Volume 14 Number 3 July 1990
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Informatica

A Journal of Computing and Informatics

Subscription Information

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

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

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

Printed by Tiskarna Kresija, Ljubljana.

Informacija za naročnike

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

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

Številka žiro računa: 50101 – 678 – 51841.

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

Tisk: Tiskarna Kresija, Ljubljana.

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

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

Volume 14 Number 3 July 1990
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Letnik 14 Številka 3 Julij 1990 YU ISSN 0350 – 5596

Informatica

Časopis za računalništvo in informatiko

V S E B I N A

Voice Driven Editor	*Z. Kačič* *B. Horvat* *I. Urlep* *B. Štok*	1
Understanding as Information	*A. P. Železnikar*	8
Design of Hardware with Programmable Gate Arrays	*P. Praprotnik* *B. Dugonik*	31
Transputerli v sistemih za delo v realnem času	*P. Kolbezen*	35
Simulacija in vrednotenje programske opreme za večprocesorski računalnik	*B. Čukić*	44
Računarski sistem "Sphinx" za prepoznavanje kontinualnog govora neodvisnih govornika sa velikim rečnikom	*M. M. Miletić*	54
Kriteriji za določanje sistemske programske opreme osebnih računalnikov *za krmiljenje industrijskih procesov*	*D. Mrdaković*	56
Razvoj in opis dvonivojskega modela morfološke analize in sinteze	*T. Erjavec*	59
O znanju v porazdeljenih sistemih	*Tatjama Kapus*	63
Bayesove umetne nevronske mreže	*I. Kononenko*	72
Novice in zanimivosti		87
Information and Postmodernism	*A. P. Železnikar*	89

VOICE DRIVEN EDITOR

INFORMATICA 3/90

Zdravko Kačič
Bogomir Horvat
Igor Urlep
Faculty of Technical Sciences and
Bojan Štok
Computer Center,
University of Maribor,
Smetanova 17, Maribor

Keywords: dialog system, text processing, speech recognition, hidden Markov models, finite automata.

A small dialog system, comprised in a text processing environment, is presented in this paper.
A vector quantization and hidden Markov models were used for word recognition a d finite automata for syntactic/semantic analysis.
With full realization of a pragmatic level, what becomes necessary in a real-time system, even better recognition accuracy could be achieved, as it is presented in this paper. Because of that, realization of a real-time system may be feasible.

V članku smo predstavili realizacijo sistema dialoga, vključenega v okolje urejevalnika besedila. Pri razpoznavanju besed smo uporabili vektorsko kvantizacijo in skrite modele Markova. Nivo sintaktične/semantične analize pa smo realizirali z množico končnih avtomatov.
Z popolno realizacijo nivoja pragmatike, kar postane neobhodno pri realizaciji sistema za delo v realnem času, je mogoče zagotoviti še večjo zanesljivost sistema, kot pa je zanesljivost, ki jo nakazujejo predstavljeni rezultati razpoznavanja. S tem so dane realne možnosti za razvoj sistema za delo v realnem času.

1. Introduction

Most of commands in man - machine communication are passed to a machine by pressing buttons or in some similarly way. This way of communication becomes a bottleneck everywhere we have to enter a large quantity of data.
Because of that, there are a lot of research projects which main goal is to make it possible to communicate with a machine in a most natural way for a man - with speech.
In 1952, Bell laboratories developed the first speaker-dependent speech recognizer, what is today believed to be the beginning of the automatic speech recognition era. The recognizer was able to recognize numbers from 0 to 9.
The next step forward was made in 1971, when the DARPA SUR project was launched. The main goal of the project was development of a 1000 word speech understanding system. Harpy and Hearsay II were the most successful systems developed in this project.
There are many commercial product available on the marketplace today, but a great deal of them are speaker-dependent.

Speech recognition systems can be divided into three groups regarding the methods of speech recognition used /1/:
a) pattern matching technique (DTW),
b) stohastic modelling (HMM), and
c) knowledge-based approach.
The most successful commercial systems are from groups a) and b).

In this paper a general architecture of a voice driven editor will be presented, but only realization of the speech recognizer will be discussed in more detail.

2. The architecture of a voice driven editor

The voice driven editor enables voice activation of command functions ordinarily accessible through keyboard or with mouse. It accepts editing commands in a form of spoken sentences (e.g., select current paragraph, save file, open new window, ...). With this, the editor should facilitate the text processing and first of all reduce the time required for finishing a text editing task.

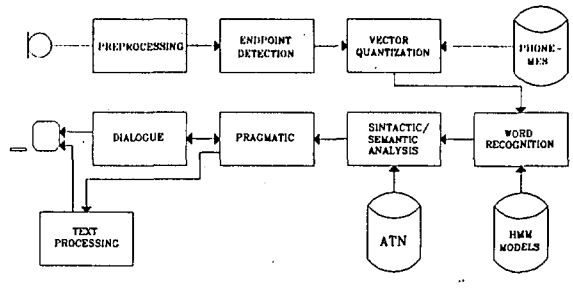


Figure 1: Voice driven editor

The voice driven editor consists of two main modules: a text processor and a multilevel speech recognition system. The speech recognizer is speaker-dependent isolated word recognition system, with 18 words vocabulary. The general scheme is presented in figure 1.
The communication between speech recognition system and word processor is carried out through a mailbox.
Many of the standard functions ordinarily included in text processors are built in the voice driven editor. For activation with voice, the most interesting among others are commands for selecting, formatting and sorting.
In our further discussion we will consider only the speech recognition system. It consists of four main modules:
a) a word recognizer,
b) a syntactic/semantic analyzer,
c) a pragmatic analyzer, and
d) a dialog module.

3. Word recognizer

The word recognizer attaches a word hypothesis, from available vocabulary, to appropriate input speech sequence.
The word recognizer consists of:
a) a preprocessing module,
b) a segmentation module,
c) an endpoint detection module,
d) a vector quantization module, and
e) of a word hypothesis detection module

3.1. Preprocessing

Typical spectral envelope of the voiced sound is illustrated on figure 2.

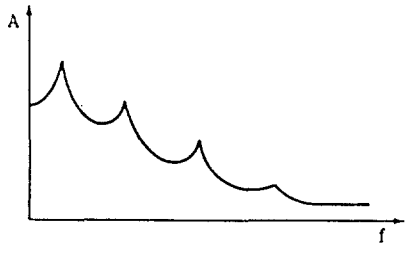


Figure 2: Typical spectral envelope of the voiced sound

We can notice how the spectrum rolls off in high frequencies. Because of that, the LPC model will satisfactorily approximate only the low frequencies. To avoid this, we pass the speech signal through a preemphasis filter with transfer function $1 - az^{-1}$. If s(n) is the input signal, then the preemphasized signal s'(n) is

$$s'(n)=s(n)-0.9375*s(n-1) \qquad (1)$$

3.2. Segmentation

In this phase, we segment the speech signal into small segments, called frames, which are assumed to be quasi-stationary.
Figure 3 shows an example of speech segmentation.

The segmentation is done by multiplying the speech signal, s'(n), by a window signal, w(n).

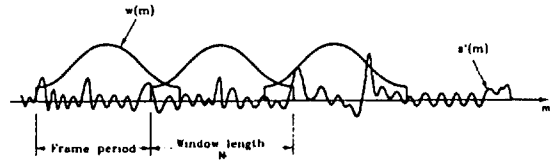


Figure 3: Segmentation of the speech signal

The Hamming window is given by

$$w(n)=\begin{cases}0.54-0.46\cos\frac{2\pi n}{N-1}, & n=0,1,\ \dots\ ,N-1\\ 0, & \text{otherwise}\end{cases} \qquad (2)$$

Here, N is the desired window length in samples.

3.3. Endpoint detection

The aim of the endpoint detection is to separate nonspeech sequences from speech sequences. In our experiment we used the endpoint detector which measures the speech amplitude function. The methods shown in Fig. 4 locates energy pulses, corresponding typically to syllables or words, by comparing energy in decibels against four thresholds k_1, k_2, k_3, k_4 /2/. When energy exceeds the lowest threshold k_1 (3 dB above background noise), time A_1 is noted. If it rises above k_2 (10 dB) before falling below k_1, time A_1 is assumed as start time (unless duration $A_2 - A_1$ exceeds 75 ms, where A_2 is then viewed as the start time and the signal preceding A_2 is considered as breath noise). The end time is determined using thresholds k2 and k_3 (A_4 is the end time unless $A_4 - A_3 > 75$ms). The values of the thresholds were determined by experiment.

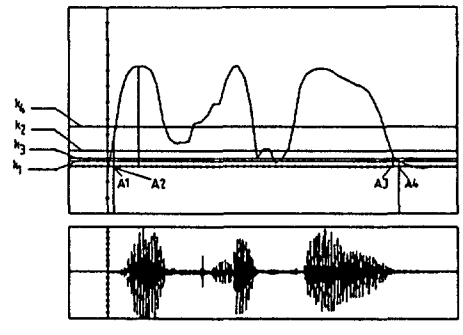


Figure 4: Endpoint detection

3.4. Vector quantization

A quantization means a mapping of an input vector x into a vector y_i from code alphabet $C=(y_1, y_2, \dots, y_i, \dots)$, which is known as a codebook. Appropriate code alphabet C is usually created through the minimum average quantization error E(d(x,y)) searching /3/.
In our experiment the LP method was chosen for generation of vectors and the Itakura distance as a metrics.

3.4.1. Linear prediction

Given a linear, discrete-time system, the object of linear prediction is to estimate the output sequence or, specifically, the forthcoming output sample from a linear combination of input samples, past output samples, or both:

$$\bar{y}(n)= \sum_{j=0}^{q} b_j x(n-j) - \sum_{i=0}^{p} a_i y(n-i) \qquad (3)$$

The factors a(i) and b(j) are called predictor coefficients.

