem in asinhrono izvajanje, večsmerno izvajanje, prilagodljivost v realnem času, robustnost glede na okvare in glede na manjkajoče podatke, sposobnost učenja in avtomatska generalizacija, mreža ne potrebuje programske opreme in eksplicitne konfiguracije, ni razlike med podatki in adreso, področje ima dobro matematično podlago. Glavna pomanjkljivost nevronskih mrež je nezmožnost razložiti svojo odločitev.

NEURAL NETWORKS

In resent years a lot of researches pay attention to the design and application of neural networks. A neural network is constructed from a large number of simple elements, 'neurons', which operate independently and communicate to each other via connections called 'synapses'. A certain weight is associated to each synapse representing the network's distributed memory. Each neuron combines its weighted inputs into its output. All information that a neuron needs for operation is locally available. A good introduction into the field of neural networks are (Rumelhart & McClelland 86), (McClelland et al. 86) and (Kohonen 84). The following are attractive features of neural networks (for each feature there can be found exceptions and as far as the author knows there is no neural network with all these features): biological similarity, high scale parallelism with asynchronous update, real time adaptiveness, multidirectionality, roboustness with respect to damage of a part of a network and to missing data, learning capabilities and automatic generalization, network needs no software, no algorithms, no explicit configuration, there is no distinction between an address and data, good theoretical background. The main weakness of neural networks is the lack of the ability to explain their decisions.

Intelligence emerges from the interaction of large numbers of simple processing units.

(Rumelhart & McClelland 1986, Predgovor)

1. UVOD

Oglejmo si zgled matričnega računa, ki ga je zmožna opraviti nevronska mreža na sliki 1.

Vsak nevron ima dve različni stanji. Za naš primer je stanje vsakega nevrona lahko 1 ali -1. Naloga nevronske mreže je naučiti se razpoznavati naslednja dva vzorca:

$$X1 = (1,1,1)^T \quad \text{in} \quad X2 = (1,-1,-1)^T$$

Oglejmo si najprej, kako to zmore matrični račun. Sestavimo matriko kot vsoto matrik, ki jih dobimo kot produkt vektorja s transponiranim vektorjem samim:

$$M = X1*X1^T + X2*X2^T = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix}$$

Definirajmo še odločitveno funkcijo (pragovni element):

$$f(X) = \begin{cases} 1, & X > 0 \\ 0, & X = 0 \\ -1, & X < 0 \end{cases}$$

Če posplošimo odločitveno funkcijo tudi na vektorje, lahko zapišemo naslednje:

$$f(M*X1) = f((2,4,4)^T) = (1,1,1)^T = X1$$

$$f(M*X2) = f((2,-4,-4)^T) = (1,-1,-1)^T = X2$$

Vidimo, da produkt matrike z danim vektorjem obdrži približno smer danega vektorja, ki jo z odločitveno funkcijo še popravimo in dobimo isti vektor.

Poskusimo uporabiti isti postopek, ko prvotne vektorje poznamo le delno. Z 0 označimo neznano vrednost komponente:

$$X1' = (1,1,0)^T \quad \text{in} \quad X2' = (1,0,-1)^T$$

$$f(M*X1') = f((2,2,2)^T) = (1,1,1)^T = X1$$

$$f(M*X2') = f((2,-2,-2)^T) = (1,-1,-1)^T = X2$$

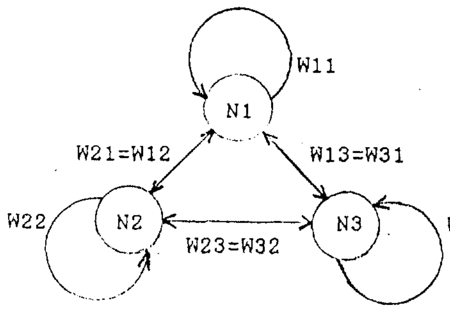
Produkt matrike z delno poznanim vektorjem nadomesti manjkajočo vrednost. Poskusimo sedaj še z napačnimi vrednostmi:

$$X1'' = (1,1,-1)^T = X2''$$

$$f(M*X1'') = f((2,0,0)^T) = (1,0,0)^T$$

V tem primeru je odgovor neodločen. Dejansko je v našem primeru nemogoče ugotoviti, ali je napačna druga ali tretja komponenta.

Dosedanje razmišljanje ne velja vedno oziroma velja pod določenimi pogoji (npr. zahteva se ortogonalnost vektorjev), vendar to presega okvir tega članka. Zanimivo je, da ima vsak shranjeni vzorec shranjen tudi komplement (npr. če shranimo vektor (1,-1,-1), se avtomatsko shrani tudi vektor (-1,1,1)).



Slika 1: Primer nevronske mreže

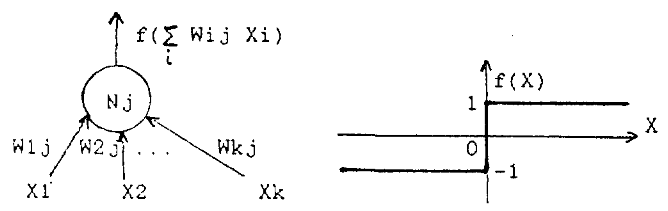
Oglejmo si, kako lahko gornji račun realiziramo z nevronsko mrežo. Na sliki 1 je prikazana nevronska mreža, sestavljena iz treh nevronov, tako da je vsak povezan z vsakim. Vezi med nevroni ustrezajo elementom matrike. Vezi so dvosmerne, zato je matrika simetrična. Diagonalni elementi matrike ustrezajo vezem nevronov samih s seboj. Vsaka vez ima pridruženo realno vrednost (pozitivno ali negativno), ki ustreza povezavi med nevronoma, ki ju vez povezuje. Tem vrednostim pravimo tudi uteži.

Pred učenjem so vse uteži enake 0. Učenje poteka tako, da se vsakemu nevronu vsili vrednost (stanje) ustrezne komponente iz vhodnega vektorja. Če imata nevrona, ki ju vez povezuje, enake vrednosti, se utež vezi poveča za 1, sicer se zmanjša za 1 (to je posplošeno Hebbovo pravilo učenja, opisano v razdelku 4.3.1). Prej opisan postopek ponovimo za vsak vhodni vzorec. Ko je mreža naučena, se uteži več ne spreminjajo.

Preverjanje novega primera poteka tako, da se vrednosti ustreznih komponent (delnega) vzorca vsilijo nevronom in vsak nevron paralelno izračuna svoje stanje, ki je hkrati tudi njegov novi izhod, po pravilu:

$$Yj = f(\sum_i Wij\ Xi)$$

kjer je Xj stanje j-tega nevrona, Wij utež sinapse med i-tim in j-tim nevronom in Yi novo stanje i-tega nevrona. Vsak nevron je procesni element, ki zna izračunati obteženo vsoto vhodov in z odločitveno funkcijo preslikati dobljeno vsoto v eno od dveh izhodnih vrednosti (slika 2). Obteženi vsoti pravimo funkcija aktivacije in odločitveni funkciji f pravimo izhodna funkcija. V razdelku 4.4 bomo videli, da lahko izhodna funkcija vrne tudi več kot 2 različni izhodni vrednosti.



Slika 2: Funkcija, ki jo izračunava posamezni nevron

V našem primeru so vsi nevroni paralelno in <u>sinhrono</u> naenkrat izračunali svoj izhod. Ponovitev iste operacije nebi več spremenila nobenega stanja nevronov. Pravimo, da je izvajanje nevronske mreže prišlo v <u>fiksno točko</u> (ekvilibrium). V splošnem nevronska mreža iteracije ponavlja toliko časa, dokler se stanja nevronov ne ustalijo. Če se to zgodi v končnem času, pravimo da mreža konvergira. Izvajanje nevronske mreže je lahko tudi <u>asinhrono</u>, tako da samo en nevron spremeni svoje stanje naenkrat.

Če je utež na sinapsi pozitivna, pravimo, da je sinapsa <u>vzbujevalna</u>. Če je utež negativna, je sinapsa <u>zaviralna</u>. Manjkajoči sinapsi ustreza utež 0. Ponavadi so uteži na sinapsah predstavljene z matriko.

Dokazano je, da če je matrika simetrična, potem asinhroni model vedno konvergira (Kosko 1988), sinhroni pa ne vedno (McEliece in sod. 1987). Za asinhroni model se zahteva, da samo en nevron naenkrat spremeni svoje stanje. Obstajajo tudi modeli, ki uporabljajo asimetrične matrike, za katere je empirično pokazana konvergenca (Wong 1988).

Če uporabljamo nevronsko mrežo kot asociativni pomnilnik, potem ponavadi mreža konvergira k fiksni točki, ki ustreza najbližjemu (najbolj podobnemu) shranjenemu vzorcu danega vhodnega vzorca. Fiksna točka pa ni nujno eden od shranjenih vektorjev in ni nujno shranjeni vektor, ki je najbolj podoben vhodnemu vektorju (McEliece in sod. 1987). Fiksne točke so linearno neodvisni vektorji, katerih podmnožica so tudi lastni vektorji matrike.

2. PORAZDELJEN POMNILNIK

Osnovni princip nevronske mreže je porazdeljen (distribuiran) pomnilnik. Pomnilniške celice so v povezavah med nevroni. Vsak nevron ima dostop do pomnilnika na povezavah z drugimi nevroni (ni nujno, da je vsak nevron povezan z vsakim). To lahko interpretiramo kot veliko število omejitev, ki se hkrati upoštevajo pri generiranju obnašanja. Porazdeljen pomnilnik ne vrne vedno eksaktnega podatka, ker je podatek shranjen v množici povezav z drugimi podatki.

Lastnosti porazdeljene reprezentacije so:
- avtomatska generalizacija: vsaka pomnilniška celica predstavlja povezanost med aktivnostima dveh nevronov (povprečno glede na vse situacije, v katerih se je mreža do sedaj nahajala oziroma glede na vse do sedaj prikazane učne primere). To znanje se lahko uporablja za aproksimacijo manjkajočih vrednosti ali za popravljanje napačnih vrednosti. Tako znanje je t.i. <u>površinsko znanje</u>. Mreža ne pozna <u>globokega znanja</u>, ki vsebuje znane relacije in zakone, ki veljajo v dani problemski domeni.
- prilagodljivost na spreminjajoče se okolje: s spreminjanjem okolja se mreža inkrementalno uči in s tem prilagaja na okolje; pozabljanje že naučenega lahko kontroliramo z določenimi parametri
- čim večja je distribuiranost, tem bolj natančno lahko shranimo iste podatke z enakim številom nevronov (Hinton in sod. 1986).

Distribuiranost povečamo z drugačnim kodiranjem, npr. namesto da en nevron predstavlja eno komponento lahko več nevronov zakodira vrednosti več komponent vhodnih vektorjev. Na ta način je ena komponenta shranjena v povezavah z več ostalimi komponentami, kar omogoča bolj natančno rekonstrukcijo komponente.
- konstruktivni karakter: vsak procesni element (nevron) predstavlja mikro-atribut, ki hkrati kombinira več osnovnih atributov; vsak mikro-atribut opisuje dano domeno iz svojega zornega kota, ki pa ga je težko interpretirati
- ker je vsak podatek v nevronski mreži predstavljen z mnogo elementi in vsak nevron vključen v reprezentacijo več podatkov, odstranitev enega nevrona povzroči delni padec v natančnosti za več podatkov in ne popolno izgubo enega podatka (Hinton in sod. 1986)
- spontano spominjanje pozabljenih podatkov brez ponovnega učenja istih podatkov med ponovnim učenjem drugih podatkov (Hinton in sod. 1986, Hinton & Sejnowski 1986).

Ko je mreža naučena (uteži na vezeh fiksirane), deluje tako, da nevroni dobijo začetne vrednosti (vhodni vzorec) in zatem neodvisno drug od drugega spreminjajo svoja stanja glede na vhodne obtežene signale. Delovanje je paralelno in asinhrono. Delovanje mreže se ustavi, ko ni več nobene spremembe v stanjih nevronov. Pravimo, da je nevronska mreža prišla v stabilno (fiksno) točko.

Za razliko od lokalne reprezentacije v računalnikih, kjer podatek lokalno adresiramo, pri porazdeljeni reprezentaciji v nevronski mreži podatek adresiramo s podatkom samim (to velja za auto-asociativni pomnilnik, glej razdelek 4.2.1). Če je adresa točna, dobimo identičen podatek, sicer dobimo podatek, ki je po določenih kriterijih podoben adresi oziroma vhodnemu podatku. Temu pravimo tudi vsebinsko naslavljanje. V nevronski mreži torej ni razlike med podatkom in njegovo adreso.

Za reprezentacijo popolnoma različnih podatkov ali podatkov na različnih nivojih abstrakcije zahteva porazdeljena reprezentacija različne module (podatek je lokalno shranjen z globalnega gledišča, vendar globalno shranjen z lokalnega gledišča).

3. LASTNOSTI NEVRONSKIH MREŽ

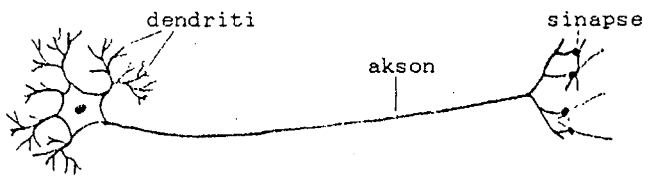
Lepe lastnosti nevronskih mrež so biološka podobnost, visok paralelizem in asinhrono izvajanje, večsmerno izvajanje, prilagodljivost v realnem času, robustnost glede na okvare in glede na manjkajoče podatke, sposobnost učenja in avtomatska generalizacija, mreža ne potrebuje programske opreme in eksplicitne konfiguracije, ni razlike med podatki in adreso, področje ima dobro matematično podlago. Glavna pomankljivost nevronskih mrež je nezmožnost razložiti svojo odločitev. Za vsako lastnost lahko najdemo izjemo, in kolikor je avtorju znano, ne obstaja nevronska mreža, ki bi imela vse naštete lastnosti.

3.1 Biološka podobnost

Pristop do reševanja problemov z nevronskimi mrežami skuša oponašati delovanje človeških možganov in hkrati doseči večjo učinkovitost pri reševanju kompleksnih problemov. Nevronske mreže so abstrakcija in poenostavitev možganskih celic. Nevroni v različnih delih možganov se med seboj močno razlikujejo po obliki, vendar je njihova osnovna struktura enaka.

Primer možganskega nevrona je shematično prikazan na sliki 3. Akson prenaša dražljaje iz celičnega telesa, dendriti pa vodijo vhodne dražljaje v celico. Akson se na koncu razveji in se preko povezav, imenovanih sinapse, povezuje z dendriti ostalih nevronov (Russell 1979).

Aproksimacija z modeli nevronskih mrež kljub poenostavitvam zadostuje za učinkovito modeliranje in analizo. Te raziskave vodijo tudi do boljšega razumevanja procesov v možganih. Več o tem je napisano v razdelku 6.



Slika 3: Shematični prikaz možganskega nevrona

3.2 Paralelizem

Visok paralelizem v nevronskih mrežah je posledica dejstva, da vsak nevron deluje neodvisno od ostalih. To omogoča izredno hitro izvajanje v celoti. Zato so prav nevronske mreže zmožne prilagajanja v realnem času. Večina današnjih aplikacij je simuliranih s klasičnimi računalniki. Deluje pa že nekaj prototipov bolj ali manj specializirane strojne opreme. Implementacijo olajšuje dejstvo, da lahko nevroni delujejo asinhrono.

Primer manj specializirane strojne opreme, ki je primerna za implementacijo nevronskih mrež je t.i. povezovalni računalnik (connection machine). Blelloch in Rosenberg (1987) sta primerjala hitrosti izvajanja posplošenega delta pravila (glej razdelek 4.3.2) na različnih računalnikih vključno s povezovalnim računalnikom, ki je sestavljen iz 65536 enobitnih procesorjev. Zaradi paralelnega izvajanja je bilo izvajanje na povezovalnem računalniku dvakrat hitrejše kot na računalniku Cray-2, ki je nevronsko mrežo simuliral zaporedno.

Graf je s sodelavci (1988) razvil čip za implementacijo algoritmov nevronskih mrež. Uporablja se lahko kot asociativni pomnilnik ali za klasifikacijo vzorcev. Čas izvajanja v čipu (zaporedje iteracij do fiksne točke) je zanemarljiv (cca 1 urin cikel) glede na potreben čas za vhod/izhod (25-50 urinih ciklov).

3.3 Večsmerno izvajanje

Pri nevronski mreži, kjer je vsak nevron povezan z vsakim, je vsak nevron hkrati vhoden in izhoden. Nevronska mreža v tem primeru ne dela razlike med vhodom in izhodom. Poljubna podmnožica nevronov je lahko vhodna. Delovanje mreže bo pri danem vhodu aproksimiralo neznane vrednosti, ki bodo v tem primeru izhod. Primer take mreže je na sliki 1.

3.4 Robustnost

Nevronska mreža je robustna glede na okvaro posameznih nevronov in sinaps. Robustnost je posledica porazdeljene reprezentacije. Ker je vsak nevron vključen v reprezentacijo več podatkovnih elementov in ker je vsak podatkovni element predstavljen z množico nevronov in sinaps, z odstranitvijo enega nevrona ali ene sinapse ne izgubimo nobenega podatka v celoti. Taka okvara povzroči manjšo napako pri večjem številu shranjenih podatkov. Natančnost nevronske mreže pada sorazmerno s številom uničenih nevronov (kar ustreza delovanju možganov - glej razdelek 6).

Nevronska mreža je robustna tudi glede na pomankljive vhodne podatke. To izvira iz načina delovanja nevronske mreže. V mreži, kjer je vsak nevron povezan z vsakim, se bodo manjkajoči vhodni podatki aproksimirali že v prvi iteraciji iz razpoložljivih vhodnih podatkov in naučenega znanja. Primer take aproksimacije je podan v razdelku 1. Seveda bo aproksimacija tem slabša, čim pomankljivejši so vhodni podatki.

3.5 Učenje

Učenje v nevronskih mrežah poteka s spontanim spreminjanjem uteži na sinapsah. Ena pomnilniška celica (sinapsa) predstavlja povezanost med aktivnostima dveh nevronov, ki ju sinapsa povezuje. Ker se utež spreminja glede na spreminjanje vhodnih podatkov (okolja, učnih primerov), utež predstavlja povezanost glede na vse situacije, v katerih se je mreža do sedaj nahajala oziroma glede na vse do sedaj prezentirane učne primere. Ko je mreža naučena, je vsak nevron sposoben napovedati svoje stanje v odvisnosti od danih stanj z njim povezanih nevronov.

3.6 Relacija med strojno in programsko opremo

V nevronski mreži ni programske opreme v klasičnem pomenu te besede. Edini algoritem je algoritem, po katerem deluje individualni nevron. Vse ostalo delovanje je spontano. Mreža teži k stabilni točki tako, da vsi nevroni paralelno in asinhrono delujejo. Besedi računalnik ali izpeljevalnik (inference machine) nista več primerni za opis delovanja takega stroja. Bolj primerno ime bi bilo npr. relaksator.

Strojna oprema je v nevronski mreži lahko fiksna, lahko pa se tudi spreminja (sinapse lahko propadajo ali se pojavijo nove). Za določeno izvajanje konfiguracija ni enolično določena. Za določene statistične lastnosti ni nujno, da je vsak nevron povezan z vsakim. Zadostuje, da je število povezav med nevroni mnogo večje od števila nevronov (Kohonen 1984). Konfiguracija sama pa je lahko naključna, kar je analogno z možgani (glej razdelek 6).

3.7 Matematična podlaga

Področje ima dobre teoretične temelje v linearni algebri (Jordan 1986, Kohonen 1984). Pojmi, kot so linearna transformacija, inverzna transformacija, linearnost, ortogonalnost vektorjev, lastna vrednost, lastni vektor, fiksna točka in konvergenca, so formalno definirani in dobro obdelani. Vse več je teoretičnih prispevkov, vezanih na nevronske mreže (Guez in sod. 1988, Kosko 1988, McEliece in sod. 1987, Smolensky 1986, Williams 1986, Wong 1988).

3.8 Razlaga odločitve

Največja slabost nevronskih mrež je težavnost razložiti svojo odločitev. Z gledišča raziskovalcev umetne inteligence sistem, ki svoje odločitve ne more razložiti, ni boljši od katerega koli statističnega sistema za razpoznavanje vzorcev, ki lahko dobro deluje, vendar interpretacija njegovih rezultatov ni možna brez veliko znanja in izkušenj iz matematike in statistike (glej razdelek 7).

4. VRSTE NEVRONSKIH MREŽ

Nevronske mreže delimo po naslednjih kriterijih:
- topologija nevronske mreže
- namen nevronske mreže
- pravilo učenja
- funkcija kombiniranja vhodov nevrona v izhod

4.1 Topologija nevronske mreže

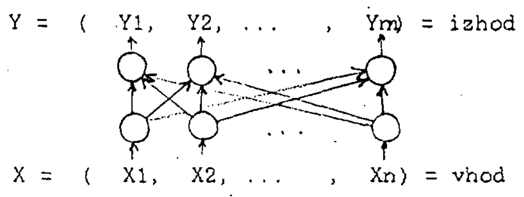
4.1.1 Brez nivojev

Najbolj splošna oblika nevronske mreže je taka, da je vsak nevron hkrati vhoden in izhoden in da je vsak nevron povezan z vsakim nevronom v obeh smereh. Primer je podan na sliki 1.

Nevronska mreža deluje tako, da se na začetku nevronom vsili vrednosti komponent vhodnega vzorca, zatem pa nevroni paralelno sinhrono ali asinhrono spreminjajo svoja stanja in izhod glede na vhod toliko časa, dokler se izhod celotne mreže ne ustali.

4.1.2 Dvonivojske nevronske mreže

Dvonivojska nevronska mreža je sestavljena iz skupine vhodnih in skupine izhodnih nevronov. Vsak vhodni nevron je z enosmerno vezjo povezan z vsakim izhodnim nevronom (glej sliko 4). Izračun take nevronske mreže poteka tako, da se vhodnim nevronom vsilijo vrednosti komponent vhodnega vzorca in zatem v enem koraku vsi izhodni nevroni paralelno izračunajo izhodne vrednosti. Včasih pravijo takim mrežam tudi enonivojske, ker so vhodni nevroni samo senzorji. Primer take mreže je perceptron (Minsky & Papert 1969). Dvonivojske nevronske mreže lahko rešijo samo linearne probleme (npr. 'ekskluzivni ali' ni rešljiv z dvonivojsko mrežo).

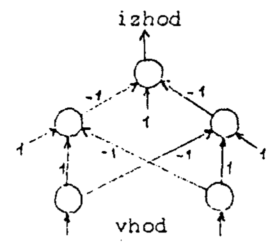

Y = (Y1, Y2, ... , Ym) = izhod
...
...
X = (X1, X2, ... , Xn) = vhod

Slika 4: Dvonivojska nevronska mreža

4.1.3 Večnivojske nevronske mreže

Če v dvonivojski nevronski mreži med vhodni in izhodni nivo dodamo še enega ali več skritih nivojev, dobimo večnivojsko nevronsko mrežo. Večnivojske nevronske mreže delujejo podobno kot dvonivojske, le da je število korakov enako številu skritih nivojev plus ena. Z večnivojskimi mrežami lahko rešimo tudi nelinearne probleme (glej 4.3.2).

Primer tronivojske mreže, ki izračunava 'ekskluzivni ali', je na sliki 5. Dva nevrona sta vhodna, dva skrita in en izhodni. Skrita nevrona in izhodni nevron imajo tudi poseben konstanten vhod z utežjo 1. Mreža je simetrična glede na vhod. Ce sta oba vhodna nevrona v enakem stanju (oba 1 ali oba -1), potem sta oba skrita nevrona aktivna (imata izhod 1) in izhodni nevron postane pasiven (izhodna vrednost je -1). Ce sta vhoda različna (eden 1 in drugi -1), potem sta tudi stanji skritih nevronov različni in je izhodni nevron aktiven (izhodna vrednost je 1).



vhod	izhod
1 1	-1
1 -1	1
-1 1	1
-1 -1	-1

Slika 5: Tronivojska nevronska mreža, ki izračunava ekskluzivni ali.

4.1.4 Dvosmerni asociativni pomnilnik

Dvosmerni asociativni pomnilnik (Kosko 1987,1988) je sestavljen iz dveh nivojev nevronov, ki sta povezani med seboj z dvosmernimi vezmi. Izračunavanje poteka tako, da nevronom ustreznega nivoja vsilimo vrednosti komponent vhodnega vektorja. Zatem nevroni iz drugega nivoja izračunajo svoje izhode, zatem nevroni iz prvega nivoja izračunajo svoje izhode in tako iterativno naprej, dokler se izhodi obeh nivojev ne ustalijo. Delovanje nevronov je lahko sihrono ali asihrono. Že samo ime pove, da se taka mreža uporablja kot asociativni pomnilnik, ki na osnovi vhodnega vzorca izračuna ustrezen vzorec, ki je bil shranjen kot par danemu vhodnemu vzorcu. Dokazano je, da pri asinhronemu izvajanju dvosmerni asociativni pomnilnik konvergira pri poljubno izbranih utežeh na vezeh med nevroni (Kosko 1988).

4.2 Namen nevronske mreže

4.2.1 Avto-asociativni pomnilnik

Avto-asociativni pomnilnik ponavadi deluje tako, da so vsi nevroni hkrati vhodni in izhodni. Na vhodu damo vzorec ali del vzorca (pomankljiv vzorec). Mreža nato iterativno generira (aproksimira) manjkajoči del vzorca in popravlja napačne dele (šum) v vhodnem vzorcu. Ko se izvajanje ustali (nadaljnje iteracije ne spremenijo več vzorca), dobimo na izhodu celoten, dopolnjen in popravljen vzorec. Če je vhodni vzorec preveč pomankljiv ali napačen, se seveda lahko zgodi, da je dobljeni vzorec napačen. McEliece s sod. (1987) je dokazal, da z N nevroni lahko eksaktno shranimo N/(4*log N) vektorjev (N dimenzionalnih). Wong (1988) je empirično pokazal, da mreža z N nevroni lahko shrani N vektorjev.

4.2.2 Hetero-asociativni pomnilnik

Hetero-asociativni pomnilnik je varianta avto-asociativnega pomnilnika, pri kateri je vzorec sestavljen iz več podvzorcev. Na vhodu damo enega ali več podvzorcev, na izhodu pa dobimo preostali del vzorca.

4.2.3 Časovni asociativni pomnilnik

Časovni asociativni pomnilnik deluje kot hetero-asociativni pomnilnik, le da je na vhodu vedno vzorec iz nekega časovnega intervala, izhodni vzorec pa je napoved vzorca za naslednji časovni interval (Kosko 1988). Torej časovni asociativni pomnilnik deluje kot naivni kvalitativni simulator, ki na osnovi dane situacije generira naslednjo situacijo.

Učenje časovnega asociativnega pomnilnika poteka na osnovi primerov. Torej simulator upošteva samo površinsko znanje brez poznavanja zakonov in relacij, ki veljajo v dani problemski domeni. Zato je napovedovanje aproksimacija z upoštevanjem primerov, ki so bili mreži prej prezentirani, in ne izpeljava iz znanih zakonov iz domene.

Časovni asociativni pomnilnik lahko realiziramo tudi tako, da upošteva kontekst in določeno predznanje, kjer opis dane situacije dopušča več alternativnih nadaljevanj. V tem primeru je vhod razdeljen na podvzorec trenutne situacije in podvzorec konteksta.

4.2.4 Klasifikacija

Klasifikacija je poseben primer hetero-asociativnega pomnilnika, ki ima fiksno določene vhodne nevrone in fiksno število izhodnih nevronov, ki določajo razred. Učenje poteka na osnovi primerov z znanimi vhodnimi podatki in znanimi razredi. Primer, za katerega so znani samo vhodni podatki ali samo del vhodnih podatkov, se klasificira v enega od razredov.

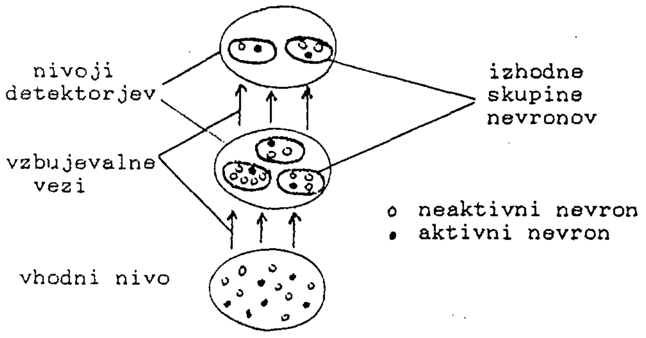
4.2.5 Grupiranje

Tu je problem podan tako, da mora sistem sam znati grupirati vhodne primere v fiksno število razredov. Število primerov je lahko v naprej podano ali pa tudi ne. Spodaj je opisan primer nevronske mreže iz (Rumelhart in Zipser 1986), ki zna grupirati primere v vnaprej določeno število razredov.

Mreža je sestavljena iz vhodnih in izhodnih nevronov. Vsak nevron je lahko aktiven ali pa neaktiven. Vsak vhodni nevron je povezan z vzbujevalnimi vezmi z vsakim izhodnim nevronom. Število izhodnih nevronov določa število končnih razredov. Vsak izhodni nevron je povezan z zaviralnimi vezmi z vsemi ostalimi izhodnimi nevroni, tako da je lahko naenkrat aktiven samo en izhodni nevron.

Pred učenjem ima vsak izhodni nevron na vezi z vsemi vhodnimi nevroni enako utež in sicer tako, da je vsota vseh uteži enaka 1. Med učnjem izhodni nevron premika uteži od neaktivnih vhodov k aktivnim tako, da vzdržuje vsoto uteži enako 1. Najbolj spreminja svoje uteži tisti izhodni nevron, ki postane aktiven (zmaga). Zato pravimo takemu učenju tudi tekmovalno učenje (competitive learning). Sčasoma se mreža nauči, da določen izhodni nevron zmaga vedno pri vhodnih vzorcih, ki so si po določenih lastnostih podobni.