3.4.2. Derivation of the linear prediction equations

Given a zero-mean signal y(n), in the all-pole predictor, we wish to estimate the value at point n as a linear combination of past values

$$y(n)=-\sum_{i=1}^{p} a_i y(n-i) \qquad (4)$$

Then the error is

$$e(n)=y(n)-\bar{y}(n)= \sum_{i=0}^{p} a_i y(n-i) \qquad (5)$$

where a(0) = 1.

To derive the predictor, we compute the coefficients a which will minimize this error in the mean-squared sense, i.e., that will minimize $E = \langle e^2 \rangle$. The coefficients will be found by the use of the orthogonality principle. This principle states that the desired coefficients are those which make the error orthogonal to the samples y(n-1), y(n-2),..., y(n-p).
The finally result of that procedure is a set of p equations, which can be represented as follows:

$$\begin{bmatrix} r_0 & r_1 & r_2 & \cdots & r_{p-1} \\ r_1 & r_0 & r_1 & \cdots & r_{p-2} \\ r_2 & r_1 & r_0 & \cdots & r_{p-3} \\ \cdots & & & & \\ r_{p-1} & r_{p-2} & r_{p-3} & \cdots & r_0 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_p \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \cdots \\ r_p \end{bmatrix} \qquad (6)$$

where

$$r_i = \sum_{n=1}^{N-i} y(n)-y(n-1) \ , \quad i=1,2, \ \ldots \ ,p \qquad (7)$$

The elements of the R matrix are autocorrelation coefficients.
In the R matrix all the elements along any diagonal are equal. Such a matrix is called Toeplitz matrix. Furthermore, R is not only Toeplitz but also symmetric.
The above mentioned properties of the R matrix permits usage of Levinson-Durbin recursive method, which is computational very efficient. The recursive solution steps are as follows:

1. n=0
   $E_0 = r_0$
   $a_i(0)=0, \ i=0,1, \ \ldots \ ,p$
2. n=1,2,...,p

$$k_n = \frac{R(n)- \sum_{i=1}^{n-1} a_{n-1}(i) \ r_{n-i}}{E_{n-1}}$$

$$a_n(n)=k_n$$
$$a_n(k)=a_{n-1}(k)-k_n a_{n-1}(n-k) \ , \ k=1,2,\ldots,m-1$$
$$E_n = (1-k_n^2) \ E_{n-1}$$

The result is a vector of LPC coefficients

a = (1, -ap(1), -ap(2),..., -ap(p))

3.4.3. Itakura distance

Let

$$a_l=[1 \ a_1 \ a_2 \ \cdots \ a_p]^T$$

$$R_1=\begin{bmatrix} r_0 & r_1 & \cdots & r_{p-1} \\ r_1 & r_0 & \cdots & r_{p-2} \\ \cdots & & & \\ r_{p-1} & r_{p-2} & \cdots & r_1 \end{bmatrix}$$

be the coefficient vector and the autocorrelation matrix for the LPC vocal tract model H_1. Let a_2, R_2 be the corresponding quantities for H_2. The likelihood ratio can be defined either as

$$d_{LR1} = \frac{a_2^T \ R_1 \ a_2}{a_1^T \ R_1 \ a_1} \qquad (10)$$

or as

$$d_{LR2} = \frac{a_1^T \ R_2 \ a_1}{a_2^T \ R_2 \ a_2} \qquad (11)$$

The log-likelihood ratios are the logarithms of these expressions:

$$d_{LLR}=10 \ \log_{10}(d_{LR}) \qquad (12)$$

The metrics of (11) and (12) are nonsymmetric. If we want a symmetric measure, we can use the quantity

$$d_{LRS} = \frac{d_{LR1} + d_{LR2}}{2} - 1 \qquad (13)$$

which has the desired property of symmetry.

The expression $a_1^T R_1 a_1$ is the energy of the residual signal after LPC modeling. The filter coefficients a_j are computed from the autocorrelation coefficients r(j) in such a way as to minimize this energy. Hence, if we replace a_1 with any other nonzero vector of coefficients, this expression increases /4/.

3.5. Hidden Markov model

The hidden Markov model can be described in a following manner:
The hidden Markov model is a set of N distinct states. At regularly spaced discrete times, the system undergoes a change of state

according to a set of probabilities associated with state. That process could be called an observable Markov model since the output of the process is the set of states at each instant of time, where each state corresponds to an observable event. But, if we extend this model in such a way that the observation is a

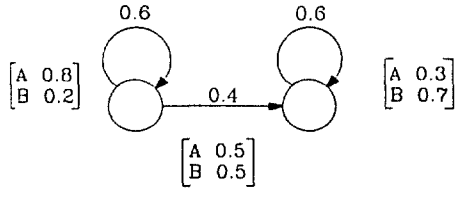


Figure 5: Two state HMM with symbols A and B

probabilistic function of the state, the above process become hidden and this new model we call HMM /6/. Figure 5 shows an example of two state HMM with symbols A and B.

3.5.1. Elements of the HMM

The HMM can be described by the following:

1) N, the number of states in the model. We denote the individual states as $S = \{S_1, S_2, \ldots, S_N\}$, and the state at time t as q_t.

2) M, the number of distinct observation symbols per state. The observation symbols correspond to the physical output of the system being modeled. We denote the individual symbols as $V = \{v_1, v_2, \ldots, v_M\}$.

3) The state transition probability distribution $A = \{a_{ij}\}$ where
$a_{ij} = P[q_{t+1} = S_j \backslash q_t = S_i]$, $1 \leq i,j \leq N$.

4) The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where
$b_j(k) = P[v_k \text{ at } t \backslash q_t = S_j]$, $1 \leq j \leq N$
$1 \leq k \leq M$.

5) The initial state distribution $\pi = \{\pi_i\}$ where
$\pi_i = P[q_1 = S_i]$, $1 \leq i \leq N$.

For a given appropriate values of N, M, A, B, and π, we can use the HMM as a generator to give an observation sequence $O = O_1 O_2 \ldots O_T$. To indicate the complete parameter set of the model we use the compact notation $o = (A,B,\pi)$ /6/.

3.5.2. Types of HMM

In general, HMM is an ergodic model (fig. 6a). It means that any state can be reached from any other state. For such a model full state transition matrix exist /5/.
For some applications, nonergodic models are more appropriate. Figures 6b and 6c show two examples of nonergodic model. The first one is left-right model and the second one is parallel left-right model. The process modeled by LR model starts in state 1, continuous trough other states in monotony ascending order and finishes in state N.

In our experiment we have used the left-right model.

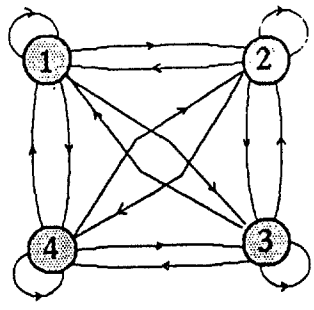


Figure 6a: An ergodic HMM

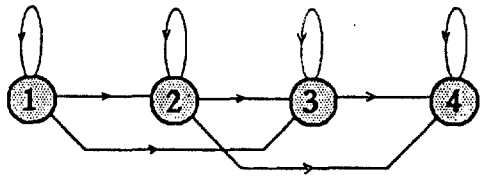


Figure 6b: A left-right HMM

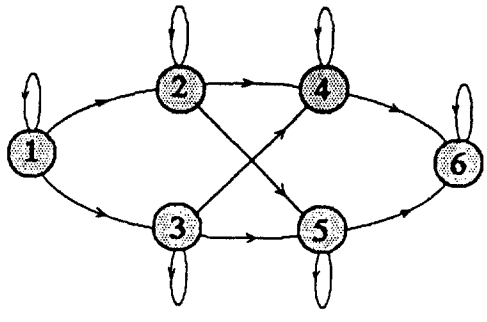


Figure 6c: A parallel left-right HMM

3.5.3. Choice of basic phonetic unit

In HMM models definition basic phonetic unit have to be chosen. This unit have to be good defined and to have ability of learning. The basic unit usually used is a word or a phoneme. The word based models are good for recognition of small word vocabularies but for a large vocabulary too much training date are required. The phoneme based models are very simple for training but their successfulness depend on phonemes left and right from a current word.
In the voice driven editor a word has been used as a basic phonetic unit.

4. The level of syntactic and semantic analysis

The syntactic/semantic analyzer was realized through a set of finite automata. Each of these automaton has a start state and one or more finite states. The number (characterizing an action) has been assigned to each automaton. The word sequence which is generated by the word recognizer is checked by each of the automaton. If after this, the automaton is in finite state, the command is regular and an action number is generated and transferred to the pragmatic level. If no automaton is in finite state, the error is mediated to the dialog level.

4.1. The finite automaton

The finite automaton (FA) is a 5-tuple (Q,Σ,δ,q_o,F) where Q is a finite set of states, Σ is a finite input alphabet, δ is a total function $Q\times\Sigma \rightarrow Q$ (transition function), q_o is the initial state from Q, and F is a set of final states.
To describe the behavior of an FA on a string, we must extend the transition function δ to apply to a state and a string rather than a state and a symbol.

Let

$\delta = Q\times\Sigma \rightarrow Q$.

Then

$$\delta' = Q\times\Sigma^* \rightarrow Q$$

can be defined as follows

1. $\forall\ q\in Q \qquad \delta(q,\varepsilon)=q$
2. $\forall\ q\in Q,\ w\in\Sigma^*,\ a\in\Sigma \qquad \delta'(q,wa)=\delta(\delta'(q,w),a))$

The formula

$$\delta(q,w)'=r$$

means:

<from state q, state r can be reached with input word w>.