Če povežemo več izhodnih skupin nevronov vzporedno brez medsebojnih povezav, lahko hkrati dobimo več različnih grupiranj. Vsaka izhodna skupina nevronov se nauči grupirati vhodne primere glede na določene lastnosti. Lahko rečemo, da se vsak nevron nauči detektirati določen nov atribut vhodnega vzorca. Zato takemu učenju pravimo tudi odkrivanje atributov (feature discovery). En nivo detektorjev atributov lahko serijsko povežemo v več nivojev (glej sliko 6). Vsak naslednji nivo bo dobil vhodne primere, opisane z nekimi novimi atributi.



Slika 6: Nevronska mreža za grupiranje

4.2.6 Samoorganizacija in sortiranje

Včasih je za učinkovito predstavitev informacije potrebno nevrone urediti tako, da vsak odgovarja na določeno vrednost vhodnega parametra (npr. frekvenca zvoka). Z ustrezno topologijo nevronov dosežemo, da se nevroni sami uredijo tako, da ustrezno ležeči nevroni odgovarjajo na ustrezne vhodne signale.
Primer take mreže je dvonivojska mreža, kjer je vsak izhodni nevron povezan še z bližnjimi izhodnimi nevroni s povratnimi vezmi. Z najbližjimi nevroni je povezan z vzbujevalnimi vezmi in z manj bližjimi z zaviralnimi vezmi (Kohonen 1984). Vsak nevron sprejema na vhodu isti signal, ki ustreza realni vrednosti danega vhodnega parametra. Vsak nevron ima spremenljivo utež za vhodni signal, ki jo primerja z vhodnim signalom. Čim manjša je razlika uteži in vhodnega signala, tem močnejši bo nevronov odgovor.

Učenje poteka tako, da se spreminjajo samo vhodne uteži zmagovalnih nevronov, to je nevronov, ki pri danem vhodu postanejo aktivni. Vhodne uteži se spreminjajo v smeri vhodnega signala. Sčasoma bodo v nevronski mreži vhodne uteži urejene, torej bo lega nevrona, ki reagira na signal dane velikosti, odgovarajala velikosti vhodnega signala. Kohonen (1984) je pokazal konvergenco tega procesa pod določenimi pogoji.

V gornjem učenju Kohonen navaja dva procesa. Eden je ta, da uteži skušajo aproksimirati trenutni signal. Drugi pa je ta, da lokalne interakcije med nevroni skušajo ohranjati kontinuiteto. Analogijo lahko najdemo v algoritmu sortiranja po metodi mehurčkov. Dva sosednja elementa se podpirata, če sta v pravilnem vrstnem redu. To ustreza drugemu principu. Če pa nista v pravilnem vrstnem redu, prvi princip prevlada in vrednosti sosednjih elementov se zamenjata.

4.3 Pravilo učenja

4.3.1 Hebbovo pravilo

Osnovno pravilo, ki ga v različnih variantah uporabljajo nevronske mreže pri učenju, je definiral že Hebb (1949). Pravilo pravi, da se vez med dvema aktivnima nevronoma ojača (utež se poveča). To pravilo zadostuje, da si mreža zapomni frekvenco (in s tem verjetnost), s katero sta dva sosednja nevrona hkrati aktivna. Biološka verjetnost tega pravila (verjetnost, da se po takem ali podobnem pravilu učijo možgani) je v uporabi informacije, ki je lokalno dosegljiva vsakemu nevronu.

Posplošeno Hebbovo pravilo (Rumelhart in sod. 1986) pravi, da se vez med dvem nevronoma ojača, če sta oba hkrati aktivna ali če sta oba hkrati pasivna. Na ta način si mreža zapomni povezanost med aktivnostima dveh povezanih nevronov.

4.3.2 Pravilo delta

Preprosto pravilo, ki omogoča učenje v dvonivojski mreži, sta definirala že Minsky in Papert (1969). V dvonivojski mreži je prvi nivo nevronov vhoden in drugi izhoden. Vsak vhodni nevron je povezan z vsakom izhodnim nevronom. Pred začetkom učenja so uteži izbrane naključno. Učenje poteka z znanimi pari vhodnih in izhodnih vzorcev. Vhodni nevroni dobijo vhodni vzorec. Zatem se v enem koraku izračuna izhod pri danem vhodu. Izračuna se razlika med dejanskim izhodom in željenim izhodom. Zatem se uteži povezav med vhodnimi in izhodnimi nevroni zmanjšajo ali povečajo sorazmerno razliki med dejanskim in željenim izhodom.

Z iterativnim prikazovanjem znanih parov vhodnih in izhodnih vzorcev dosežemo, da se mreža nauči pravilno odgovarjati na vse vhode. Minsky in Papert (1969) sta dokazala, da ta algoritem konvergira k fiksni točki, če je funkcija, ki se jo mora mreža naučiti, linearna. Dejansko dvonivojska mreža ne zmore rešiti nelinearnih problemov (npr. ekskluzivni ali).

Rumelhart je s sodelavci (1986) razvil posplošeno pravilo delta. Le ta omogoča učenje mreže, sestavljene iz poljubnega števila nivojev. Večnivojska mreža lahko reši tudi nelinearne probleme (npr. ekskluzivni ali, glej sliko 5).

Osnovni princip posplošenega pravila delta je isti, kot pri navadnem pravilu delta. Na začetku so uteži naključne. Na vhodu mreža dobi vhodni vzorec in s propagiranjem po nivojih do izhodnega nivoja izračuna izhod. Zatem se izračuna razlika med dejanskim in željenim izhodom. Najprej se spremenijo uteži med zadnjim in predzadnjim nivojem kot pri osnovnemu pravilu delta. Zatem se izračunajo želene vrednosti nevronov na predzadnjem nivoju (to je ključni korak algoritma). Izračuna se razlika med želenim in dejenskimi vrednostmi nevronov na predzadnjem nivoju in rekurzivno se nadaljuje spreminjanje uteži vse do vhodnega nivoja nevronov.

Za posplošeno pravilo delta velja, da ne konvergira vedno (lahko obtiči v lokalnem minimumu). Razen tega zahteva zelo veliko število prehodov preko učnih primerov, t.j. mreži moramo velikokrat pokazati iste učne primere (tipično nekaj sto do nekaj tisoč krat). Tako bi nevronska mreža na sliki 5 potrebovala okoli 500 prehodov preko štirih učnih primerov za rešitev problema XOR (Rumelhart in sod. 1986). Prav tako pravilo delta in posplošeno pravilo delta nimata biološke analogije z možgani (Wassermann & Schwartz 1988).

4.3.3 Tekmovalno pravilo

Pri tekmovalnem pravilu se zahteva, da je v določeni skupini nevronov aktiven vedno natanko eden. To se doseže z ustrezno topologijo, kjer je v skupini vsak nevron povezan z vsakim z zaviralnimi vezmi. Ko en nevron postane aktiven, z zaviralnimi vezmi 'zatre' ostale nevrone, da ne morejo postati aktivni. Pri tem pravilu se uči samo zmagovalni nevron in sicer tako, da poveča uteži vezem, ki so mu pomagale, da je 'zmagal', in zmanjša uteži ostalim vezem.

Primer tekmovalnega pravila je metoda grupiranja Rumelharta in Zipserja (1986) (glej 4.2.5). Kohonenov primer samoorganizacije, ki tudi vključuje tekmovalno pravilo, je opisan v razdelku 4.2.6.

4.3.4 Pozabljanje

Nekatere metode učenja vključujejo tudi pozabljanje. To dosežejo tako, da prej naučenega ne upoševajo enakovredno z novo naučenim. Na ta način je novo naučeno vedno "najbolj sveže". Včasih pozabljanje realizirajo tako, da vse uteži med nevroni s časom zvezno znižujejo v sorazmerju z njihovimi velikostmi. Ta metoda preprečuje, da bi se uteži preveč povečale (Kohonen 1984).

4.4 Funkcija kombiniranja vhodov nevrona v izhod

Funkcija kombiniranja je sestavljena iz dveh delov (glej sliko 2): funkcija aktivacije (A) in izhodna funkcija (f). Funkcija aktivacije kombinira vhode v vmesno vrednost. Izhodna funkcija, ki je tipično pragovni logični element, preslika rezultat aktivacije v izhod nevrona.

4.4.1 Funkcija aktivacije

Najbolj pogosta funkcija aktivacije je obtežena vsota:

$$A(Xj) = \sum_i Wij * Xi + Cj$$

kjer je Wij utež vezi med i-tim in j-tim nevronom, Xi stanje
i-tega nevrona in Cj konstantna aktivacija j-tega nevrona (prag). Če opišemo stanja n nevronov z vektorjem

$$X = (X1, X2, \ldots, Xn)^T$$

in uteži vezi Wij med nevronoma i in j z matriko

$$M = \begin{bmatrix} W11 & W12 & \ldots & W1n \\ W21 & W22 & \ldots & W2n \\ \ldots & & & \ldots \\ Wn1 & Wn2 & & Wnn \end{bmatrix}$$

dobimo stanje Y nevronske mreže po eni iteraciji z

$$Y = M * X$$

Ker matrika M predstavlja linearno transformacijo, je funkcija aktivacije Y = A(X) linearna.

Bolj splošna je t.i. funkcija sigma-pi, katere splošna oblika je

$$sp(X1, \ldots, Xn) = \sum_{S_i \in P} Wi * \prod_{k \in S_i} Xk ,$$

kjer je P potenčna množica množice indeksov vseh nevronov. Si je množica indeksov nevronov, katerih stanja se med seboj zmnožijo in nato se zmnožek 'obteži' z utežjo Wi.

Williams (1986) je pokazal, da s funkcijo sigma-pi in s pragovnimi elementi lahko realiziramo poljubno monotono boolovo funkcijo. Pri tej funkciji je pomembno to, da je en sam nevron ne more realizirati, ker število podmnožic Si narašča s številom vhodnih nevronov. Vsak nevron ima na razpolago toliko spominskih enot za uteži Wi, kolikor je vhodnih nevronov. Zato pa lahko s kombiniranjem več nevronov realiziramo poljubno monotono funkcijo.

4.4.2 Izhodna funkcija

Izhodna funkcija, ki preslika aktivacijo v izhod nevrona, je lahko deterministična binarna, deterministična zvezna ali stohastična, ki je ponavadi binarna. Pri deterministični funkciji je izhod nevrona enolično določen z vhodom. Pri stohastični funkciji pa je izhod določen samo z določeno verjetnostjo.

Binarna deterministična funkcija je ponavadi pragovni logični element. Izhodne vrednosti so ponavadi 0 in 1 ali pa -1 in 1. Prag je lahko fiksen ali pa se ob učenju spreminja.

Zvezna deterministična funkcija je ponavadi sigmoidna funkcija, ki preslika poljubno realno vrednost na interval od 0 do 1. Izhod nevrona je torej poljubno realno število med 0 in 1. Zvezna funkcija občutno poveča pomnilniško kapaciteto nevronske mreže. Guez s sod. (1988) je pokazal, da ima mreža z N nevroni 3 na N fiksnih točk. Primer take funkcije je

$$f(X) = 1/(1 + e^{-X})$$

Da se izognejo lokalnim rešitvam, uporabljajo v nevronskih mrežah tudi stohastične funkcije kombiniranja. To so funkcije, ki ne dajo ene ali druge vrednosti, vrnejo verjetnost za npr. prvo vrednost. Nevron nato izbere eno od vrednosti z dano verjetnostjo. Proces konvergence je zato počasnejši. Na ta način se lahko izognemo večini manjših lokalnih ekstremov in sistem z veliko verjetnostjo poišče globalni optimum.

V zvezi s stohastičnimi funkcijami je povezana vpeljava temperature sistema po analogiji iz termodinamike. Čim višja je temperatura, tem večja je entropija sistema. Z zniževanjem temperature sistem postaja bolj in bolj determinističen. Pri temperaturi 0 so vse odločitve deterministične. Proces v sistemu začnemo pri visoki temperaturi, ki jo počasi znižujemo. Čim počasneje se temperatura znižuje, tem večja je verjetnost, da bo sistem pristal v globalnem optimumu.
Primer stohastične funkcije je Smolensky-jeva teorija harmonije (1986). Harmonija je obratno sorazmerna Hopfieldovi energiji sistema. Čim večja je harmonija, tem nižja je energija, tem bližje je sistem rešitvi. V globalnem optimumu je harmonija največja. Stohastično funkcijo uporabljata tudi Hinton in Sejnowski (1986) v Boltzmannovih strojih (funkcija sledi Boltzmannovi distribuciji).

5. PRIMERI UPORABE

V zadnjem času je vse več komercialno dosegljivih paketov za razvoj nevronskih mrež (Nestor Development System, Cognitron, NeuralWorks, NetsSet, Axon itd.), prodajajo pa tudi že kartico (ANZA) za IBM-PC kompatibilne računalnike in SUN delovne postaje, ki omogoča nekaj 100 krat hitrejše izvajanje. Paketi omogočajo hitro definicijo nevronske mreže in uporabo nekaterih standardnih modelov nevronskih mrež kot so Hopfieldov, Boltzmannov, Grossbergov, 'back-propagation', itd. Za Nestor Development system navajajo naslednje aplikacije (Nestor Inc. 1988):

- Kontrola kvalitete izdelave trdih diskov, ki v naprej napove možna odstopanja od zahtevane natančnosti, tako da jih pravočasno z ustrezno obdelavo lahko odpravimo. Navajajo 100 % natančnost klasifikacije delnih izdelkov v eno od štirih kategorij.
- Diagnostika na osnovi signala EKG. Navajajo 100 % natančnost ločevanja med normalnim EKG in določeno aritmijo.
- Razpoznavanje ročno zapisanih znakov. Navajajo 97.7 % natančnost, pri tem da lahko sistem izloči znake, ki jih ne more zanesljivo klasificirati.
- Preverjanje podpisov na čekih. Navajajo 4 % podpisov narobe klasificiranih kot napačen podpis, kar daleč presega ljudi strokovnjake.
- Identifikacija objektov z radarjem. Navajajo 100 % natančnost pri določenem problemu v primerjavi z 93% natančnostjo Bayesovega klasifikatorja.

Veliko je aplikacij v računalniškem razpoznavanju govora in računalniškem vidu. Kohonen (1988) navaja od 92 do 97 % natančnost razpoznavanja govora (pisanja po nareku, ne da bi sistem razumel, kaj piše) in 96 do 98 % natančnost razpoznavanja izgovorjenih izoliranih besed iz slovarja 1000 besed. Uporabljena metoda učenja je samoorganiziranje (glej razdelek 4.2.6), kjer se vsak nevron specializira za določen vhodni signal.
Računalniški vid bi moral biti invarianten glede na rotacijo, translacijo in velikost opazovanih objektov. Hinton (1981) je pokazal, kako lahko sistem razpozna vhodni vzorec, čeprav ne ve vnaprej, za kakšen vzorec gre in katera transformacija je potrebna. Ideja je v tem, da sistem poskuša paralelno vse transformacije naenkrat in jih primerja z vsemi shranjenimi vzorci naenkrat (kar je naravni princip v nevronskih mrežah). Z iterativnim ponavljanjem se bosta ojačala tista transformacija in tisti vzorec, ki najbolj ustrezata vhodnemu vzorcu.

Fukushima (1988) je s posebno topologijo nevronske mreže dosegel razpoznavanje cifer invariantno glede na velikost in translacijo in delno invariantno glede na deformacijo. Widrow in Winter (1988) predlagata razpoznavanje v dveh korakih z dvema nevronskima mrežama. Prva mreža bi se naučila spreminjati vhodne vzorce v vzorce invariantne glede rotacije, translacije in velikosti. Druga mreža bi se lahko naučila iz vmesnih vzorcev generirati originalne v standardni poziciji, orientaciji in velikosti.

Grabec in Sachse (1988) sta uporabila nevronsko mrežo za analizo in procesiranje signalov akustične emisije. Pokazala sta, da se z nevronsko mrežo lahko učinkovito reši inverzni problem akustične emisije, t.j. odkrivanje izvora zvoka na netrivialnem problemu, ki je eksaktno praktično nerešljiv. Nevronska mreža se brez uporabe enačb iz elastodinamične teorije iz primerov nauči odkrivati karakteristike izvora akustičnih signalov. Ta aplikacija napoveduje novo generacijo merilnih instrumentov, ki naj bi temeljila na principu nevronskih mrež.

Nevronske mreže lahko hitro najdejo približne resitve NP-polnih problemov (v linearnem ali celo konstantnem času). Pri tem mora število nevronov ustrezati velikostnemu parametru problema. Hopfiled in Tank (1985) sta demonstrirala hitro reševanje problema trgovskega potnika (pri N danih mestih poiskati najkrajšo pot skozi vsa mesta, tako da nobenega mesta ne obiščemo več kot enkrat).

Barto in sod. (1983) zagovarjajo bolj kompleksne modele nevronov. Samo dva (različna) ustrezno povezana nevrona sta se naučila balansirati palico na premičnem vozičku mnogo bolje kot originalni sistem BOXES, ki sta ga razvila Michie in Chambers (1968). En nevron se je na napakah učil kontrolirati voziček iz danega opisa stanja sistema. Drugi nevron pa se je na napakah naučil iz danega stanja sistema napovedovati napake (odkrival je kritična stanja sistema).

Popolnoma drugo področje uporabe nevronskih mrež je kognitivno modeliranje. Nevronske mreže so dovolj močne za simulacijo določenih kognitivnih procesov, po drugi strani pa dovolj preproste in formalno definirane, da jih je razmeroma preprosto modelirati in analizirati. Hood (1986) uporablja nevronske mreže za raziskovanje učenja nižjih organizmov. Pazzani in Dyer (1987) primerjata človekovo učenje konceptov in učenje nevronskih mrež.

6. ANALOGIJA Z MOŽGANI

Možgani so vsebinsko naslovljivi. Mentalni procesi uporabljajo možgane v celoti. Ni možno lokalno ločiti funkcije pomnilnika od procesorjev, kot je to možno v klasičnih digitalnih računalnikih. Pomnilnik v možganski skorji je distribuiranega tipa in ne lokalnega. Najbolj pomemben delež zapomnjenja v možganih prispevajo sinaptične povezave (Kohonen 1984).

Hitrost nevronov v možganih je velikostnega reda milisekund. Percepcija, procesiranje jezika, intuitivno razmišljanje, dostop do spomina zahteva v možganih približno desetinko sekunde, torej kakih 100 korakov. Tako je Feldman definiral "100 koračni program" kot omejitev za izvajanje elementarnih operacij programa, ki bi obrazložil mentalne procese. Realizirati tak program je možno seveda samo z uporabo izredno visoke stopnje paralelizma.

Tako v možganih kot v nevronski mreži učinkovitost počasi (zvezno) pada s številom uničenih nevronov (Russell 1979). Ni nobenega ključnega nevrona, katerega okvara bi 'sesula' celoten sistem. V možganski skorji ni nobenega dela, od katerega bi bili odvisni vsi ostali deli (Rumelhart in McClelland 1986a). Prav tako noben del možganov ni nenadomestljiv. Dokazano je npr., da pri otrocih, rojenih samo z eno možgansko poloblo, le ta prevzame funkcije manjkajoče poloble in otroci odrastejo brez okvarjenih funkcij (Russell 1979).

Človeška percepcija je do določene mere invariantna glede na velikost, translacijo in rotacijo (Kohonen 1984). Fizični sistem je sposoben avtomatično generirati reducirano reprezentacijo. Različna področja v možganski skorji so se specializirala za različne senzorske signale in sicer tako, da ohranjajo topologijo prostorskih senzorjev. Ta princip lahko simuliramo z nevronsko mrežo (glej 4.2.6). Kohonen (1984) navaja, da je precej anatomske in psihološke evidence, da obstojajo v možganski skorji povezave po istem principu, kot je opisano v razdelku 4.2.6.

Za učinkovito delovanje nevronske mreže ni potrebno povezati vsak par nevronov med seboj. Za določeno statistično natančnost zadošča, da je število povezav, če so le-te statistično porazdeljene po mreži, mnogo večje od števila nevronov. Če bi možgani z 10^{12} nevroni in s povprečno 10^{3} povezavami na nevron delovali po principu preprostega pragovnega elementa, imajo zadostno kapaciteto, da z zadostno natančnostjo shranijo vse, kar človek doživi v 100 letih (Kohonen 1984). Tudi psihologi navajajo, da je precej empirične evidence, da si lahko človek zapomni prav vse, kar se mu pripeti v življenju (Russell 1979).

Geni ne določajo vseh povezav v možganih, določene povezave so omejene zaradi prostorske lege, in očitno je del povezav naključen (Rumelhart in McClelland 1986a). V možganih ni niti hardware-a v strogem pomenu besede niti software-a v običajnem pomenu besede (glej razdelek 3.6).

Rekurzija ni domača človekovemu načinu razmišljanja. Rumelhart in McClelland (1986a) navajata stavek "The man the boy the girl hit kissed moved" kot preprost primer rekurzivnega stavka, ki pa je očitno težko razumljiv.

Modeli nevronskih mrež ne sledijo popolnoma analogiji z možgani. Večina nevronov v možganski skorji ima ali vzbujevalne ali zaviralne vezi, kar je v nasprotju z nevronsko mrežo, ki dopušča obe vrste vezi pri enem nevronu. Velikost signalov v možganih je podana s frekvenco impulzov in ne z jakostjo signala samega kot v modelih nevronskih mrež (Kohonen 1984). Današnji modeli nevronskih mrež ne upoštevajo globalne komunikacije, ki v možganih poteka s pomočjo določenih kemičnih snovi, ki se pretakajo po krvi med različnimi predeli v možganih.

7. RELACIJA Z UMETNO INTELIGENCO

Zastarel očitek nevronskim mrežam je ta, da niso zmožne vsega izračunati. Že Minsky in Papert (1969) sta pokazala, da lahko univerzalni računski stroj simuliramo z nevronsko mrežo. Seveda pa ta rezultat nima praktične uporabnosti.

Bolj pereč problem pri nevronskih mrežah je težavnost razložiti svojo odločitev. Kot že rečeno v razdelku 3.8, z gledišča raziskovalcev umetne inteligence tak sistem ni boljši od katerega koli statističnega sistema za razpoznavanje vzorcev, ki lahko dobro deluje, vendar interpretacija njegovih rezultatov ni možna brez veliko znanja in izkušenj na področju matematike in statistike.

Današnje raziskave umetne inteligence so usmerjene v razvoj metod in orodij, ki bi omogočile računalnikom bolj inteligentno obnašanje. Pri tem je temeljna zahteva, da zna sistem svojo odločitev obrazložiti in argumentirati. Inspiracije za razvoj metod pogosto pridejo iz analogije s človekovim načinom razmišljanja, vendar samo na visokem, simboličnem in logičnem nivoju. Tako se raziskovalci ozirajo na to, kaj so možgani zmožni napraviti, ne pa tudi, kako to dejansko napravijo.

Pristop z nevronskimi mrežami prav tako išče analogijo s človeškimi možgani, vendar na nizkem, 'subsimboličnem' nivoju. Pri tem raziskovalcev ne zanima samo, kaj možgani zmorejo, ampak tudi, kako možgani delujejo.

Standardni opis človekovega učenja, ki ga navajajo raziskovalci umetne inteligence, je naslednji (Smolensky 1986). Neizkušeni strokovnjaki pri svojem delu uporabljajo splošna pravila, veljavna v dani domeni. Ker so pravila splošna, je delo z njimi počasno. Sledenje pravilom je zavestno. Sčasoma pa si strokovnjak ob reševanju problemov ustvarja svoja bolj specialna pravila, ki jih lahko uporabi brez poglabljanja v samo teorijo, zato postane njegovo delo hitrejše. Sledenje specialnim pravilom je podzavestno.

Večina prevladujočih računalniških modelov v umetni inteligenci je daleč od biološke podobnosti in analogije. Induktivno učenje (Kononenko 1985), ki temelji na metodah umetne inteligence, generira eksplicitna specialna pravila na osnovi že rešenih problemov, kar zahteva velike računalniške zmogljivosti.

Podobno velja pri nevronski mreži, le da specialna pravila niso eksplicitno shranjena, ampak se dinamično kreirajo po potrebi. Shranjeni vzorci, pridobljeni skozi izkušnje, se lahko kombinirajo dinamično na popolnoma nov način, kar omogoča generiranje pravila, ki ga strokovnjak sam ni nikoli videl, oziroma hitro rešitev problema, s katerim se je strokovnjak prvič srečal. Mreža, ki deluje kot klasifikator, se obnaša tako, kot da pozna pravila. Za izvajanje zadoščajo preproste procesne enote, ki delujejo asinhrono in potrebujejo samo lokalno dosegljivo informacijo.

Lahko bi rekli, da nevronske mreže simulirajo funkcijo desne možganske poloble, ki deluje bolj paralelno in podzavestno (intuitivno), medtem ko metode umetne inteligence simulirajo funkcije leve možganske poloble, katere delovanje je bolj zavestno, logično in zaporedno (čeprav je seveda osnovni princip obeh polobel enak). Z ustrezno združitvijo obeh metod lahko pričakujemo velik skok v zmožnosti in učinkovitosti računalniške obdelave.

Rumelhart in McClelland (1986a) poudarjata, da ko bomo dovolj razumeli mikro nivo, bomo mogoče hoteli formulirati popolnoma drugačne makro nivojske modele. Navajata namišljeni računalnik, ki bi imel primitivne ukaze kot so: "sprosti se v stanje, ki optimalno interpretira trenutni vhod", "dobi iz pomnilnika reprezentacijo, ki maksimalno ustreza trenutnemu vhodu in dodaj manjkajoče podrobnosti k shranjeni reprezentaciji" ter "konstruiraj dinamično konfiguracijo struktur znanja, ki ustreza dani situaciji z ustrezno opredeljenimi spremenljivkami". Takemu sistemu bi bolje ustrezalo ime 'relaksator' kot računalnik.

ZAHVALA

Zahvaljujem se svojim kolegom Marku Bohancu, Marku Coklinu, Andreju Dobnikarju, Matevžu Kovačiču in Tanji Urbančič za natančen pregled rokopisa in za koristne pripombe, ki so precej dopolnile tale prispevek. Zahvaljujem se Andreju Dobnikarju, Bojanu Petku in Igorju Grabcu za skupne napore pri zbiranju literature.

LITERATURA

Barto, A.G., Sutton, R.S. & Anderson C.W. (1983) Neuronlike Adaptive Elements that Can Solve Difficult Learning Control Problems. IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-13, no. 5, pp. 834-846.

Blelloch, G. & Rosenberg, C.R. (1987) Network Learning on the Connection Machine. Proc. of 10th Internat. Conf. on Artificial Intelligence, Milano, August 23-28, pp. 323-326.

Grabec, I. & Sachse, W. (1988) Application of an Intelligent Signal Processing System to Acoustic Emission Analysis. Report #6428, Dep. of Theoretical and Applied Mechanics, Cornell University, Ithaca, New York.

Graf, H.P., Jackel, L.D & Hubbard, W.E. (1988) VLSI Implementation of a Neural Network Model. IEEE Computer, March 1988, pp.41-49.

Guez, A., Protopopsecu, V. & Barhen, J. (1988) On the Stability, Storage Capacity and Design of Nonlinear Continuous Neural Networks. IEEE Trans. on Systems, Man, and Cybernetics, Vol. 18, No. 1, pp. 80-87.

Fukushima, K. (1988) A neural Network for Visual Pattern Recognition., IEEE Computer, March 1988, pp. 65-75.

Hebb, D.O. (1949) The Organization of Behavior. New York: Wiley.

Hinton, G.E. (1981) A parallel computation that assigns cannonical object based frames of reference. Proc. of 7th. Internat. joint conf. on AI.

Hinton, G.E., McClelland, J.L. & Rumelhart, D.E. (1986) Distributed Representations. in Rumelhart, D.E. & McClelland, J.L. (eds.) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Hinton, G.E. & Sejnowski, T.J. (1986) Learning and Relearning in Boltzmann Machines. in Rumelhart, D.E. & McClelland, J.L. (eds.) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Hood, G. (1986) Neural Modeling as one Approach to Machine Learning. in Mitchell, T.M., Carbonell, J.G. & Michalski, R.S. (eds.) Machine Learning - A Guide to Current Research. Kluwer Academic Publichers.

Hopfield, J.J. & Tank, D.W. (1985) "Neural" Computation of Decisions in Optimization Problems. Biological Cybernetics, Vol. 52, pp.141-152.

Jordan, M.I. (1986) An Introduction to Linear Algebra in Parallel Distributed Processing. in Rumelhart, D.E. & McClelland, J.L. (eds.) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Kohonen, T. (1984) Self-Organization and Associative Memory. Berlin: Springer-Verlag.

Kohonen, T. (1988) The "Neural" Phonetic Typewriter. IEEE Computer, March 1988, pp. 11-22.

Kononenko, I. (1985) Strukturno avtomatsko ucenje. Informatica, vol. 9, no. 4, pp. 44-55.

Kosko, B. (1987) Constructing an Associative Memory. Byte, september 1987, pp. 137-144.

Kosko, B. (1988) Bidirectional Associative Memories. IEEE Trans. on Systems, Man, and Cybernetics, Vol. 18, no. 1, pp.49-50.

McClelland, J.L., Rumelhart, D.E. & PDP Research Group (1986) Parallel Distributed Processing, Vol. 2: Psychological and Biological Models. Cambridge: MIT Press.

McClelland, J.L., Rumelhart, D.E. & Hinton, G.E. (1986a) The Appeal of Parallel Distributed Processing. In: Rumelhart, D.E. & McClelland, J.L. (eds.), Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

McEliece, R.J., Posner, E.C., Rodemich, E.R. & Venkatesh, S.S. (1987) The Capacity of the Hopfield Assiciative Memory. IEEE Trans. on Information Theory, Vol IT-33, No. 4, pp. 461-482.

Michie, D. & Chambers, R.A. (1968) BOXES: An Experiment in Adaptive Control. in Dale, E. & Michie, D. (eds.) Machine Intelligence 2. Edinburgh: Oliver and Boyd.

Minsky, M. & Papert, S. (1969) Perceptrons. Cambridge, MA: MIT Press.

Nestor Inc. (1988) The Nestor Development System. Reklamni material.