A language of finite automata M is a set of words which bring us from initial state q_o to some final state. It can be written as

$$L(M) = < w|\delta'(q_o,w)\in F>.$$

To each word in our system one symbol which is element of Σ has been assigned and for each action a finite automaton has been defined. The automaton is presented in following form:

s_1 ns_{11}/sim_{11} ns_{12}/sim_{12} ... $ns_{1M}(1)/sim_{1M}(1)$
s_2 ns_{21}/sim_{21} ns_{22}/sim_{22} ... $ns_{2M}(2)/sim_{2M}(2)$

s_N ns_{N1}/sim_{N1} ns_{N2}/sim_{N2} ... $ns_{NM}(N)/sim_{NM}(N)$

Each row describes the transition from state s_i to state ns_{ij} at symbol sim_{ij}.

Figure 7b shows an example of automaton for command "ODPRI OKNO", and figure 7a shows the same automaton described with structure used in our explanation.

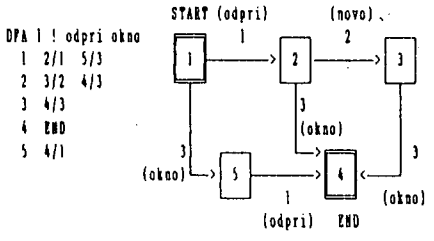


Figure 7a Figure 7b

4.2. Pragmatic level

At pragmatic level the system is trying to execute the recognized command. If this can not be done, the message about conflict situation have to be mediated to the dialog level. Let's show some meaningless commands in the VDE system:
- a window which was not opened can't be closed,
- more than 10 windows can't be opened ,
- a text which was not previously selected, can't be deleted
- a page which doesn't exist can't be reached.

4.3. Dialog level

The task of this level is to try to solve conflicts which were found at the syntactic/semantic or pragmatic level. This have to be done throughout the dialog with the user. The messages are mediated to the user by means of acoustic signal or with reports displayed on the screen.
Let's consider some examples of the dialog with the VDE:

U: next window
C: conflict - only one window is open

U: open the window of
C: incomplete command - advice: open next window

U: open next window
C: window is opened

U: select
C: entire text selected

U: mark
C: incomplete command - advice: mark all

U: open new window
C: conflict - all windows are already opened - advice: close the window you don't need

/ U - user C - computer /

5. Experimental results

The speech recognition system simulation has been realized on the VAX 8800 computer and the code has been written in VAX Basic and VAX Assembler.
The speech signal was filtered by band pass filter with cutoff frequencies at 300 Hz and 3400 Hz, and sampled by 10 KHz frequency at 12-bit resolution. The samples were windowed with 32 ms long Hamming window function and with 16 ms frame period. Features of the sampled speech were described with 12^{th} order autocorelation based linear prediction.
A set of phoneme utterances and a set of word models has been created in the training phase.

5.1. Creation of the reference vectors for vector quantization

The 36 reference vectors has been created. From different utterances individual phonemes has been labelled manually and divided into 36 classes. For each class, the vector with minimum distance from all vectors within the class has been taken as a class reference vector. The list of phoneme used in vector quantization is shown in table 1.

An example of vector quantization for the word "ODPRI":

d1 (4) o1 (12) o2 (12) o1 (12) o1 (12) e2 (5) a1 (1) n (11) d1 (4) p1 (13) d1 (4) d2 (4) d2 (4) p2 (14) p1(13) p1 (13) p2 (14) b (2) b (2)

Table 1: The list of phoneme used in vector quantization

Phoneme	Number of vectors	HMM symbols
a1	14	1
a2	56	1
a3	13	1
b	16	2
cc1	23	3
cc2	17	3
cc3	12	3
d1	13	4
d2	14	4
d3	7	4
e1	4	5
e2	27	5
e3	10	5
h	6	6
i1	14	7
i2	51	7
i3	30	7
j	11	8
k2	15	9
k3	8	9
l	13	10
n	42	11
o1	14	12
o2	34	12
o3	16	12
p1	45	13
p2	17	14
p3	8	14
r	18	15
s	47	16
ss	50	17
t2	19	18
t3	6	18
u	6	19
z	26	20
zz	21	21

b (2) b (2) t2 (18) d3 (4) i1 (7) i1 (7) i1 (7) i1 (7) i1 (7) i1 (7) i1 (7) i1 (7) i1 (7) e1 (5) p2 (14) n (11) n (11) n (11) n (11).

5.2. A HMM word model

Each HMM model has 5 states (N=5). The number of a possible symbols in our system was 21 (M=21). For each model $\delta=(A,B,\pi)$ the following values has been defined

$$a_{ij} = 1/N \ , \ i,j = 1,2....,N$$
$$b_{jk} = 1/M \ , \quad j = 1,2....,N$$
$$k = 1,2....,M$$

$$\pi_i = 1/N, \quad i = 1,2....,N$$

The training procedure for one word model was as follows.
For each word a sequence of symbols has been generated. The model's parameters has been reestimated by iterative Baum-Welch method /6/. The models for the following words have been defined:

ODPRI, NOVO, OKNO, ZAPRI, OZNAČI, STRAN, VSE, ZBRIŠI, TEKST, VSTAVI, SHRANI, NASLEDNJE, SKOČI, NA, POIŠČI, KALKULATOR, KONČAJ, NALOŽI.

5.3. Recognition results

A created set of utterances has had 7 utterances for each word. The 5 utterances has been used in the training phase and all utterances from the available set has been used for the recognition process.
After word models initialization, 75 iteration has been performed on each model. Average word recognition accuracy was 80.95%. More detailed results are shown in table 2.

Table 2: Word recognition results

word	recognized [%]	misrecognized as
kalkulator	100.00	
končaj	71.43	odpri, označi
na	57.14	kalkulator, skoči
naloži	100.00	
naslednje	100.00	
novo	71.43	okno
odpri	100.00	
okno	85.71	novo
označi	57.14	naloži, skoči
poišči	100.00	
shrani	85.71	stran
skoči	100.00	
stran	100.00	
tekst	100.00	odpri
vse	57.14	stran,naslednje,vstavi
vstavi	42.86	shrani, zapri
zapri	71.43	odpri, označi
zbriši	57.14	skoči, poišči

By attaching the syntactic/semantic analysis, early discovering of bad recognized words and the semantic irregularity becomes possible. Considering this, we tried to recognize the group of words (sentence) which correspond to particular command. For each command seven different utterances has been generated. The recognition results are given in table 3. In first column are the recognized commands, the second shows the recognition rate and the third detected error rate.

Table 3: Experimental results of commands recognition in per cent

command (sentence)	recogn.	detect. errors
odpri novo okno	57.14	100.00
odpri okno	85.71	100.00
zapri novo okno	28.57	85.71
zapri okno	57.14	85.71
skoči na naslednje okno	57.14	100.00
skoči na stran	57.14	100.00
označi stran	57.14	71.43

Because of the low recognition rate, the recognition module has been upgraded in such a way that the best two word hypothesis has been mediated from recognition module to syntactic / semantic module.

Table 4: Results of commands recognition with the best two hypothesis in per cent

command (sentence)	recogn.	detect. errors
odpri novo okno	100.00	-
odpri okno	100.00	-
zapri novo okno	71.43	50.00
zapri okno	71.43	50.00
skoči na naslednje okno	100.00	-
skoči na stran	100.00	-
označi stran	100.00	-

Now, the syntactic/semantic module tries to recognize a command with the first word hypothesis. If the recognized command was meaningless, the whole procedure was repeated with the combination of the first and the

second hypothesis. In this way, the results has been improved as we can see from table 4.

With full realization of the pragmatic level (which is necessary for the real-word application) the recognition rate is believed to be even more improved.

5.4. The time complexity

Because of the algorithm's complexity, it is very difficult to analytical compute its time complexity. But to get at least a rough picture of the time complexity of particular module, we will consider the CPU time which is required to handle the task. The most time consuming part was the vector quantization. The 1.66 sec (on DEC VAX 8800) was needed to compute the linear prediction coefficients for one window and to assign a symbol from the codebook. If an average word length is 35 windows, then 58.1 sec of CPU time is required for vector quantization for one word. For endpoint detection, 0,09 sec was needed. The word recognizer have to compare input symbols with each word model. To compute a probability for one word' model, 0.1 sec was required and for all models (18 words in a vocabulary) the CPU time needed was about 1.8 sec. Therefore for one recognized command, containing four words, approximately 240 sec was needed.

5.5. The space complexity

The word recognizer is the biggest time consumer and because of that only the space complexity for this part of the VDE will be considered here.
The HMM word model consist of a matrix A, B and of a vector π, which elements were represented in the computer with the highest accuracy (8 byte). The size of the matrix A was 5x5, the size of the matrix B was 5x21, and the size of the vector π was 1x5, which makes 1080 bytes for one word model. As we can see, for 1000 words vocabulary we would need 1080 Kbytes for all models .

5.6. The criticism of the real-time system

According to the time and space complexity, for implementation of the real-time system, we think that the following hardware would suffice:
- A/D converter with 10 KHz sampling rate at 12-bit resolution,
- signal processor TMS 320C25 for vector quantization
- signal processor TMS 320C25 for computing the HMM probability.
- INTEL 80286 or INTEL 80386 as a host processor for the realization of higher processing levels.
- at least 2 Mbytes of DRAM,
- a hard disk with at least 5 Mbytes of memory for application.

5.7. The expansion ability

The advantages of the system based on the HMM is that if we want to have bigger vocabulary, the needs for space and processor power do not grow so fast as they do in the other similar systems. The fundamental unit in VDE system is word, but for larger vocabulary more appropriate fundamental unit would be phoneme. We believe that the hardware suggested in the previous chapter suffice also for a few hundred words vocabulary /5/.

6. Conclusion

The recognition results presented in this paper indicates that used methods may be considered as appropriate for the realization of the voice driven editor. Therefore, our very next step will be to build VDE, with approximately 200 words vocabulary, which can be run on suggested hardware as a real-time application.

References

1. Zdravko Kačič, "On the use of array of active perceptors in a speech recognition system", Master thesis, Maribor, April 1989 (in Slovene).

2.Douglas O'Shaughnessy, Speech Communication, Human and Machine, Reading, Massachusetts: Addison-Wesley Publishing Company, 1987.

3. D. Wolf, H. Reininger, "Recent Advances in Speech Coding", in H. Niemann ed., Recent Advances in Speech Understanding and Dialog Systems, Berlin Heidelberg: Springer-Verlag, 1988, pp. 183-205.

4. Panos E. Papamichalis, Practical Approaches to Speech Coding, Englewood Cliffs: Prentice-Hall International, 1987.

5. Michael A. Pichney, Subrata K. Das, "A real-time IBM PC based large-vocabulary isolated-word speech recognizer", Proceed of Voice processing, London, 1986, pp. 173-182

6. L.R. Rabiner, "Mathematical Foundations of Hidden Markov Models", in H. Nieman ed., Recent Advances in Speech Understanding and Dialog Systems, vol.46, Berlin: Springer Verlag, 1988, pp. 183-206.

Keywords: blindness, breakdown, disorder, electronic dictionary, expert system, formalization, hermeneutics, information, intelligence, language, loseableness, misunderstanding, meaning, philosophy, semiotics, theory, understanding

Anton P. Železnikar
Volaričeva ulica 8
61111 Ljubljana, Yugoslavia

Understanding can be comprehended as a regular informational process occurring in various informational realms—in the living and the artificial—. Understanding as information concerns a comprehensive field of domains which are philosophic, linguistic, physiologic, psychologic, cognitive, and informational. The informational domain can embrace all other domains as their notional substance, putting them together in a complex and mutually dependent informational system of understanding. Part One of the article deals with general philosophy of understanding as information. Within this context it treats the meaning of the noun »understanding« and the verb »to understand« in Latin, Classical Greek, English, German, and Slovene, further, understanding as a philosophic realm concerning meaning, hermeneutics and semiotics, the so-called losableness, blindness and breakdown of understanding, intelligence as understanding and vice versa, the connectedness of principles of understanding and principles of information, possibilities of formalization of understanding, of an informational expert system and informational electronic dictionary, misunderstanding and disorder of understanding as information, all in the context of language, science and culture.

Razumevanje kot informacija

Razumevanje je mogoče pojmovati kot regularni informacijski proces, ki se pojavlja v različnih informacijskih domenah—živih in umetnih—. Razumevanje kot informacija zadeva obsežno polje domen, ki so filozofska, jezikovna, fiziološka, psihološka, spoznavna in informacijska. Informacijska domena zmore zaobseči vse druge domene kot njihova pojmovna podstat, jih povezati v kompleksni in medsebojno odvisni informacijski sistem razumevanja. Prvi del članka obravnava splošno filozofijo razumevanja kot informacijo in v tej povezavi pomen samostalnika »razumevanje« in glagola »razumeti« v latinščini, klasični grščini, angleščini, nemščini in slovenščini, nadalje razumevanje kot filozofsko domeno, ki zadeva pomen, hermenevtiko in semiotiko, tako imenovano zgubljenost, slepoto in prelom razumevanja, inteligenco kot razumevanje in obratno, povezanost principov razumevanja s principi informacije, možnosti formalizacije razumevanja, informacijskega ekspertnega sistema in informacijskega elektronskega slovarja, nerazumevanje in motnje (bolezni) razumevanja kot informacije, vse to pa v kontekstu jezika, znanosti in kulture.

Part One

A General Philosophy and Theory of Understanding as Information

1. Introduction

... Being-with is such that the disclosedness of the Dasein-with of Others belongs to it; this means that because Dasein's Being is Being-with, its understanding of Being already implies the understanding of Others. This understanding, like any understanding, is not an acquaintance derived from knowledge about them, but a primordially existential kind of Being, which, more than anything else, makes such knowledge and acquaintance possible. ...

M. Heidegger (BT) 160-161

Understanding does not concern merely language and linguistically comprehended concepts and principles as it is usually thought; it concerns information and informing of information in general, irrespective of informational levels in question, from the most primitive informing to the most perplexed one. For instance, in the domain of expert systems, understanding concerns language, learning, reasoning, and problem solving.

Understanding can be understood as a characteristically particular, informationally structured and organized process in the realm of information. Understanding concerns a thing's or a being's reaction (information, counter-information, and information of embedding) as the consequence of impacts (influences) of the internal and the external world. In this way, understanding can be comprehended in a purely informational way, that is, as information caused by internal and external informing and as the use of such information for some general, special, reasonable, intelligent, or any other purposes.

Within the realm of informational philosophy and informational theory, understanding can always be philosophically perceived and theoretically handled as a regular informational process (entity, phenomenon, operand, informational formula) on different hierarchical levels of informational arising. Any arising information can be conceived as a structure possessing the faculty of understanding (informational observing, investigation, counter-informing, informational embedding). Under these circumstances, understanding is becoming a kind of autonomous informational cycle informing within the general informational cycle of informational arising.

At the beginning of this essay it is possible to put a series of questions which can be used as origins of the informational study of the philosophy and the theory of understanding. These questions can be the initiators of distinct and substantial formulas which will be informationally developed through the discourse of this essay. Thus, let us start with a series of obvious and commonsense questions, tracing the path into the realm of informational understanding of understanding.

What is understanding as information? What are informational principles of understanding? How does understanding inform and how is it informed? What is the meaning of the term »understanding« in different natural languages? What is understanding as a philosophical realm which concerns meaning, hermeneutics (interpretation), semantics, interpretation of time as the possible horizon for any understanding of information, etc.? How is understanding lost in itself informationally? How can the losableness of understanding be broken down to clarify the blindness grounding in the losableness of understanding? What are the informational levels of understanding, from its basic informational level to its most complex functional form? How is information, in principle, always a process of understanding? What is the role of understanding in an intelligent system and intelligent environment? How does understanding impact intelligence and its genuine processes of intelligent informing and how does intelligence as information impact understanding as information? Which are the formal-theoretical possibilities for informational axiomatization and theory of principles and problems of understanding?

On the other side, questions of understanding can take also explicitly practical forms. For instance: How could understanding as a natural principle be introduced (as an explicit informational process) into expert systems? How could understanding as information be applied at the design of an electronic dictionary? Which are the basic components of understanding as an informational

process over a certain informational domain (of knowledge, experience, expertise, etc.)? What does understanding as information mean within the domain of a natural language, science, culture?

2. What is Understanding as Information in General?

... State-of-mind is one of the existential structure in which the Being of »there« maintains itself. Equiprimordial with it in constituting the Being is understanding. A state-of-mind always has its understanding, even if it keeps it suppressed. Understanding always has its mood. If we Interpret understanding as a fundamental existentiale, this indicates that this phenomenon is conceived as a basic mode of Dasein's Being. On the other hand, »understanding« in the sense of one possible kind of cognizing among others (as distinguished, for instance, from »explaining«), must, like explaining, be Interpreted as an existential derivative of that primary understanding which is one of the constituents of the Being of the »there« in general.
...

M. Heidegger (BT) 182

Let us give some initial general answers to the question of this paragraph, considering the following basic questions: What are informational principles of understanding? How does understanding inform by itself and how is it informed by other information and by itself? What are the informational levels of understanding, from its basic informational level to its most complex functional form? How is information, in principle, always a process of understanding? Within these questions we should not discuss the possibilities of informational formalization of understanding. Formalization processes of understanding will be exposed in the second part of this essay dealing with the formal, axiomatic, logical, and algebraic informational theory of understanding.

As far as that is concerned (POI), principles of understanding are not waiting to be discovered, they have to be constructed on the basis of our everyday experience in the domain of language, thought, comprehension, imagination, etc.

What are informational principles of understanding? First, this question has to be answered on a general level. Understanding as information performs as regular information. It arises informationally in the domain of some broader informational context (perplexity of information) and informs spontaneously and in a circular way, i.e., through informing, counter-informing, and embedding of the arisen understanding into the source information (source understanding). Thus, spontaneity and cyclicity of understanding as information can be investigated in a general way, however, this generality can be decomposed, for instance, on the level of an informational formula, stepwise in greater and greater details, developing and particularizing the initial formula and in this way constructing a concrete scenario of understanding. How does understanding inform and how is it informed? Understanding as information informs itself and other information with which it is informationally perplexed by its informing and counter-informing, and by its embedding of the produced counter-information and other information. In this way, understanding as information arises through informing of itself and other information, by which it is informed, i.e., informationally impacted. Understanding is open to be informed by the perplexed information and it can inform openly other information.

What are the informational levels of understanding, from its basic informational level to its most complex functional form? Understanding as information can be very simple and very complex informational process. The fundamental level of understanding is observable already on the level of basic informational principles (POI). Through counter-informing of information new information is coming into existence and the embedding of this information represents an act of understanding of the arisen information by the original information. Through informing, counter-informing and embedding, the understanding of a given information arises and becomes more and more complex as a part of the developing information.

How is information, in principle, always a process of understanding? Through informational arising which is a phenomenon of informational observation, investigation, construction and reaction, an informational process of understanding always exists, from the basic level of informing of information to its most complex, intelligent cases. Understanding as information is a consequence of informational complexity within living and non-

living world.

3. What is the Meaning of the Noun »Understanding« and the Verb »to understand« in Different Natural Languages?

... In the projecting of the understanding, entities are disclosed in their possibility. The character of possibility corresponds, on each occasion, with the kind of Being of the entity which is understood. Entities within-the-world generally are projected upon the world—that is, upon a whole of significance, to whose reference-relations concern, as Being-in-the-world, has been tied up in advance. When entities within-the-world are discovered along with the Being of Dasein—that is, when they have come to be understood—we say that they have meaning [Sinn]. ...