Pazzani, M. & Dyer, M. (1987) A Comparison of Concept Identification in Human Learning and Network Learning with Generalized Delta Rule. Proc. of 10th Internat. Conf. on Artificial Intelligence, Milano, August 23-28, pp.147-150.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986) Learning Internal Representations by Error Propagation. in: Rumelhart, D.E. & McClelland, J.L. (eds.) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Rumelhart, D.E. & McClelland, J.L. (eds.) (1986) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Rumelhart, D.E. & McClelland, J.L. (1986a) PDP Models and General Issues in Cognitive Science. in: Rumelhart, D.E. & McClelland, J.L. (eds.) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Rumelhart, D.E. & Zipser (1986) Feature Discovery by Competitive Learning. in Rumelhart, D.E. & McClelland, J.L. (eds.) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Russell, P. (1979) The Brain Book. Routledge and Kegan Paul, London in Hawthorne, New York.

Smolensky, P. (1986) Information Processing in Dynamical Systems: Foundations of Harmony Theory. in Rumelhart, D.E. & McClelland, J.L. (eds.) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Wasserman, P.D. & Schwartz, T. (1988) Neural Networks (Part 2): What are They and Why are Everybody so Interested in them Now. IEEE Expert, Vol. 3, pp.10-15.

Widrow, B. & Winter, R. (1988) Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition. IEEE Computer, March 1988, pp.25-39.

Williams, R.J. (1986) The Logic of Activation Functions. in Rumelhart, D.E. & McClelland, J.L. (eds.) Parallel Distributed Processing, Vol. 1: Foundations. Cambridge: MIT Press.

Wong, A.J.W. (1988) Recognition of General Patterens Using Neural Networks. Biological Cybernetics, Vol. 58, no. 6, pp.361-372.

DODATEK: SLOVARčEK NEKATERIH ANGLESKIH STROKOVNIH IZRAZOV

activation function - funkcija aktivacije, ki kombinira vhode nevrona v stanje nevrona

autoassociative memory - avto asociativni pomnilnik:delni vzorec prikliče celotni vzorec

back propagation rule - glej 'generalized delta rule'

competitive learning - tekmovalno učenje; učenje poteka tako, da se uči samo nevron, ki zmaga v tekmovanju - postane aktiven, medtem ko so vsi ostali nevroni iz istega nivoja neaktivni

content addressible memory - vsebinsko naslovljiv pomnilnik (asiciativni pomnilnik); podatek se prikliče, če je podan del podatka

delta rule - pravilo učenja, ki spreminja uteži vezi med nevroni sorazmerno razliki med dejanskim in željenim izhodom

eigen value - lastna vrednost matrike

eigen vector - lastni vektor matrike

energy function - energijska funkcija opisuje stanje v nevronski mreži; lokalni minimum te funkcije ustreza fiksni točki

excitatory connection - vzbujevalna vez med dvema nevronoma: povečanje aktivnosti prvega nevrona poveča aktivnost drugega nevrona

fixed point - stabilna točka (stanje nevronske mreže opisano zvrednostmi za posamezen nevron), ki se z nadaljnimi iteracijami ne spremeni

generalized delta rule - posplošeno pravilo delta (glej 'delta rule'), ki omogoča učenje večnivojskih nevronskih mrež

Hamming distance - število komponent, v katerih se dva vektorja razlikujeta

Hebbian learning rule - pravilo učenja v nevronski mreži, ki ojača povezavo med dvema nevronoma, če sta oba nevrona aktivna

heteroassociative memory - hetero asociativni pomnilnik; vzorec prikliče ustrezno n-terico vzorcev

inhibitory connection - zaviralna vez med dvema nevronoma; povečanje aktivnosti prvega nevrona zmanjša aktivnost drugega nevrona

inner product - skalarni produkt dveh vektorjev

neural network - nevronska mreža

outer product - produkt dveh vektrojev interpretiranih kot matrike, tako da je prvi stolpčni in drugi vrstični vektor; rezultat je korelacijska matrika

output function - izhodna funkcija (tipično pragovni element), ki preslika aktivacijo nevrona (glej activation function) v njegov izhod

radius of attraction - maksimalna razdalja od fiksne točke v asociativnem pomniln., ki še omogoča konvergenco v dano fiksno točko

temporal associative memory - časovni asociativni pomnilnik; na vhodu damo vzorec iz danega časovnega intervala, na izhodu pa dobimo vzorec iz naslednjega časovnega intervala

REKURZIVNI POSTOPEK TESTIRANJA VEČNIVOJSKEGA KOMUNIKACIJSKEGA SISTEMA

INFORMATICA 2/89

Keywords: multilevel communication system, testing, recursive procedure

Tone Vidmar, Jernej Virant
Fakulteta za elektrotehniko
in računalništvo, Ljubljana

P O V Z E T E K: Struktura, arhitektura in sintaksa testnega modela je povzeta po [1] in [13]. Za privzet univerzalni testni model je podan rekurzivni postopek testiranja večnivojskega komunikacijskega sistema (*npr. OSI referenčni model*) v njegovem implementacijskem okolju [13]. Postopek omogoča testiranje sistema od najnižjega komunikacijskega nivoja proti višjim. Predlagan rekurzivni algotitem zagotavlja uporabnost testne metode (*Perturbacija Globalnega Stanja Sistema PGSS*) [2], [3], [4], [5] pri testiranju poljubno kompleksnih komunikacijskih sistemov (*redukcija drevesa globalnih stanj ob večpasovnem testiranju*). Celoten postopek je formaliziran in programabilen kar omogoča, da se predlagana metoda razvije v razred t.i. implementacijskih generatorjev.

I. IZHODIŠČA

Glede na [1] je model komunikaciskega sistema opisan z mrežo končnih t.i. komunikacijskih avtomatov E(i),E*(i),m(i). Termin "komunikacijski avtomat" predstavlja običajen Mealyjev avtomat s specifična sintakso, ki bo opisana v nadaljevanju. E(i) in E*(i) predstavljata model komunicirajočega para, medtem, ko m(i) predstavlja model komunikacijskega medija v širšem smislu [6], [7]. Dogodki komunikacijskega avtomata so pošiljanje in sprejemanje komunikacijskih sporočil (*protokolarnih, servisnih, implementacijsko odvisnih in aplikacijskih [8]*). S stališča avtomatne teorije dogodki predstavljajo vhodno/izhodne abecede komunikacijskega avtomata. Dogodek v modelu se označuje z $=(+, -, #). Komunikacijska sporočila, ki jih komunikacijski avtomat pošilja se označujejo z "-", tista pa, kijih sprejema se označujejo z "+". Lokalni dogodki, nič dogodki, se označujejo z "#" (* Predstavljajo interno funkcionalnost komunikacijskega avtomata.*).
Sekvenco dveh dogodkov [6], [8]:

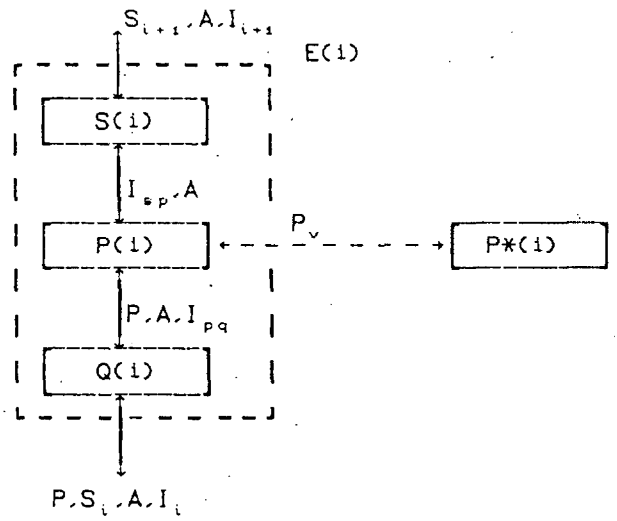
$$(f_{ij}, +e_x(P_p), f_{ik})$$
$$(f_{ik}, -e_y(P_p), f_{il})$$

se v klasičnem Mealyjevem avtomatu interpretira kot, da je vhodna črka e_x/f_{ij} generirala izhodno črko e_y in povzročila prehod v stanje f_{il}. Sprejemno oddajna sekvenca bi Mealyevi notaciji ima naslednjo obliko $f_{ij}, e_x/e_y, f_{il}$.
Podobna interpretacija je sprejemljiva za poljubne kombinacije dogodkov v komunikacijskem avtomatu. Tipe dogodkov [8] se lahko v smislu klasičnega Mealyjevega avtomata razdeli na množico vhodnih črk R..receive- in množico izhodnih črk T..transmit. Lokalni dogodek ustreza t.i. "ε" prehodu [7], [8].
Vsak komunikacijski avtomat je torej mogoče interpretirari kot klasičen Mealyjev avtomat. Ta ugotovitev olajšuje nadalnjo formalizacijo rekurzivnega postopka, ker se lahko privzame klasična avtomatna teorija [9], [10].
V splošnem se množici vhodno/izhodnih črk (* R..vhodna abeceda T..izhodna abeceda *) posameznih komunikacijskih avtomatov delita v podmnožico protokolarnih sporočil P, podmnožico servisnih sporočil S, podmnožico implementacijskih pogojev I in podmnožico aplikacijskih sporočil A. Po definiciji ISO - modela opravlja nižji nivo storitev za višji nivo, ki jo ta zahteva z pošiljanjem servisnih sporočil. Protokolarna sporočila predstavljajo tok virtualnega komunikacijskega kanala za protokolarni par P(i)/P*(i) na istem nivoju. I in A predstavljata realno okolje komunikacijskega nivoja. Za E(i) in E*(i) velja, da morata vsebovati funkcionalnost protokola in obeh protokolarnih vmesnikov S(i) in Q(i) [6], [12]. Odnose omenjenih veličin vidimo na sliki 1.
Trojka E(i),E*(i),m(i) predstavlja mrežo med seboj povezanih komunikacijskih avtomatov:

$$E(i) = \{ F_a, R_a, T_a, W_a, t_a \} = E_a$$
$$E*(i) = \{ F_b, R_b, T_b, W_b, t_b \} = E_b$$
$$m(i) = \{ F_c, R_c, T_c, W_c, t_c \} = m_c$$

ki jih nahajamo na sliki 2. Pri tem veljajo opredelitve:

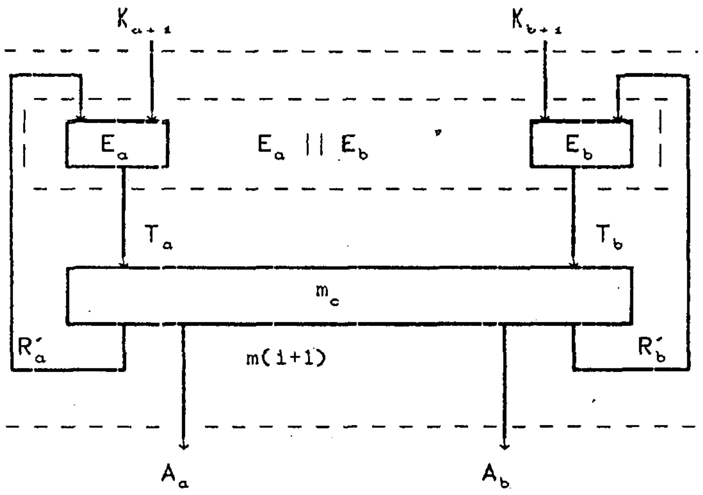
F - množica notranjih stanj komunikacijskega avtomata
R - množica vhodnih črk komunikacijskega avtomata, vhodna abeceda
T - množica izhodnih črk komunikacijskega avtomata, izhodna abeceda
w - logična časovna funkcija prehajanja stanj
t - logična funkcija izhodne črke



Slika 1. Rezina komunikacijskega nivojaISO modela

II. REKURZIVNI POSTOPEK TESTIRANJA

V [2], [3], [4], [5] in [6] je opisana osnovna ideja testiranja protokolov z metodo PGSS. Opredeljene so modifikacije metode, ki so pogojene z uvajanjem robnih pogojev v postopek testiranja [7] [8]. Vendar s tem in uvajanjem dodatnih sintaktičnih konstruktov v testni model problem ekspanzije števila globalnih stanj v drevesu globalnih stanj T še vedno ni zadovoljivo rešen. To je očitno na Sliki 3., podaja testni model sestavljen iz 2N+1 komunikacijskih avtomatov. Testna metoda načelno sicer omogoča testiranje poljubnih mrež med sebој povezanih komunikacijskih avtomatov in tako tudi te, vendar pa sta vprašljiva izvedljivost in časovne performanse testiranja.
Gledano z določenega nivoja proti nižjim komunikacijskim nivojem, zajema prenosni medij v širšem smislu m(i) fizični prenosni medij in vse nižje komunikacijske nivoje. To velja za poljuben komunikacijski nivo, razen za prvi nivo.. Za ta nivo obstaja kot model prenosnega medija samo m(1), preko katerega komunicirata E(1) in E*(1). Če se uspe trojko E(1), E*(1),m(1) nadomestiti z nekim novim avtomatom m(2), je postopek ponovljiv na nivoju E(2)/E*(2) itd. Tako ponavljanje testriranja trojke je osnova rekurzivnega algoritma. Ob tem je topologija testnega modela vedno identična s topologijo, ki je podana na Sliki 2.. Potrebno pa je poiskati način nadomeščanja trojke E(i),E*(i),m(i) z ekvivalentnim komunikacijskim avtomatom m(i+1).



Slika 2. Avtomatna mreža modela

II.1. PARALELNO SERIJSKA KOMPOZICIJA MODELA

Na Sliki 2. vidimo, da se m(i+1) lahko dobi s paralelno kompozicijo E(i) in E*(i) v avtomat QP, ki ji sledi serijska kompozicija dobljenega komunikacijskega avtomata QP z m(i). Ob ugotovitvi, da je komunikacijski avtomat izrazljiv z Mealyjevim avtomatom gre torej za problem paralelno serijske kompozicije takih avtomatov. Formalna osnova paralelno/serijske kompozicije je privzeta po [9] in [10]. Na Sliki 2 nahajamo naslednje relacije med abecedami posameznih komunikacijskih avtomatov:

$$
\begin{aligned}
T_a &= P_b \cup S_a \cup I_{ac} \cup A_b \\
T_b &= P_a \cup S_b \cup I_{bc} \cup A_a \\
T_c &= (P_a \cup I_{ca} \cup A_a) \times (P_b \cup I_{cb} \cup A_b) \\
R_a' &= (P_a \cup I_{ca} \cup A_a) \\
R_b' &= (P_b \cup I_{cb} \cup A_b) \\
R_a &= R_a' \times K_{a+1} \\
R_b &= R_b' \times K_{b+1} \\
R_c &= T_a \times T_b \\
K_{a+1} &= S_{a+1} \cup A_b \\
K_{b+1} &= S_{b+1} \cup A_a
\end{aligned}
$$

A_a/A_a - možica aplikacijskih sporočil

P_a/P_b - množica protokolarnih sporočil, ki jih pošilja avtomat -a/b- in sprejme avtomat -b/a-.

S_a/S_b - množica servisnih sporočil avtomatov E(i)/E*(i)

S_{a+1}/S_{b+1} - množica servisnih sporočil višjih komunikacijskih nivojev

I_{ac}/I_{bc} - množica implementacijsko odvisnih sporočil avtomatov E(i)/E*(i).

I_{ca}/I_{cb} - množica implementacijsko odvisnih sporočil avtomata m(i)

Struktura posameznih komunikacijskih avtomatov -a/b/c- je na osnovi zgornih definicij sledeča:

$$E_a = (F_a,((P_a \cup I_{ca} \cup A_b) X (S_{a+1} \cup A_b),(P_b \cup S_a \cup I_{ac} \cup A_b),w_a,t_a)$$
$$E_b = (F_b,((P_b \cup I_{cb} \cup A_a) X (S_{b+1} \cup A_a),(P_a \cup S_b \cup I_{bc} \cup A_a),w_b,t_b)$$
$$m_c = (F_c,((T_a X T_b),(P_a \cup I_{ca} \cup A_a) X (P_b \cup I_{cb} \cup A_b) ,w_c,t_c)$$

Paralelna kompozicija komunikacijskih avtomatov -a/b- ima sledečo strukturo [9],[10]:

$$QP = ((F_a X F_b),(R_a X R_b),(T_a X T_b),w_p,t_p)$$
$$w_p = (w_a(f_a,r_a),w_b(f_b,r_b))$$
$$t_p = (t_a(f_a,r_a),w_b(f_b,r_b))$$

$$f_a \in F_a$$
$$f_b \in F_b$$
$$r_a \in R_a$$
$$r_b \in R_b$$

Po serijski kompoziciji avtomata QP z komunikacijskim avtomatom -c- se dobi strukturo komunikacijskega medija v širšrm smislu m(i+1):

$$m(i+1) = QP <=> m(i)$$
$$m(i+1) = ((F_a X F_b X F_c),(R_a X R_b),T_c,W_{i+1},t_{i+1})$$
$$W_{i+1} = (w_a(f_a,r_a),w_b(f_b,r_b),w_c(f_c,(t_a,t_b)))$$
$$t_{i+1} = t_c(f_c,(t_a,t_b))$$
$$t_a \in T_a$$
$$t_b \in T_b$$

Dobljena algebrajska struktura je komunikacijski avtomat. Mreža treh komunikacijskih avtomatov E(i),E*(i),m(i) je ekvivalentno nadomeščena avtomatom m(i+1). Med posameznimi podmnožicami komunikacijskih sporočil tega komunikacijskega avtomata veljajo relacije in lastnosti, ki se morajo upoštevati v abecedah komunikacijskega avtomata m(i+1):

L1: - Sporočila iz podmnožic $S_a,S_b,I_{ac},I_{bc},I_{ca}$ in I_{cb} se interno zaključijo v m(i+1) in ne generirajo izhodne črke.

L2: - Sporočila iz podmnožic P_a in P_b, se "skozi" proces -c- prenašajo transparentno od procea -a- k procesu -b- in obratno.

L3: - Sporočila in podmnožic A_a in A_b se transparentno prenašajo skozi sistem.

Iz teh ugotovitev oziroma trditrv sledi, da imajo za testiranje višjega komunikacijskega nivoja E(i+1)/E*(i+1) relevantno vlogo le dogodki povezani z komunikacijskimi sporočili iz podmnožic S_{a+1}, S_{b+1}, A_a in A_b. Minimizirana logična struktura komunikacijskega avtomata m(i+1), ki predstavlja za višji komunikacijski nivo komunikacijski medij v širšem smislu, se formalno dobi tako, da se vse črke iz abeced $P_a,P_b,S_a,S_b,I_{ac},I_{bc},I_{ca}$ in I_{cb} nadomesti z pogoji D.C. -Dont't Care Condition- [10 str. 55, 119]. Dokaz o formalni upravičenosti takega postopka je privzet konceptualno, njegovo formalna osnova pa je podana v [9] na straneh 34, 88, 91 in 146. Postopek uvajanja pogojev D.C. ne zagotavlja optimalne kompresije notranjih stanj komunikacijskega avtomata m(i+1) [10]. Na nekatere stranske efekte postopka redukcije notranjih stanj je opozoril J. Hartmanis v [14] Za konkreten primer, ko gre za strukture elementa m(i+1), se je izkazalo, da je pomembna zgolj korespondenca med stanji strukture m(i+1) po postopku paralelno serijske kompozicije, z reduciranimi stanji, po uvedbi D.C. pogojev. m(i+1) predstavlja servisni vmesnik komunikacijskega nivoja napram višjemu nivoju in model prenosnega medija, ki zagotovi transparentni prenos komunikacijskih sporočil višjega komunikacijskega nivoja. Ob upoštevanju predstavljenega koncepta je struktura komunikacijskega avtomata m(i+1) sledeča:

$$m(i+1) = ((F_a X F_b X F_c),(S_{a+1} \cup A_b) X (S_{b+1} \cup A_b) ,(A_a X A_b),W_{i+1},t_{i+1})$$
$$W_{i+1} = (w_a(f_a,(s_{a+1},a_a)) ,w_b(f_b,(s_{b+1},a_b)))$$
$$t_{i+1} = (t_c(f_c,(a_a,a_b))$$

II.2. GENERACIJA VMESNIKA S(i)

Iz strukture mreže avtomatov ki jih nadomešča avtomat m(i+1) je razvidno, da med posameznimi podmnožicami vhodno/izhodnih črk veljajo sledeče relacije:

$$R_{i+1} = (S_{a+1} \cup A_b) X (S_{b+1} \cup A_a) = (S_{a+1} X S_{b+1}) \cup (A_a X A_b)$$
$$T_{i+1} = (A_a X A_b)$$

$$R_{i+1} \cap T_{i+1} = S_{a+1} X S_{b+1}$$
$$R_{i+1} \cup T_{i+1} = R_{i+1}$$

Še pred izpeljavo gornjih formalnih izrazov, se je pokazalo, da so intuitivno postavljene trditve L1, L2 in L3 na mestu.

Iz gornje razprave izhaja, da je funkcionalno gledano komunikacijski avtomat m(i+1) hibridni model servisnega vmesnika S(i)- in prenosnega medija v ožjem smislu m(1).

Formalen dokaz za zgornjo trditev je mogoče zasnovati na postopku serijske dekompozicije [9], [10] avtomata m(i+1) na komunikacijska avtomata S(i) in m(1). Postopek serijske dekompozicije ni trivialen in možen samo pod posebnimo pogoji. Formalno je definiran spolšen primer serijske dekompozicije [10 str. 97, 137]. Serijska dekompozicija avtomata m(i+1) je možna samo če obstoja particija Π_s, nortanje abecede F_{i+1} tako, da ima Π_s substitucijsko lastnost (*S.P.*) za F_{i+1}. S.P. predstavlja izraz [9]:

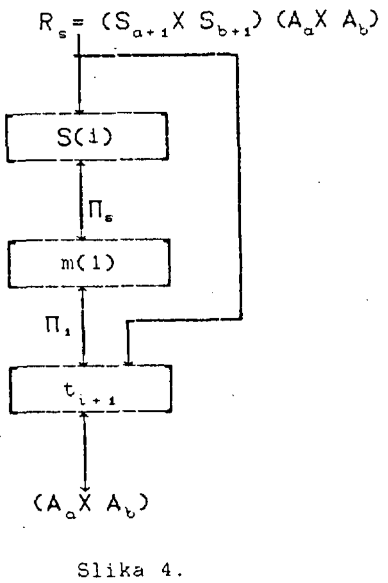
$$W_{i+1}(f_{i+1}, r_{i+1}) = W_{i+1}(f_s, r_{i+1})$$

Za avtomata dobljena s tem postopkom velja:

$$S(i) = (\Pi_s, R_s, W_s)$$
$$m(1) = (\Pi_1, R_1, W_1)$$

$$t: \Pi_1 \times R_s \to T = A_a \times A_b$$

Shematska prezentacija take dekompozicije je sledeča:



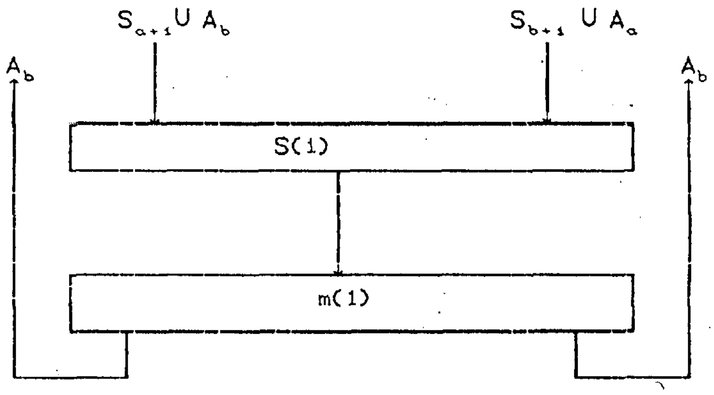
Slika 4.

Pogoj, ki zadošča za izpeljavo serijske dekompozicije je torej sledeč [9]:

$$\forall \exists ((\Pi_s \subset F_{i+1}, \Pi_1 \subset F_{i+1}) \vee (\Pi_s \Pi_1 = P_o) \vee$$
$$\vee (f_{i+1} = W_s(\pi_s)) \vee (W_{i+1}(f_{i+1}, r_{i+1}) = W_{i+1}(f_s, r_{i+1}))$$

P_o = particija niča

Vprašanje pa je, ali ima S(i) substitucijsko lastnost. Dokaz tega je lahko zasnovan na reverzibilnosti postopka kompozicije in dekompozicije [9]. Koncept dekompozicije kot formalen dokaz vsebine m(i+1) potrjuje začetno trditev, da je rekurzivni postopek testiranja tudi aplikativni generator strukture servisnega vmesnika S(i), katerega prikazuje slika 5.



Slika 5.

O opisanem postopku lahko govoromo kot o aplikacijskem generatorju, ker le ta na osnovi določenih vhodnih podatkov avtomatsko generira strukturo vmesnika, ki predstavlja implementacijsko odvisen del rezine komunikacijskega modela.

II.3. MINIMIZACIJA ABECED F, R, S AVTOMATA m(i+1)

Iz definicije globalnega stanja sistema, [1], [2], [3], [4] (matrika globalnega stanja) in drevesa globalnih stanj, kot rezultata perturbacijskega postopka, sledi, da je notranja abeceda avtomata m(i+1) definirana z glavno diagonalo globalnih stanj drevesa globalnih stanj. Samo drevo globalnih stanj je reprezentacija avtomata, zato je na osnovi stanj čakalnih vrst, oziroma stranskih elementov matrike stanj, možno avtomatično generirati tudi abecedi R in T [13 str. 91]. Temu lahko sledi postopek minimizacije opisan v poglavju II.1. Tudi generacija abeced in posredno same strukture avtomata m(i+1) je lastnost rekurzivne metode kot aplikativnega generatorja.

II.4. VPELJAVA VMESNIKA Q(i+1)

Vpeljava vmesnika Q(i+1) je vezana na strukturo komunikacijskega nivoja prikazanaga na sliki 1. Vpeljava je ekvivalentna problemu modeliranja protokola P(i+1)\P*(i+1) na osnovi neformalne specifikacije s tem, da je za vmesnik Q(i+1) znan vmesnik S(i). Isto velja tudi za uvajanje implementacijskih pogojev bodisi na nivoju avtomatov E_a in E_b bodisi na nivoju m_c. Opirati se treba predvsem na strukturo in sintakso testnega modela [8], vse ob upoštevanju robnih pogojev testiranja [8], [13].

III. ZAKLJUČEK

Članek podaja formalno osnovo rekurzivnemu algoritmu -RECALG- večnivojskih komunikacijskih sistemov. Za N - nivojski komunikacijski sistem je struktura rekurzivnega algoritma sledeča:

```
RECALG
  i=1
  MODELIRANJE m(1),S(1)
  DOKLER i <= N
    MODELIRANJE E(i),E*(i),Q(i)
    DOKLER TEST NIVOJA NI POZITIVEN
      GENERACIJA DREVESA GLOBALNIH STANJ
    KONEC
    GENERACIJA ABECED $F_{i+1}, R_{i+1}, S_{i+1}, m_{i+1}$
    MINIMIZACIJSKI POSTOPKI PROCESA m(i+1)
    GENERACIJA S(i)
    i = i + 1
  KONEC
RECALGEND
```

Praktičen primer izvajanja algoritma je podan v [13]. Opis uporabljene programske opreme pa v [6]. Metoda omogoča "držati" dinamične strukture podatkov, ki se tvorijo v procesu perturbiranja, v implementacijsko obvladljivem obsegu, tudi pri testiranju velikih realnih sistemom. Metoda je bila preverjena in polavtomatsko simulirana na velikem realnemm telekomunikacijskem sistemu. V nadaljevanju dela želimo uporabnost metode razširiti za poljubne topologije med seboj povezanih procesov, obstoječo pa avtomatizirati.

LITERATURA:

[1] Pitro Zafiropulo, "Protocol Validation by Duologue-Matrix Analysis", IEEE Transactions on Communications, Vol. Com- 26, decembar, 1980.

[2] Pitro Zafiropulo, Colin H. West, Harry Rudin & Daniel Brand,"Towards Analysing and Synthesizing Protocols", Transactions on Communications, Vol. Com-28, April 1980.

[3] Harry Rudin, Colin H. West, "A Validation Technique for Tihtly Coupled Protocols", IEEE Transactions on Computers, Vol. C-31, July 1982.

[4] Harry Rudin, "Automated Protocol Validation: Some Practical Examples", Proceedings of The Sixth International Conference on Computer Communications Protocol Validation", IBM J. RES. Vol. 22, July 1978.

[5] Colin H. West, "General Technique for Communication Protocol Validation", IBM J. RES. Vol. 22, July 1978

[6] Tone Vidmar, "Modeliranje komunikacijskih protokolv s končnimi avtomati", Mipro 88, maj 1988, Opatija

[7] Tone Vidmar, "Logični protokolarni analizator", Mipro 88, maj 1988, Opatija

[8] Tone Vidmar, "Sintaksa medprocesnega komuniciranja", ETAN 88, junij 1988, Sarajevo

[9] J. Hartmanis, R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice - Hall 1966.

[10] Jernej Virant, Preklopne funkcije, strukture in sistemi, Fakulteta za elektrotehniko 1985.

[12] Tomaž Kalin, Tone Vidmar, "Logično testiranje protokolov", Informatica Vol. 1, Ljubljana januar 1988

[13] Tone Vidmar, Doktorska disertacija, Fakulteta za elektrotehniko, Ljubljana maj 1987

[14] J. Hartmanis "Further Results on the Structure of Sequential Machines", Journal of the Assosiation for Computing Machinery, Vol 10, No. 1, January 1963, 78-88.

UPRAVLJANJE Z IMENI V PORAZDELJENIH SISTEMIH

INFORMATICA 2/89

Keywords: management by naming, distributed systems

Jože Rugelj
Institut Jožef Stefan, Ljubljana

POVZETEK Članek podaja pregled in definicije osnovnih pojmov, ki so povezani z imenovanjem objektov v računalniških sistemih. Imenske strukture pomagajo premagovati razlike med potrebami računalnika in željami njegovega uporabnika. Pomen imenskih struktur se veča s kompleksnostjo sistemov in njihov procesno močjo.

ABSTRACT This article gives an overview and definitions of the basic terms concerning naming of the objects in computer systems. Naming structures can reduce the differences between computer and its user. The importance of naming structures is growing with the complexity of systems and with their process power.