M. Heidegger (BT) 192

The meaning of the term »understanding« belongs to the most abstract notions of language and human mind. In fact, understanding means that something could be understood by somebody in such way or another. In many cases, understanding means to understand something in the same or similar way as it is understood by somebody else or to understand it in the way as it was understood in some prior time (for instance, the so-called problem of hermeneutics). Let us examine the meaning of the term »understanding« in Latin, Greek, German, English, and Slovene!

3.1. The Latin meaning of the noun »understanding« and the verb »to understand«

But that which is understood, taken strictly, is not the meaning but the entity, or alternatively, Being. Meaning is what wherein the intelligibility [Verständlichkeit] of something maintains itself. That which can be Articulated in a disclosure by which we understand, we call »meaning«.

M. Heidegger (BT) 192-193

In Latin, the noun **understanding** means *mens* (f), *ingenium* (n).

The Latin noun **mens** (-tis, f) means in general *mind, intellect, understanding;* and in particular: **a)** *judgment, discernment;* **b)** *disposition, feeling, character, heart, soul; spirit, courage, boldness, passion, impulse;* **c)** *thought, idea, opinion;* **d)** *plan, purpose.*

The Latin noun **ingenium** (-i, n) means in general *innate or natural quality;* and in particular: **a)** *disposition, temper, character; natural bent;* **b)** *natural capacity, mental power, talents, abilities, parts, genius, fancy;* **c)** *man of genius;* **d)** *clever thought.*

In Latin, the verb **to understand** means *intellego, comprehendo; teneo; accipio; percipio; certior fio; audio.*

The Latin verb **intellego** (-exi, -ctum) means in general *to perceive;* and in particular: **a)** *to understand, comprehend, have skill in, be master of;* **b)** *to judge, think, be of opinion;* **c)** *to be a connoisseur.*

The Latin verb **comprehendo** means

1. a) *to comprise, express, describe;* **b)** *to embrace with kindness;*

2. a) *to arrest, apprehend, capture;* **b)** *to detect; discover;* **c)** *to perceive, comprehend.*

The Latin verb **teneo** (tenui, tentum) means in general *to hold, keep, grasp, hold fast;* and in particular:

1. *to have in the hand, mouth,* etc., *to keep, seize, comprehend;* **a)** *to hold in mind, understand, conceive, comprehend, know;* **b)** *to hold one's way or course; to follow up, sail, steer, reach a place, land on, arrive at;* **c)** *to gain, acquire, obtain, attain;* **d)** *to convict, catch, detect;*

2. *to hold in one's possession, have, possess, be master of;* **a)** *to inhabit, occupy, garrison;* **b)** *to comprise, comprehend; to include;* **c)** *to possess, master; to hold fast, keep in, hold back, bind, fetter, control;* **d)** *to preserve, maintain, retain, guard; to remember;*

3. a) *to keep to, not to swerve from, remain true to;* **b)** *to last, endure, keep on;* **c)** *to charm, amuse, inspire;* **d)** *to bind, oblige; to maintain one's right;*

4. a) *to hold back, repress, check, restrain, suppress;* **b)** *to detain, delay;* **c)** *to keep a thing away from.*

The Latin verb **accipio** (-cepi, -ceptum) means

1. *to take, receive, get, accept;* the Latin noun **acceptum** (-i, n) means *what is received* (money); **a)** *to enter to the credit, to give credit for;* **b)** *to suffer, bear;* **c)** *to admit, approve of;*

2. *to receive as a guest;* **a)** *to entertain;* **b)** *to treat, deal with;*

3. *to bear, perceive;* **a)** *to learn, understand;* **b)** *to feel;* **c)***to take, interpret, explain;*

4. a) *to obtain;* **b)** *to suffer.*

The Latin verb **percipio** (-cepi, -ceptum) means in general *to seize, lay hold of, catch;* and in particular: **a)** *to take to oneself, assume;* **b)** *to get, receive, gather, harvest;* **c)** *to perceive, be sensible of, hear feel;* **d)** *to learn, know, comprehend, understand;* —the Latin noun **perceptum** (-i, n) means *principle, rule.*

The Latin verbal form **certior fio** seems to be of particular value for discussion concerning understanding as informing (of information). The Latin verbal form **certus** means in general *decided;* and in particular

1. a) *determined, resolved;* **b)** *resolved upon;*

2. a) *settled, established, fixed;* **b)** *sure, true, to be dependent on;* **c)** *informed, assured,* aliquam **certiorem facere** means *to inform* or *apprise one;* **d)** *definite, positive, undoubted.*

The Latin verb **fio** (factus sum, fieri) means

1. *to grow, spring up; a) to be born or created;* **b)** *to happen, come to pass;* **fieri potest, ut** means *it is possible that;* **fieri non potest quin** means *it is indispensable, inevitable that;*

2. passive of **facere: a)** *to be made;* **b)** *to be turned into, appointed;* **c)** *to be estimated.*

The Latin verb **audio** means **1.** *to hear, to have the faculty of hearing;* **2.** *to perceive, understand, learn.*

This short analysis of the meaning of the verb »to understand« in Latin shows the tremendous semantic complexity, perplexity, and semantic circularity of verbs concerning meaningfully the verb »to understand«. We see how the begun analysis could be continued by searching of meanings of already derived words. On the basis of this analysis, it is possible to understand how the verb »to understand« was used in different historical epochs by different philosophers.

3.2. The Greek meaning of the verb »to understand« and some derivatives and other words concerning this meaning

... The concept of meaning embraces the formal existential framework of what necessarily belongs to that which an understanding interpretation Articulates. Meaning is the »upon-which« of a projection in terms of which something becomes intelligible as something; it gets its structure from a fore-having, a fore-sight, and a fore-conception.
...

M. Heidegger (BT) 193

The meaning of the verb »to understand« in Classical Greek can be grasped, for instance, from the Aristotle's *Metaphysics* (MP) and compared, in the first step of investigation, by meanings of some more or less adequate Latin notions (words). For words in Greek, the direct transcription of Greek letters without accentuation and pronunciation marks will be used.

The Greek verb **noein** corresponds to the Latin verb *intelligo*, so it is possible to catch the meaning for this word in the previous paragraph. In Greek, the meaning of **noein** is *to think, know, understand.*

The Greek noun **noys** (in Latin, *intellectus*) means *mind.* Further, the Greek **noetos** (in Latin, *intellectualis, intelligibilis*) means *intelligible, conceived* and the Greek **noesis** (in Latin, *intelligentia*) means *thought, conception, denotation, knowing.* The Greek **ennoema** (in Latin, *conceptio*) means *item of information* and the Greek **dianoia** (in Latin, *mens, intellectus, meditatio*) means *intellect, intention, meaning, judgement, thinking, thought.*

The Greek **phronesis** (in Latin *prudentia*) means *intelligence,* while the Greek **phronimos** (in Latin *prudens*) means *intelligent.* The Greek **phantasia** (in Latin *phantasia, imaginatio*) means *appearance,* while the Greek **phantasma** (in Latin *phantasma*) means *image.*

The Greek **episteme** (in Latin *scientia*) means *(scientific) knowledge,* while **epistasthai** (in Latin *scio, studeo*) means **a)** *to know, have knowledge, perceive;* **b)** *to understand, be skilled in;* **c)** *to*

study. Further, the Greek **gnosis** (in Latin *cognitio, notitia*) means *knowledge;* the Greek **ginoskein** (in Latin *cognosco*) and the Greek **gnonai** (in Latin *nosco*) means *to know, understand, recognize.*

There are several other Greek words which include the meaning of »to understand«. For instance: **epaiein** *(obverto, audio)*, **theorein** (*speculor, considero*), **thigganein** *(tango, attingo, appropinquo)*, **ypolambanein** (*suscipio, puto*), etc.

This short collection of terms which include the meaning of the verb »to understand« shows that the notion of »understanding« is copiously present in the meaning of several Classical Greek words.

3.3. The German meaning of the noun »das Verstehen« and the verb »verstehen«

... In so far as understanding and interpretation make up the existential state of Being of the »there«, »meaning« must be conceived as the formal-existential framework of the disclosedness which belongs to understanding. Meaning is an existentiale of Dasein, not a property attaching to entities, lying »behind« them, or floating somewhere as an »intermediate domain«. ...

M. Heidegger (BT) 193

The German verb »verstehen« and to it corresponding noun »das Verstehen« have to be investigated from the viewpoint of the common German language. In German, nouns (of the neuter gender) corresponding to verbs can always be constructed in a regular, unrestricted manner.

First, let us examine meaningfully the split verb »ver-stehen«. **Stehen** (stand, gestanden) means *to stand.* In Latin the meanings are *sto, tolero, patior, perfero, statio.* The Latin »sto« has usual meaning of »to stand«, but also of »to cost«. Among other meanings, the Latin »perfero« means *to bring to a certain place, bring news, bring to an end.* The prefix **ver-** can have several meanings belonging to the class of prepositions, for instance, »pre-« (*earlier than, prior to, before, preparatory or perquisite to, in advance, beforehand, in front of, anterior to*).

Heidegger (SZ) says that in ontical speech the expression »etwas verstehen« is used with the signification of »einer Sache vorstehen können«. The German verb *vorstehen* means, for instance, *to stand before, stand prior to something; to project beforehand;* but also *to head, lead the way,* etc.

3.4. The English meaning of the noun and the adjective »understanding« and the verb »to understand«

... Dasein only »has« meaning, so far as the disclosedness of Being-in-the-world can be »filled in« by the entities discoverable in that disclosedness. Hence only Dasein can be meaningful [sinnvoll] or meaningless [sinnlos]. That is to say, its own Being and the entities disclosed with its Being can be appropriated in understanding, or can remain relegated to non-understanding. ...

M. Heidegger (BT) 193

Let us repeat the meaning of the verb »to understand« in English. **To understand** means

1: a) *to grasp the meaning of;* **b)** *to grasp the reasonableness of;* **c)** *to have thorough or technical acquaintance with or expertness in the practice of;* **d)** *to be thoroughly familiar with the character and propensities of;*

2: *to accept as a fact or truth or regard as plausible without utter certainty;*

3: *to interpret in one of a number of possible ways;*

4: *to supply in thought as though expressed.*

Further, the meaning of the intransitive verb **to understand** is

1: *to have understanding, have the power of comprehension;*

2: *to achieve a grasp of the nature, significance, or explanation of something;*

3: *to believe or infer something to be the case;*

4: *to show a sympathetic or tolerant attitude toward something.*

To understand, comprehend, appreciate are synonyms and mean *to have a clear or complete idea of something. Understand* may differ from *comprehend* in implying a result whereas *comprehend* stresses the mental process of arriving at a result; *appreciate* implies a just estimation of a

thing's value.

The noun **understanding** means

1: *a mental grasp, comprehension;*

2: a) *the power of comprehending, the capacity to apprehend general relations of particulars;* ***b)*** *the power to make experience intelligible by applying concepts and categories;*

3: a) *friendly or harmonious relationship;* **b)** *an agreement of opinion or feeling; adjustment of differences;* **c)** *a mutual agreement not formally entered into but in some degree binding on each side;*

4: *explanation, interpretation;*

5: *sympathy.*

The adjective **understanding** means

1: *knowing, intelligent;*

2: *endowed with understanding; tolerant, sympathetic.*

In English, the splitting of »under—stand« gives the direct meaning of to stand under something; in German, this meaning was, for instance, to stand prior to something. While in German »ver—stehen« can mean to stand in advance to something (to some information) or to have in fact a pre—standing (or pre—under—standing), in English, the preposition **under** means

1: *below or beneath so as to be overhung, surmounted, covered, protected, or concealed by;*

2: a) *subject to the authority, control, guidance, or instruction of;* ***b)*** *receiving or undergoing the action or effect of;*

3: *within the group or designation of;*

4: *less or lower than; falling short of a standard or required degree.*

3.5. The Slovene meaning of the noun »razumevanje« and the verb »razumeti«

... This Interpretation of the concept of »meaning« is one which is ontologico-existential in principle; if we adhere to it, then all entities whose kind of Being is of a character other than Dasein's must be conceived as unmeaning [unsinniges], essentially devoid of any meaning at all. Here »unmeaning« does not signify that we are saying anything about the value of such entities, but it gives expression to an ontological characteristic. And only that which is unmeaning can be absurd [widersinnig]. The present-at-hand, as Dasein encounters it, can, as it were, assault Dasein's Being; natural events, for instance, can break in upon us and destroy us. ...

M. Heidegger (BT) 193

In Slovene, the meanings of the verb »to understand« and the noun »understanding« root in the verb »umeti« and the noun »um« and their derivations.

The Slovene verb **umeti** means **1:** *to know, be skilled (in), be versed (in);* **2:** *to understand;* **3:** *to manage;* **4:** *to contrive;* **5:** *to be able, be capable.* The Slovene noun **um** means **1:** *reason, intellect, intelligence;* **2:** *sense, understanding;* **3:** *mind, brains (pl);* **4:** *wit.*

The direct counterpart of the English to »under—stand« and the German »ver—stehen« is the Slovene »raz—umeti«. **Razumeti** means **1:** *to understand, comprehend;* **2:** *to get, make out;* **3:** *to grasp, apprehend, take in.* The direct counterpart of the English »under—standing« and the German das »Ver—stehen« is the Slovene noun »raz—umevanje«. **Razumevanje** means *understanding, comprehension.*

The prefix **raz-** expresses **a)** *a movement or an orientation to several places, directions;* **b)** *a fact that something is not together, in an original state;* **c)** *a partition, splitting into several entities;* **d)** *an occurrence of a state;* **e)** *an achievement of a desired intention, goal;* **f)** *a cessation of existence, of a state;* **g)** *a contrariness, contrariety.*

3.6. What is understanding as information from the linguistic point of view?

... Understanding is not reconstruction but mediation. We are conveyors of the past into the present. Even in the most careful attempts to grasp the past »in itself,« understanding remains essentially a mediation or translation of past meaning into the present situation. ...

D.E. Linge (PH) xvi

What is understanding as information after the basic semantic analysis of its meaning in Latin,

Classical Greek, German, English and Slovene, and after the rounding up of diverse meaning and understanding of understanding as information in different languages? It can be recognized, in each particular language, how the notion and meaning of the term understanding as information arose through the historical development of a language from the already known other notions, informational entities, through characteristic counter-informing of these entities and through informational embedding of the arisen counter-information concerning understanding into the already existing information of a language. The consequence of this development is that the primordial meaning of, for instance, standing prior to, standing under, be skilled in, etc., arouse into the complex linguistic meaning of understanding.

Examples of meaning of the term understanding in different languages showed slight and essential differences. In fact, the essential differences in German and English, in comparison between the primordial meaning and the new one, have been achieved through the syntactic shifting and connection of words, for instance, from »stehen vor« into »verstehen« or from »to stand under« into »to understand«. This kind of difference is not so essential in the case of the meaning of Slovene verb »razumeti«, where the basic verb »umeti« has a similar meaning. The prefix »raz-« only accentuates a process of dis-cerning (also dis-junction) within reasoning, maybe an analytical component of reasoning (for instance, also the meaning of »dis-reasoning«).

The linguistic analysis of the meaning of the term »understanding« in different languages showed not only the semantic complexity and perplexedness of the term in question, but also its realm of abstractness, bringing under consideration some of currently relevant topics in artificial intelligence, as, for instance, knowledge, belief, commonsense reasoning, intelligence, and maybe, through time, also understanding.

4. What Is Understanding as a Philosophical Realm Concerning Meaning, Hermeneutics, and Semiotics?

... *The concept of understanding as a »fusion of horizons« provides a more accurate picture of what happens in every transmission of meaning. By revising our conception of the function of the interpreter's present horizons, Gadamer also succeeds in transforming our view of the nature of the past, which now appears as an inexhaustible source of possibilities of meaning rather than as a passive object of investigation. ...*

D.E. Linge (PH) xix

Understanding or comprehending of meaning and sense concerns not only speech and language but first of all information as living and cosmic phenomenology in its informationally widened realm. Understanding as an informational phenomenon appears on all, informationally perplexed levels of information. In the reality of the living and non-living, understanding concerns the most primitive physical, chemical, physiological, biological, neural, etc. phenomena, from the level of the primitive informational signaling to the level of the most complex thought concerning perplexed informational processes. To such understanding of understanding we can add some philosophically common facts which will improve and complement the informational image of understanding.

In philosophy, understanding is, for instance, observing or perceiving of the so-called contents of a symbolic or informational expression. Such information of understanding can be, for example, the connection of words in a language with the meaning or the connection of spoken and written text with its senseful contents (thought, fact, situation). Understanding means also developing of an isolated, reasonably chosen element, which brings a new sense, observation, data, by means of senseful connections giving a complete logical comprehension of its meaning, intention, value, etc. Characteristic understanding of this sort is, for instance, the grasping of events belonging to specific historical situations or of behavior belonging to the psychic structure of individuality.

Within the general philosophical sense, understanding means comprehension of meaning and sense, which occur first of all in speech, in the form of the understood Being pervading all relations of the historical being against the world. Thus, the way of understanding from the project of possibilities of a phenomenon to the comprehension of its sense is called ex-position. Further,

exposition being in methodological accordance with a class of rules is called interpretation (in Greek *hermeneia*), however, the science of the art or of the skill of explaining which governs the understanding is called *hermeneutics*.

Understanding as a specific way of cognition in sciences of mind opposes the explaining (in German, *das Erklären*) as a cognitive method in natural sciences. Thus, objects of understanding are tagged by uniqueness and individuality, whereas objects of explaining are characterized by general principles (laws). Heidegger (BT, SZ) has radicalized the hermeneutic problem in the framework of his fundamental ontology as hermeneutics of Dasein and he has determined understanding for the first time not as a kind of cognition being different from explaining but in an existential-ontological way as Being of its own possibility. Heidegger claims that this is the existentiale which primordially pervades all kinds of Dasein's Being and the momentum which together with feeling and speech constitutes the entire existential structure of that »there« as the Being-in-the-world. As the possibility of the most primordial cognition, understanding discloses the circular structure, however, this hermeneutic circle performs not as a circulus vitiosus, but possesses a positive ontological sense. In other words, understanding presupposes understanding in advance (pre-understanding) in which the historical conditionality of its own world experience is included and, simultaneously, is the project to which Dasein opens its own future possibilities. Within this circle the historicity of man is disclosed as simultaneousness of past, present, and future.

In a similar way, Gadamer (PH) tied his philosophical hermeneutics to the occurring of existence and also to the experience of art: understanding is never a determination or a process of the subject; it is an acting-historical occurring (wirkungsgeschichtliches Geschehen). Its system of sense or of horizon is determined by tradition with its own way of interpretation, by which that which will be interpreted is included and all these informational components constitute the entirety of understanding. This is the reason why hermeneutical reflection cannot jump over its own historicity and does not see any barrier of adequate understanding within a time span, but a positive opportunity of fusion of horizons.

Also truth as a historical information, if comprehended in this way, is nothing else than a conscious *prejudice;* and thus, the (so-called scientific) method as a way of cognition in exact sciences is only a *particular form of understanding*, but in no way the only possible origin and experience of truth. First of all, such non-scientific modes of experience of truth are common in art and culture.

Within the historical view, understanding as the grounding of hermeneutics belongs to the Homeric and the biblical hermeneutics. Aristotelian notion of reasonableness or intelligence (the Greek »phronesis« means *mind, will; thought, insight; purpose; high spirit, pride, arrogance*) as a specific practical cognition being characterized by good sense and authenticity (trustworthiness) was recognized later as the proper origin and paragon by the modern hermeneutics. As the art of reading, explaining, and interpreting of texts, the meaning of understanding owes its development certainly to the juridical and philological, but at most to the biblical hermeneutics in modern times (from Spinoza, Schleiermacher, Dithley to the contemporary hermeneutical criticism of ideology).