Imena imajo v računalniških sistemih zelo pomembno vlogo, saj z njihovo pomočjo identificiramo izbrane objekte na vseh nivojih abstrakcije. Imena uporabljamo na različnih področjih pri upravljanju, npr. pri zaščiti, nadzoru napak in pri upravljanju objektov ter pri njihovem lociranju in delitvi med več uporabnikov. V porazdeljenih tesno sklopljenih računalniških sistemih je ta vloga še pomembnejša in bolj kompleksna zaradi dovoljene raznolikosti posameznih elementov sistema in njihove porazdeljenosti ter želje, da bi uporabnik videl sistem kot enovito celoto [Ruge88].

1 DEFINICIJE OSNOVNIH POJMOV

Najbolj splošna definicija za ime bi bila:

Ime je niz znakov iz neke abecede, ki označuje neko množico objektov.

Vsa imena, ki jih na osnovih pravil lahko tvorimo nad dano abecedo, sestavljajo *imenski prostor*. Pravila za tvorbo imen pa so lahko zelo različna in so odvisna od kompleksnosti in strukture sistema, v katerem označujemo objekte s tako dobljenimi imeni.

Najbolj enostavna oblika imenskega prostora je *ploščat* imenski prostor. Vsak imenski strežnik mora poznati vsa imena v sistemu. Imena sama ne izražajo nobene strukture. Pri kompleksnejših sistemih mora biti imenskemu sistemu pridružena neka dodatna možnost strukturiranja, ki navzven ni vidna.

Bolj kompleksne strukture imen pa imenujemo *hierarhične*. V tem primeru je ime sestavljeno iz zaporedja enostavnih imen. Enostavna imena so med seboj povezana z ločilnimi simboli. Posamezne komponente sestavljenega hierarhičnega imena lahko vsebujejo informacije o fizični lokaciji imenovanega objekta ali o njegovi organizacijski pripadnosti. Taki podatki sicer pomagajo pri iskanju objektov, hkrati pa zmanjšujejo prilagodljivost sistema kot celote: onemogočajo premikanje objektov v sistemu, nadomeščanje objektov z drugimi objekti in podvajanje objektov. Vse te zahteve pa so aktualne v sodobnih porazdeljenih sistemih.

Tudi same oznake imen so v različnih virih različne. Med prvimi je poskušal probleme z imenskimi strukturami v porazdeljenih sistemih formalizirati J.F. Shoch [Shoc78], ki razlikuje tri vidike identifikacije objektov: ime, naslov in pot. Razlika med identifikatorji je v tem primeru semantična.

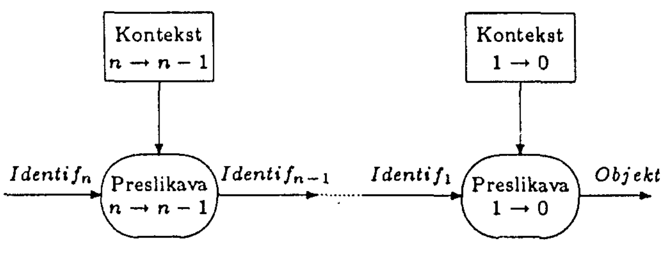
Ime pove <u>kaj</u> iščemo,
naslov kaže <u>kje</u> to je in
pot pokaže <u>kako pridemo tja</u>.

Sintaktična razlika med imeni je v izboru abecede in v pravilih za tvorbo imen, predvsem pri določanju dolžine in pri njeni spremenljivosti. Značilna povezanost semantičnih in sintaktičnih lastnosti identifikatorjev je v tem, da so naslovi običajno fiksne dolžine in sestavljeni iz niza številk, imena pa so spremenljive dolžine in sestavljena iz alfanumeričnih znakov z nekaterimi omejitvami.

J.H.Saltzer v svojem delu [Salt78] obravnava problem imen v centraliziranih operacijskih sistemih, ki se je pojavil zaradi povečevanja računalniških sistemov in združevanja ter prenašanja velikih programskih paketov, vendar v zaključku dela ugotavlja, da so dobljene ideje in ugotovitve uporabne tudi v porazdeljenih sistemih, saj gre pri njih za podobne okoliščine. Za preučevanje imenske problematike uporablja objektni model računalniškega sistema. Računalniški sistem upravlja objekte, ki izvajajo procesiranje. Objekt lahko vsebuje druge objekte, ki jih identificira z imeni. Povezava med imeni in objekti je preslikovalna funkcija, ki je pridružena objektu in jo Saltzer imenuje *kontekst*. Realizirana je kot tabela, ki vsebuje pare $< ime, naslov >$. Tak pristop v različnih oblikah in z različnimi izboljšavami so kasneje privzeli skoraj vsi snovalci imenskih sistemov v porazdeljenih sistemih. Na tak način je dosežena velika fleksibilnost imenske strukture.

Problemi, ki se pojavljajo pri uporabi opisanega imenskega sistema, so imenski konflikti, ki se pojavljajo pri dodajanju neodvisno snovanih objektov in posledice napačno izbranih začetnih kontekstov. Pri bolj kompleksnih sistemih se konteksti pojavljajo na več nivojih zaradi večkratne vsebovanosti objektov. Zaradi fleksibilnosti imenskega sistema in dinamičnega spreminjanja mora uporabnik tudi paziti na konsistentnost danih povezav imen in naslovov. Tudi povezanost imenskega sistema s funkcijami zaščite, upravljanja virov ali upoštevanje faktorjev ekonomičnisti prinaša omejitve in z njimi povezane probleme.

Watson za imena v najširšem smislu pomena uporablja termin *identifikator$_{nivo}$*, da bi se s tem poenotil imensko hierarhijo in se izognil dvoumnostim (slika 1). Imena so definirana na več logičnih nivojih, ki se ločijo po stopnji abstrakcije pri predstavitvi sistema [Wats81].



Kontekst $n \rightarrow n-1$
Kontekst $1 \rightarrow 0$
$Identif_n$
Preslikava $n \rightarrow n-1$
$Identif_{n-1}$
$Identif_1$
Preslikava $1 \rightarrow 0$
$Objekt$

Slika 1: Watsonova imenska hierarhija

Ime se interpretira v okviru nekega *konteksta*. Kontekst je množica povezav imen in objektov. Vsako ime je element vsaj enega konteksta. Postopek iskanja objekta, ki pripada imenu v določenem kontekstu, imenuje *razreševanje imena (resolving)*. Razreševanje imena lahko poteka v več stopnjah. Rezultat razrešitve imena znotraj konteksta lahko namreč da nov kontekst (nižji logični nivo abstrakcije) in šele na najnižjem nivoju dosežemo objekt. Vsak nivo ima svojo imensko strukturo in pravila za dodeljevanje in razreševanje imen. Rezultat razreševanja je običajno identifikator na nižjem logičnem nivoju.

Potreba po več nivojih imen izhaja predvsem iz dveh vzrokov:

- različnosti potreb človeka in računalnika in
- različnosti med lokalnimi in globalnimi potrebami porazdeljenega računalniškega sistema.

Pri izbiri imen se človek odloča za imena, ki so mnemonično uporabna in zato sestavljena iz črk, dolžine imen so različne in zaradi tega, ker jih izbira človek, so lahko tudi dvoumna. Taka imena so za računalnik slabo uporabna, saj zahteva nedvoumna, kratka imena stalne dolžine, nad abecedo z dvema črkama.

Globalna imena so v velikih sistemih lahko zelo dolga in nerodna za uporabo. Zato v lokalnih okoljih lahko uporabimo skrajšana imena ali tako imenovana *domača imena.*

Na izbiro imen pa poleg prej omenjenih faktorjev vplivajo še drugi, predvsem ekonomski, in želja, da imenski sistem neposredno podpira še druge upravljalske funkcije, predvsem zaščito.

Ekonomski faktor običajno vpliva na velikost imenskega prostora, ki je odvisna od velikosti pomnilnika, namenjenega za shranjevanje imenske podatkovne baze in procesorskih zmožnosti, ki morajo biti pri velikem imenskem prostoru in dolgih imenih precej večje. Velik imenski prostor pa ima seveda vrsto prednosti. V njem je praktično neskončno mnogo imen in zato jih po uporabi ni treba ponovno uporabljati, ampak jih enostavno zavržemo. Število imen lahko povečamo tudi tako, da dovolimo različno dolga imena, saj se na ta način zelo poveča število možnih kombinacij znakov iz abecede.

2 STRUKTURA IMENSKEGA PROSTORA

V obstoječih porazdeljenih sistemih je upravljanje z imeni v glavnem odvisno od imenske strukture.

Večina sistemov ima drevesno strukturo imenskega prostora. Veje drevesa imajo oznake, informacija o objektih pa je v listih dreves. Ime objekta predstavlja niz oznak z vej, ki so med korenom drevesa in listi.

Imena v najširšem pomenu besede vsebujejo poleg oznake objekta še dodatne informacije o specifičnih atributih objekta. Primeri takih atributov na uporabniškem nivoju so informacije o upravno-administrativni, geografski ali mrežno topološki pripadnosti objekta [Su83]. Po Shochovi definiciji je ime edini identifikator, ki ne vsebuje nobenih karakterističnih atributov.

Če ime ne vsebuje karakterističnih atributov, ima to svoje prednosti in slabosti. Glavna slabost je v tem, da je razreševanje imena razmeroma kompleksno. Ime ne vsebuje nobenih namigov o lokaciji objekta oziroma o njegovem naslovu. Ravno ta transparentnost lokacije pa je zaželjena pri rekonfiguraciji porazdeljenega sistema, pri združevanju ali cepitvi porazdelejnih sistemov in pri spremembah lokacije objekta. Vse te spremembe vplivajo samo na preslikovalne mehanizme, uporabljene v postopku razreševanja imena.

Kot smo že omenili, imenujemo iskanje povezave imena objekta in njegovega naslova (in s tem lokacije objekta) razreševanje imena. Pravila za imenovanje objektov določajo, v katerem trenutku se izvede razreševanje. Kolikor kasneje se imena preslikajo v naslove, toliko večja je fleksibilnost sistema.

Obravnavo problemov, povezanih s poimenovanjem, lahko poenotimo z vključitvijo v model stranka-strežnik. Vse aktivnosti, ki so potrebne v postopku razreševanje, izvaja imenski strežnik in jih v obliki servisov nudi uporabniku. Imenski strežnik je tesno povezan s pojmom konteksta [Come86]. Kontekst predstavlja tisto particijo imenskega prostora, ki ga upravlja imenski strežnik. Strežniku je tako pridružena preslikovalna funkcija, ki služi za razreševanje imen znotraj particije. Seveda pa je znotraj imenskega prostora lahko več strežnikov in s tem tudi kontekstov.

3 PORAZDELJENO UPRAVLJANJE Z IMENI

Porazdeljeno upravljanje z imeni sestavljajo tri glavne aktivnosti, ki so tesno povezane in sodelujejo med seboj in z uporabniki servisov, ki jih nudijo:

- Dodeljevenje imen
- Porazdeljevanje imenskega prostora
- Razreševanje imen

Dodeljevanje imen

Dodeljevanje imen novim objektom je prepuščeno tistim, ki nov objekt kreirajo. Vsako novo ime pa je treba registrirati pri imenskem strežniku. Le-ta preveri, če tako ime še ne obstaja, da ne bi prišlo do dvoumnosti. V primerih, ko imena, ki ga želi uporabnik registrirati, še ni v sistemu, imenski strežnik tako ime sprejme. Če pa tako ime slučajno že obstaja, strežnik ime zavrne in uporabnik poskuša znova. V porazdeljenih sistemih, kjer je več strežnikov, morajo le-ti komunicirati med seboj pri vsaki registraciji novega imena.

Porazdeljevanje imenskega prostora

Porazdeljevanje imenskega prostora je dodeljevanje odgovornosti za posamezne dele imenskega prostora imenskim strežnikom. Kot smo že prej omenili, je v porazdeljenih sistemih več strežnikov, saj se s tem poveča zanesljivost, hitrost delovanja in prilagodljivost sistema. Zaradi zanesljivosti se lahko imenski prostori posameznih strežnikov tudi prekrivajo, vendar to lahko povzroči težave predvsem pri upravljanju in delitvi odgovornosti

Pooblaščenega imenskega strežnika oziroma več strežnikov določi stranka, ki je objekt kreirala, takoj ko eden od strežnikov potrdi uporabo predlaganega imena. Med delovanjem na podoben način lahko prenesemo pooblastilo drugim strežnikom.

Izbira strežnika je popolnoma neodvisna od imena oziroma oblike imena. Strankam po izbiri strežnika ni treba ohranjati informacije o izbranem pooblaščenem strežniku, saj jo v vsakem trenutku lahko dobi od kateregakoli imenskega strežnika. Vsak strežnik hrani podatke o pooblastilih v *konfiguracijski podatkovni bazi* . V velikih sistemih je taka podatkovna baza prevelika, da bi jo vzdrževal vsak strežnik v celoti, in je porazdeljena. Iskanje določenega podatka tako zahteva sodelovanje med strežniki.

Prostor, ki ga konfiguracijska baza zavzema in čas, ki ga strežnik porabi za njeno vzdrževanje in uporabo, predstavljata ceno za strukturno neodvisno upravljanje z imeni. Pri običajnih imenskih strukturah so namreč vse te informacije vsebovane v samem imenu objekta in v njegovi strukturi.

Skupaj s sporočilom o pooblastilu mora stranka pooblaščenemu strežniku podati tudi atribute imenovanega objekta, ki potem omogočajo strežniku vzdrževanje podatkovne baze z atributi objekta za upravljanje z imeni. Kot smo že poudarili, ime sámo v splošnem ne vsebuje dodatnih informacij o objektu.

Pri izbiri pooblaščenega strežnika oziroma strežnikov je glavno vodilo izpolnitev čimvečjega števila ciljev, ki jih izpolnjujemo v porazdeljenem sistemu. Poudarili bi predvsem zanesljivost delovanja in čim večjo hitrost ter izkoriščenost virov v sistemu. Z izbiro imenskega strežnika, ki je blizu uporabniku ali imenovanim objektom, vplivamo na izpolnitev vseh naštetih ciljev.

Razreševanje imen

Ime lahko razrešimo z zahtevkom poljubnemu imenskemu strežniku v sistemu. Če strežnik ni pooblaščen za delo s podanim imenom, uporabniku vrne ime najbližjega pooblaščenega strežnika ali pa sam posreduje zahtevek le-temu. Šele pooblaščeni strežnik poišče iz podatkovne baze atributov zahtevane podatke. Količina informacije oziroma število atributov v podatkovni bazi je zelo različno, odvisno od potreb in zahtev uporabnikov imenskega servisa.

V primeru, ko so take podatkovne baze replicirane, je treba uporabljati poznane mehanizme za ohranjanje konsistentnosti. Pri tem so zaradi prevladovanja operacije branja in zelo redke uporabe pisanja mehanizmi enostavni.

4 KONTEKSTI - PORAZDELITEV IMENSKEGA PROSTORA

Pri razreševanju imen se se je kot koristna izkazala uporaba *kontekstov* . Ta pojem sta uvedla že Saltzer in Watson [Salt78], [Wats81], dobro pa je obdelana uporaba kontekstov v doktorski tezi [Pete85] in članku [Come86].

Kontekst v splošnem predstavlja razdelitev imenskega prostora na osnovi geografskih, organizacijskih ali funkcionalnih pripadnosti objektov. Bolj določno opredelitev konteksta podaja [Terr86], ki definira kontekst kot zbirko seznamov, ki vsebujejo imena pooblaščenih imenskih strežnikov.

Uporabnik imenujejo objekte v sistemu z imeni, ki so lahko sestavljena iz niza enostavnejših imen, ločenih s posebnimi ločilnimi znaki. Glavna naloga razreševalnega mehanizma je preslikava uporabniških imen v *primitivna imena*, ki imajo zelo enostavno sintakso, so enolična in so fiksne dolžine. To pomeni, da so fizični naslovi objektov ena od oblik primitivnih imen.

Imenski strežniki v porazdeljenih sistemih so po Petersonu tesno povezani s konteksti. Tvorijo jih podatkovne strukture, ki vsebujejo množico povezav in program, ki vrne povezave pridružene danemu imenu. Že Peterson omenja možnost, da interna reprezentacija imenskega strežnika ni nujno identična zunanji predstavitvi imenskega sistema. Tako lahko npr. imenski strežnik hierarhična imena 'stisne' v ploščato strukturo, če mu to olajša delo z imeni. Koncept strukturno neodvisnega upravljanja z imeni je razvijal Terry.

5 STRUKTURNO-NEODVISNO UPRAVLJANJE Z IMENI

D.B. Terry v [Terr86] uvaja pojem strukturno neodvnisnega upravljanja z imeni. Ta je konkretizacija ideje o imenih brez karakterističnih atributov. Pri velikih sistemih, ki se dinamično spreminjajo, je zelo težko ali celo nemogoče vnaprej predvideti vsa stanja sistema in njegove konfiguracije. Dodajanje elementov strojne in programske opreme in priključevanje in odhajanje uporabnikov so aktivnosti, ki se pogosto dogajajo v sistemu, vsako spreminjanje imen pa je drago zaradi možne povezanosti velikega števila objektov. Zato mora biti imenska struktura in delo imenskih strežnikov čimbolj neodvisno od stanja sistema.

5.1 Konteksti v strukturno-neodvisnem upravljanju

Po Terryevi definiciji kontekst predstavlja nedeljivo enoto za shranjevanje podatkov o pooblaščenih strežnikih. Z njim lahko dela več strežnikov. Hkrati ima lahko en strežnik več kontekstov. Imena kontekstov niso pomembna za uporabnika, saj so konteksti skriti v imenskih strežnikih in jih stranke ne vidijo. Konteksti tudi ne vsebujejo nobenih informacij o objektih, katerih imena so povezana z njimi. To pomeni, da sta si drevesna struktura kontekstov ter razreševanje imen in drevo, ki predstavlja poljubno urejenost imenskega prostora, vidno uporabniku, tuja. Tako lahko uporabnik prilagodi imensko strukturo svojim potrebam, s tem pa ne prizadene prilagodljivosti sistema in prednosti strukturno neodvisnega upravljanja z imeni.

Zaradi velike neodvisnosti kontekstov od značilnosti sistema lahko konfiguracijsko podatkovno bazo razstavimo na kontekste z enostavnimi algoritmi, brez posredovanja uporabnikov. Uporabnik lahko vpliva na porazdelitev in dodeljevanje kontekstov strežnikov samo v primeru, če to eksplicitno zahteva. Taki posegi so koristni predvsem pri optimizaciji, izvedemo pa jih z ukazi v t.i. *ortogonalnih jezikih* [Pope85].

Enostaven mehanizem za razdelitev imenskega prostora v kontekste je uporaba *pogoja za razvrščanje v skupine* . Pogoj za razvrščanje imen je enostavna funkcija, ki kot odgovor vrne logično vrednost *PRAVILNO* ali *NEPRAVILNO*, če ji kot argument podamo ime. Imena, pri katerih je odgovor pozitiven, pripadajo kontekstu, katerega last je pogoj za razvrščanje. Razpršilne funkcije so primer takih pogojev za razvrščanje. Pri imenih pa je za razdelitev v kontekste še bolj tipičen način sintaktičnega analiziranja imen in primerjava s podanim vzorcem - nizom znakov iz neke abecede.

Razvrščevalni pogoji so uporabljeni za dodelitev imen kontekstom in za razdelitev kontekstov v manjše kontekste. Postopek začnemo nad celotnim imenskim prostorom in z zaporedno uporabo razvrščevalnih pogojev dobimo željeno velikost kontekstov.

Da bi preverili pripadnost imena nekemu kontekstu je treba nad tem imenom uporabiti vse razvrščevalne pogoje, ki ustrezajo kontekstom, znotraj katerih je ime. Strežnik ob zahtevi za rezrešitev imena najprej pregleda svoj prostor. Če v svojem prostoru ne najde iskanega imena, mora uporabiti dodatne informacije, ki so v imenskem strežniku shranjene v obliki *kontekstnih povezav* . Kontekstne povezave združujejo razvrščevalne pogoje z imeni kontekstov in lokacijami imenskih strežnikov, ki so pooblaščeni za kontekst. Strežnik tako poišče pooblaščenega strežnika za dani kontekst in mu preda zahtevek. Postopek se nadaljuje, dokler ne dobi zahtevka strežnik, ki ga lahko dokončno razreši, torej vrne iskane atribute imenovanega objekta.

Ob začetku razreševanja je treba podati *začetni kontekst*. Začetni kontekst mora vsebovati vsaj kontekstne povezave za vse objekte v sistemu. V primeru, če začetni kontekst vsebuje kar vse atribute vseh objektov, torej se vsa imena razrešijo v enem koraku, je tak kontekst *globalen*. Globalna so torej tudi vsa imena. V nasprotnem primeru pa imenujemo imena *relativna* glede na začetni kontekst.

Globalna imena so praktično uporabna samo za manjše sisteme s centraliziranim upravljanjem imen. Tipično začetni konteksti vsebujejo samo kontekstne povezave. Kontekstne povezave tvorijo drevo. Samo listi tega drevesa, imenovanega *drevo za razreševanje*, vsebujejo atribute o objektih.

Upravljanje z imeni pomeni vzdrževanje informacij o imenovanih objektih. Te informacije imenujemo atributi, ki niso in ne smejo biti vključene v imenu.

Kot smo že omenili, drevo za razreševanje ni nujno podobno drevesu imenskega prostora. Ravno ta relativna nepovezanost omogoča veliko prilagodljiost imenskih strežnikov spreminjajočim se zahtevam v porazdeljenih sistemih, ne da bi pri tem uporabniki čutili spremembe ali bi celo morali spreminjati imena objektov. Dodajanje novih objektov, ki bi lahko pokvarilo uravnoteženost drevesa, rešimo enostavno tako, da v kritično vozlišče dodamo nove pogoje za razvrščanje. Zelo težko je namreč predvideti, kako velik imenski prostor bo razvijajoč porazdeljen sistem potreboval v prihodnosti. Tudi tako imenovana ploščata imena, ki nimajo posebne strukture, lahko razdelimo na kontekste. Pri tem je odprto vprašanje ustreznih funkcij za razvrščevalne pogoje, ki bi iz na videz nestrukturirane množice poljubno izbranih imen poiskale neko strukturo za razdelitev imenskega prostora v kontekste.

6 UČINKOVITOST SISTEMA ZA UPRAVLJANE Z IMENI IN KVALITETA NJEGOVEGA SERVISA

Učinkovitost sistema za upravljane z imeni lahko merimo s ceno procesiranja in porabo pomnilniškega prostora, ki jo sistem povprečno porabi za razreševanje. Seveda je jasno, da je najbolj učinkovit imenski sistem, kjer bi uporabljali neposredne naslove objektov. Take so bile rešitve v prvih računalnikih. Čimvečji je sistem in čimbolj je kompleksen, večjo ceno so pripravljeni uporabniki plačati za storitve, ki jim pomagajo pri delu. To velja tudi za imenski sistem.

Pomemben faktor kvalitete servisa je število nivojev v imenski strukturi. Zahteve za obliko in vsebino imen za računalnik in človeka so ravno nasprotne. Ker so zahteve računalnika nespremenljive, se mora prilagoditi človek. Čimvečje je število nivojev, tem bolj se lahko imena prilagodijo človekovim zahtevam. Možno je tudi doseganje tranparentnosti imen. Tudi prilagodljivost sistema se na tak način poveča.

Prilagodljivost sistema za dinamične in dolgoročne spremembe pa povečuje tudi strukturna neodvisnost razreševalnega mehanizma od imenske stukture. Pri razreševanju se poveča učinkovitost, če minimiziramo potrebo po prenašanju informacij med procesnimi mesti. Sheltzer v [Shel86] predlaga uporabo *predpomnilnikov (cache)* za zmanjševanje prometa po komunikacijskih kanalih.

7 LITERATURA

[Come86] D.E. Comer, L.L. Peterson: A Model of Name Resolution in Distributed Systems, IEEE Proc. 6. ICDCS, May 1986

[Pete85] L.L. Peterson: Defining and Naming the Fundamental Objects in a Distributed Message System, Ph.D. Thesis, Purdue University, May 1985

[Pope85] G. Popek, B.J. Walker (edts.): LOCUS Distributed System Architecture, MIT Press, 1985

[Ruge88] J. Rugelj: Upravljanje porazdeljenih računalniških sistemov, Magistrsko delo, Fakulteta za elektrotehniko, Ljubljana 1988

[Salt78] J.H. Saltzer: Naming and Binding of Objects, LNCS 60, Springer-Verlag 1978

[Shel86] A.B. Sheltzer, R. Lindell, G.J. Popek: Name Service Locality and Cache Design in a Distributed Operating Systems, IEEE Proc.6th ICDCS, May 1986

[Shoc78] J.F. Shoch: Inter-network Naming, Adressing and Routing, COMPCON 78, April 1978

[Su83] Z.S. Su: Identification in Computer Networks, ACM Computer Comm. Rev. vol.13/10, 1983

[Terr86] D.B. Terry: Structure-free Name Management for Evolving Distributed Enviroments IEEE Proc. 6th ICDCS, May 1986

[Wats81] R.W. Watson: Identifiers in distributed systems, in Distributed systems, LCNS 105, Springer-Verlag, Berlin 1981

WHY INFORMATICA CANNOT BE COVERED BY THE SCI?

INFORMATICA 2/89

Keywords: science citation index, Informatica

Ivan Rozman, Maja Drev
Faculty of Engineering, Maribor

For all of those who may not be familiar with Science Citation Index (SCI) let us present it by giving its definition:

SCI is a large scale index to the international scientific and technical literature. The SCI indexes over 600,000 items per year in over 3,300 scientific and technical journals (and nearly 1,400 multiauthored books). One of the unique indexing techniques used is citation indexing, i.e. alphabetical listing by author of all the references found in footnotes and bibliographies of journals covered by the SCI. The SCI is made up of four separate but related indexes which cover the same newly published articles, but afford access to each article in different ways. These indexes are the Citation index, the Source Index, the Corporate Index and the Permuterm Subject Index. The SCI Journal Citation Reports (JCR), an additional section, provides bibliometric analysis of science journals in the ISI data base (Institute for Scientific Information, Philadelphia, USA) and is designed as a tool for evaluating these journals. The JCR extends the use of citation analysis to examine the relationship among journals rather than among articles and their authors. Like any other tool, the SCI JCR cannot be used indiscriminately. However, the ambition of the SCI was to create and cover a list of high impact science journals established through their citation records. Besides their citation records science journals found on this list must satisfy certain criteria regarding the contents, the format, the geographic origin, the language, the present coverage in the discipline and the accessability to the international scientific community.

If we look up the list of SCI source journals we find out that Informatica is not included. Why? In order to get an answer we submitted a sample issue of Informatica, vol. 12, no.2,1988, for evaluation and possible coverage by the SCI to a coverage specialist at the ISI. Here are the results, their interpretation and our reflections in abridged form:

Informatica has not been accepted for coverage by the SCI because the field it covers is already strongly represented with journals which have very high citation records. The coverage specialist at the ISI was kind enough to include some unofficial suggestions to increase the citation record of Informatica and to point out some of its deficiencies.

If we wish to have some sort of presence in the citation indexes it is important to have as unique a name as possible for a journal. There are four journals already in publication with the title Informatica in Ulrich s International Periodicals Directory. This may cause confusion when a journal is cited. It would be appropriate to change its name and call it e.g. Informatica Slovenica.

As regards the format, Informatica does not include a clearly stated journal frequency or author adresses which indicate both the author s institutional affiliation and country of origin, which faciliates faster communication within the scientific community. The journal's adherence to its prescribed publication schedule is also very important. References for an article should be grouped at the end of an article. The use of a standard form for complete references is encouraged. The ISO 690 or ANSI Z39:29

standards are suggested. The ISI Checklist for Journal Editors and Publishers recommends the following contents page design: columnar format, a list of titles first, followed by the authors names, and a list of page numbers at the right. In Informatica it's just the opposite. Detailed recommendations for journal format are included.

As regrads the accessability in the scientific community, there is no listing for Informatica in any library acquisition tool, not even in Ulrich's International Periodicals Directory! It is also disappointing to find no citations to it from the journals of the ISI databse, although Informatica is in volume 13. This can be explained in two ways: either the authors who publish in Informatica are not reknown enough to publish in science journals covered by the SCI or they avoid citing articles from Informatica when they publish in international science journals because they underestimate it.

The question is how to increase the international impact of Informatica and enable inclusion in the SCI source journals. Firstly, Informatica should be recommended to various indexing and abstracting services. Secondly, Informatica should gain some citation records. This depends mostly on the professional community that surrounds it. It would be advisable to recommend Informatica and some articles in it to colleagues within the country and elsewhere. This may help Informatica to be cited in journals of wider reknown. Additional attention should be paid to the format.

Sustained efforts in accordance with the suggested guidelines should upgrade Informatica and make it accessible in the international scientific community.

Attention to the Readers of Informatica

Readers of Informatica are kindly requested to send information on their papers recently published in foreign professional (scientific, also philosophical) journals (periodicals). Such information will be regularly published within this column.

Some Recently Published Papers in Foreign Professional Periodicals

Borka Jerman – Bažič, Will the Multi – octet Standard Character Set Code Solve the World Coding Problems for Information Interchange? **Computer Standards & Interfaces*** *8 (1988/89) 127 – 136.*

P. Kokol, B. Vukelič, M. Mernik, I. Rozman and V. Žumer, IMCL Language and Its Implementation on the Personal Computer. **SIGSMALL/PC Notes** *14 (1988) 4, 8 – 18 (ACM Press, New York).*

L. Sluga, P. Butala, Nada Lavrač and M. Gams, An Attempt to Implement Expert Systems Techniques in CAPP. **Robotics & Computer Integrated Manufacturing** *4 (1988) 1/2, 77 – 82 (Pergamon Press).*

A. P. Železnikar, Principles of Information. **Cybernetica**** *31 (1988) 99 – 122.*

A. P. Železnikar, Information Determinations I. **Cybernetica** *31 (1988) 181 – 213.*

* **Computer Standards & Interfaces** is published by Elsevier Science Publishers B.V. (North – Holland).

** **Cybernetica** is published by the International Association for Cybernetics, Palais des Expositions, Place Andre Rijckmans, B.5000 Namur, Belgium.

Informatica

Informatica *is published four times a year in Winter, Spring, Summer and Autumn by the Slovene Society Informatika, Iskra Delta Computers, Stegne 15C, 61000 Ljubljana, Yugoslavia.*

Volume 13 Number 2 April 1989 YU ISSN 0350-5596

Informatica

A Journal of Computing and Informatics

CONTENTS