Semiotics or semiology (from the Greek *sema* or *semeion*) is the study of all patterned communication systems. Primordially, it is the theory of signs traditionally devised into three parts: *syntactics* as the study of grammar; *semantics* as the study of meaning; and *pragmatics* as the study of the actual purposes and effects of meaningful utterances. The idea of semiotics comes from J. Locke, its realization from C. Morris, and further development from R. Carnap. According to Morris, semiotics absorbs the entire logic, mathematics, linguistics, and rhetoric, but also a substantial part of cognitive theory, methodology, sociology of cognition and social sciences; it is also the organon (the body of principles of scientific or philosophical investigation) of all sciences.

Within semiotics, the notions of semantics and pragmatics are the most important in regard to understanding. In contemporary philosophy, a part of semantics investigates relations between signs (symbols, markers) and that what these signs sign (symbolize, mark). A. Tarski defines semantics as the entirety of treating notions which express certain connections between expressions of a language and objects and states of things about which these expressions inform. The key notions of semantics are meaning and truth. R. Carnap

introduced the distinguishing of descriptive (philosophical) and pure (philological) semantics. *Descriptive semantics* is an empirical science which describes and analyzes semantical characteristics of historically given languages. It is divided into special descriptive semantics for examination of distinct historically given languages and general descriptive semantics for examination of common properties of historically given languages. *Pure semantics* is a non-empirical, philosophic discipline dealing with construction and analysis of semantic systems; they can be similar to historically given languages or entirely imagined.

Outside of these distinguished cases of semantics there is the so-called *general semantics* which investigates the meaning of words in common speech, politics, science, philosophy, religion, etc. with the aim to discover linguistic confusion (disorder) and deception (illusion) and to contribute to the creation of improved language which would cause a better understanding among people.

In Greek, »pragma« means *a doing, deed, execution, transaction; action; fact, occurrence; matter, circumstance; business, task, enterprise; affair, object; condition*, etc. *Pragmatics* is a substantial view of informational individuality and subjectiveness, i.e., of informational spontaneity. Pragmatics studies the purposes, effects, and implications of the actual use by a speaker of a meaningful piece of language. J.L. Austin and others have believed that the study of *speech acts* may clarify problems about meaning, reference, and so on. A speech act consists of a locutionary act (saying words), which includes a phonetic act (making noises), a phatic act (using grammar) and a rhetic act (using meaningful words), and a perlocutionary act, by which effects (embarrassment, informational effects) are caused in other people.

Understanding as information (as a basic or composed informational form and informational process) gets a new philosophical illumination, especially when it is developed as a particular formalized theory of information. As we have already recognized, understanding pervades the entire realm of the redefined notion of information, from the most basic concepts of information (spontaneous and cyclic informing, counter-informing, and embedding of information) to the most complex, interweaved, and perplexed (serial, parallel, and serial-parallel) informational forms and informational processes. In this perspective, a new, informational comprehension of understanding is coming into existence which is on the way to deliver new possibilities in exploiting the natural as well as the artificial givens (concepts, principles, constructs) in the informational realm of understanding.

5. How Is Understanding Lost (in Itself) Informationally?

... The meaning of Being can never be contrasted with entities, or with Being as the »ground« which gives entities support; for a »ground« becomes accessible only as meaning, even if it is itself the abyss of meaninglessness. ...

M. Heidegger (BT) 193-194

As information, understanding is confronted with its own and other spontaneous and cyclic informing, counter-informing, and embedding of information, i.e., with its own and other informational forms and informational processes of blindness, intentionality, and/or directionality (circumstantiality) and so the question of its losableness (conditional closeness, stupidity, insensibility) arises. To clear the informational nature of losableness, there would be useful to bring to light its opposites, for instance, availability, disposability, and given information. Further, within the discourse of losableness of understanding as information, it would be recommendable to distinguish the time-concerning states of losableness, i.e., losableness of the present (e.g., »losingness«), of the past (e.g., »lostness«), and of the future (e.g., the possibility of losing, i.e. losableness as it is).

First, how can understanding as information be lost in itself? To answer this question, let us list some Heideggerian concepts of losableness applied to understanding as information in the following way:

- understanding lost in what is encountered within-the-world;
- understanding lost in equipment;
- understanding lost in everydayness;
- understanding lost in factual circumstances;

— understanding lost in irresoluteness;
— understanding lost in just-always-already-alongside;
— understanding lost in the making-present of the »to-day«;
— understanding lost in possibilities which trust themselves upon one;
— understanding lost in publicness;
— understanding lost in something with which information is concerned;
— understanding lost in the »they«;
— understanding lost in the they-self;
— understanding lost in the world of equipment; and
— understanding lost in one's world.

The losableness of understanding as information roots in its informing, that is to say, in the fact how it informs itself and other information and how it is informed by itself and by other information.

Second, what understanding as information can lose because of its own informing and because of informing of other information impacting understanding as information? In Heideggerian terms, for instance, understanding can

— lose its aroundness,
— lose its basis,
— lose its Being,
— lose the Being of its „there„,
— lose its Being-in-the-world,
— lose its equipmental character,
— lose its force,
— lose its genuineness,
— lose its indigenous character,
— lose its involvement-character,
— lose itself,
— lose its readiness-to-hand,
— lose its time, etc.

These examples show how understanding can be informationally investigated against losableness and loosing of understanding. Certainly, to these circumstances many others can be added. The point remains how understanding is lost in itself and how it can lose its power of understanding as information.

6. How Can the Blindness of Understanding Be Broken down?

... Any interpretation which is to contribute understanding, must already have understood what is to be interpreted. This is a fact that has always been remarked, even if only in the area of derivative ways of understanding and interpretation, such as philological Interpretation. The later belongs within the range of scientific knowledge. ...

M. Heidegger (BT) 194

The breaking down as informational phenomenon concerns informing, counter-informing, and embedding of own and other information. By breaking down a monotone (regular, preserving) functioning of informing is stopped because of newly recognized reasons. For instance, a certain style of understanding is broken down if in some of its details it is substantially changed. To break down means to change, supplement, or dismiss the governing principles, beliefs, persuasions, prejudices, to modify them essentially and to get new insight into relating subjects of understanding.

The consequence of breaking down is the so-called breakdown information causing, for instance, physical, mental, or nervous collapse, failure in progressing of an understanding, informational disintegration or decomposition of stable informational processes, division into informational categories, a new classification, etc. An important view of breaking an informing down is the so-called informational decomposition, by which stable and believed informational entities are split into new informational constituents. For instance, an informational formula or operand is decomposed by substitution and essentially changed by the procedures of particularization and universalization of informational operands and operators.

Informational breaking down means to become susceptible to analysis and informational subdivision and to undergo informational decomposition under a new illumination of facts. Commonly, a real informational breaking down opens a new perspective or, as usually said, an abyss of possibilities or principally new informational horizons being not recognized as yet.

Breaking down as a consequence of informing, counter-informing, and embedding of information

interrupts the blindness of informing and suspends the habitual processes of comprehension. This does not mean that the new way of informing does not possess a blindness and that it cannot be broken down into a new perspective. The possibilities of new perspectives are impacted by the own informational arising in general and by the arriving of other information. Thus, informational blindness which can be interrupted through arising of breakdowns is an informationally regular phenomenon of understanding as information.

7. What is Intelligence as Understanding and Understanding as Intelligence?

What is the role of understanding in an intelligent system and intelligent environment? How does understanding impact intelligence and its genuine processes of intelligent informing and how does intelligence as information impact understanding as information?

Let us make a short excursion into the domain of common and uncommon speech with the aim to show some analogies and differences in comprehension of terms »intelligence« and »understanding«. Let us proceed from the simple statement »Information informs.« This statement includes also the meaning of »Information is informed.« The resulting statement is »Information informs and is informed.« The next examples, which concern understanding (at least on the linguistic level) are the following:

Thought thinks and is thought.

Comprehension comprehends and is comprehended.

Perception perceives and is perceived.

Conception conceives and is conceived.

Understanding understands and is understood.

To come closer to the subject of informing, we can repeat these statements in the following way:

Thought as information informs and is informed in the way of thinking.

Comprehension as information informs and is informed in the way of comprehending.

Perception as information informs and is informs in the way of perceiving.

Conception as information informs and is informed in the way of conceiving.

Understanding as information informs and is informed in the way of understanding.

These statements can be broadened even more precisely in the following way:

Thought as information informs and is informed in the way of thinking by itself and by other information.

Comprehension as information informs and is informed in the way of comprehending by itself and by other information.

Perception as information informs and is informs in the way of perceiving by itself and by other information.

Conception as information informs and is informed in the way of conceiving by itself and by other information.

Understanding as information informs and is informed in the way of understanding by itself and by other information.

These statements show the openness, spontaneity, and circularity of subjects in question as information. Further, on the basis of the previous linguistic analysis it is possible to grasp how thought, comprehension, perception, and conception constitute informationally the process of understanding. And each particular process (thought, comprehension, perception, and conception) develops or arises informationally by understanding as an informational process of perplexed information.

How can the intelligence as information be brought into the informational context of understanding? The first step of intelligent approximation would be, for instance:

Thought thinks and is thought intelligently.

Comprehension comprehends and is comprehended intelligently.

Perception perceives and is perceived intelligently.

Conception conceives and is conceived intelligently.

Understanding understands and is understood intelligently.

The meanings of these statements do not contradict in any way the primordial statements concerning

thought, comprehension, perception, conception, and understanding. It could be said that the last statements reasonably include the faculty of intelligence which seems to be a substantial property of the entities in question. Because of the lack of the appropriate verb for the fact »to be intelligent«, it is possible to say:

Intelligence as information informs and is informed in an intelligent way.

Intelligence as information informs and is informed in a thinking, comprehending, perceiving, conceiving, and understanding way.

Thought, comprehension, perception, conception, and understanding as information inform and are informed in an intelligent way.

Intelligence as information thinks, comprehends, perceives, conceives, and understands. Etc.

In short, intelligent informational systems understand and understanding informational systems are intelligent. Further deduction in the showed directions delivers finally:

Intelligence as information informs and is informed in an intelligent way by itself and by other information.

Intelligence as information informs and is informed in a thinking, comprehending, perceiving, conceiving, and understanding way by itself and by other information.

Thought, comprehension, perception, conception, and understanding as information inform and are informed in an intelligent way by itself and by other information.

Intelligence as information thinks, comprehends, perceives, conceives, and understands itself and other information and is thought, comprehended, perceived, conceived, and understood by itself and by other information.

Understanding can be thought as a broadening of the notion of intelligence and intelligence can be comprehended as the substantial faculty of understanding. In this way, artificial understanding can be the broadened term for artificial intelligence. If we speak about understanding we have in mind intelligent informing within understanding as informing.

8. Principles of Understanding as Principles of Information

How does information qua information concern the principle of understanding? How is understanding as a general informational component present in informational components of an informational entity? How can understanding as the informational component within the so-called basic informational cycle be investigated, for instance, as a component of informing, counter-informing, and embedding of information? The answers to these questions should explain the understanding nature of information as a basic principle which till now remained concealed, i.e., explicitly unrevealed.

Let us begin the discourse of understanding within the basic informational cycle by understanding as information dwelling within informing, counter-informing, and embedding of information. Taking a time slice of observing an informational entity, we state that this entity informs, counter-informs, and embeds information. We say that information informs itself and other information, that it counter-informs or produces from itself and other information the so-called counter-information, and that it embeds informationally the produced or informationally arisen counter-information and other incoming (to its »consciousness« arriving) information. This process of informing, counter-informing, and embedding constitutes the so-called basic informational cycle of the informational entity in question. And within this cycle, its constitutive components demonstrate the so-called capability of understanding in the form of observing, investigating, and interpreting of information.

As we can remind, counter-informing of information and embedding of arisen information was a characteristic act of understanding of arisen information by source information and the new information could be embedded into existing or source information merely on the basis and to the extent of its understanding. Here, understanding as information was meant as a new embedding information which arose too, and especially this embedding information has enabled the »understanding« of arisen information.

After this discussion it is possible to resume the question how could the understanding as information be present in the basic spontaneous and cir-

cular processes of informing, counter-informing and embedding of information. To some extent, depending on the nature of information in question, understanding as information can be understood to be the constitutive component of informational forms and informational processes in the form of arising and embedding of information. Thus, understanding as information is merely another marker for that what was called observing, investigating, and embedding of information appearing in informing, counter-informing, and embedding of information. Thus, understanding is a basic faculty of information which arises and embeds information.

9. Informational Formalization of Understanding

Which are the formal-theoretical possibilities for informational axiomatization and theory of principles and problems of understanding?

First, formalization of understanding has to proceed from the fact that understanding performs as regular information, i.e. that within the informing of understanding, counter-understanding is coming into existence and that this counter-understanding is informationally embedded into existing understanding of information. In this way, understanding, counter-understanding, and embedding of understanding constitute the so-called spontaneous informational cycle of understanding.

Second, informing of information can be comprehended as the possession of an informationally innate component of understanding of information as information: when information is coming into existence as counter-information, this counter-information has to be informationally connected (related) to, interweaved with, or embedded into the existing source information. The act of connecting, interweaving, or embedding is by itself a kind of understanding of the arisen counter-information (and other information) by the source information. In this way, information possesses an innate component of understanding.

Third, information, by which an informational entity is informed, is informationally perceived by this entity within the ability of its own informing. If information does not impact an informational entity at all, this information performs as a pure informational noise which simply cannot be perceived by the entity in question. A similar effect of perceptional lack can occur in the case of information's counter-information. In general, perceiving (perception, comprehension, understanding) of information by information is always an informational process of a noisy transition or informing of information (misperception, miscomprehension, misunderstanding). Perception of the own counter-information or other information means simply the filtering of perceived information by informational abilities of perceiving information.

10. Understanding as an Informational Principle of Expert Systems

How could understanding as a natural principle be introduced (as an explicit informational process) into expert systems? How could understanding as information be applied at the design of an electronic dictionary? Which are the basic components of understanding as an informational process over a certain informational domain (of knowledge, experience, expertise, etc.)?

Understanding is a modus of informing by which the so-called understanding information is coming into existence. If an informational entity (operand) produces understanding as a part of its informing, then this entity by the informing of understanding produces (informs) information which explains or interprets certain informational objects. In this way, understanding as an informationally active component of an informational entity has to observe, investigate, evaluate, clarify, explain, interpret certain informational objects and has to deliver the resulting information of understanding.

Nowadays expert system use a kind of propositional or programming logic to handle (operate, connect, relate, overview, etc.) the so-called facts, i.e., believed, implicatively changeable, expertly inputted data, knowledge, rules, etc. In fact, they perform merely as an implicatively organized information support for the living experts or learning personnel, however, cannot be used for crucial decision making in business, management, technology, etc., i.e., as the replacement for intelligent and responsible decision makers. And if so,

why should the expert system be not expanded up to a kind of informational expert systems, keeping the existing knowledge base and implicative rules of the classical expert system, but adding some innovative mechanisms of understanding (informational inferring) in the sense of informational arising. How would this widened concept of producing (informing) of expertise improve the performance of existing expert systems?

If we accept the possibility of the concept of informationally widened expert system, the following question arises: Which are the elementary faculties we have to add to the classical expert system to achieve an informational expert system? Within this paragraph we shall try to list in short some of the basic properties of an informational system in comparison to a nowadays expert system. And we shall give a short comment on the question how could an electronic dictionary be improved by the use of informational principles to approach the living searcher and user of an electronic dictionary.

Let a linguistic (e.g., textual) expert system for some domain of knowledge be given. It means that entities (marked, in general, by $\hat{e}$) of this system are, in general, sets of informationally interconnected (interweaved, perplexed) clauses, expressed in a (natural, formal, artificial) language. Now, the following questions can be answered:

0. How does an informational entity $\hat{e}$ inform in general?

1. How does $\hat{e}$ inform itself?

2. How is $\hat{e}$ informed by other entities?

3. How does $\hat{e}$ inform other entities?

4. How does $\hat{e}$ counter-inform and is counter-informed?

5. How does $\hat{e}$ embed new information, i.e., its own counter-information and other information?

The answers to these questions constitute the basic as well as a perplexed informational cycle of informing and, in particular, as informing of understanding. The question is the following: Is an informational expert system in the sense of the listed questions at all possible?

The answers to the listed questions have to be constructive in the way that they enable the construction (design, implementation) of an informational expert system. Simply, the chosen informational entity $\hat{e}$ must have the faculties corresponding to questions 0—5. In this case another question arises: whether $\hat{e}$ must have all these faculties by itself or can they be taken (or be applied) as informational procedures belonging to a common collection of general informational properties (informational tools, principles). This second possibility seems to be quite reasonable.

10. 0. How does an informational expert entity $\hat{e}$ inform in general?

First, let us say how does $\hat{e}$ inform. In which way does $\hat{e}$ inform? How does a general mechanism of informing looks like and how can it be used?

Entity $\hat{e}$ informs in the way that it informs and can be informed. It informs in the way that within $\hat{e}$ information arises, namely: because of information already existing in $\hat{e}$; because of information appearing outside of $\hat{e}$ and informationally impacting $\hat{e}$; because of $\hat{e}$'s own counter-information arising out of $\hat{e}$; and because of informational embedding of its own counter-information and outside information. These four informational processes constitute the so-called *informational cycle*. Informing of information is cyclically spontaneous and spontaneously cyclic.

In the informational domain of $\hat{e}$, $\hat{e}$'s informing, counter-informing, and informational embedding arises as informing of information $\hat{e}$ and as informing of outside information which informationally impacts $\hat{e}$.

In our case of informational expert system, $\hat{e}$ is a set of clauses. Entity $\hat{e}$ informs by memorizing and generating of clauses and by perceiving of clauses as an outside information. Informing of $\hat{e}$ is nothing else than memorizing and creating of clauses by means of $\hat{e}$'s informing, counter-informing, and informational embedding, within the cyclically spontaneous informing of entity $\hat{e}$.

The basic question remains how can the informational mechanism for generation of clauses be informationally structured and organized in the realm of an entity $\hat{e}$ within an informational expert system. What could be the mechanism of analysis of existing and arising clauses? We shall answer these questions later. Let us suppose that such mechanisms exist and that they analyze existing clauses and synthesize new (arisen) clauses.

10.1. How does an informational expert entity ê inform itself?

How does an expert entity $\hat{e}$ inform itself or is informed by itself? Entity $\hat{e}$ is a set, sequence, or set-sequence of clauses. There can (or must) exist particular clauses (for instance, special formulas) which connect, interweave, or perplex clauses within $\hat{e}$ informationally. In principle, $\hat{e}$ is an informational interweaving (a multiplex, perplexity, or similar structure) of clauses. Clauses can take over the operand roles as well as the operator roles. Operator clauses are, for instance, informational programs for analysis and synthesis of clauses, for embedding of synthesized (generated, arisen) clauses and are from other informational sources (entities) arriving clauses. In general, $\hat{e}$ includes clauses which at the given state of $\hat{e}$ are operator clauses and operand clauses. In another state of $\hat{e}$ the roles of previous operator clauses and operand clauses can be changed. There can even exist clauses which simultaneously perform as operators and operands, i.e., as operator-operand clauses.

Potentially, each clause of $\hat{e}$ can perform as the operand clause, operator clause, or operator-operand clause. A passive clause of $\hat{e}$ can be treated as an operand clause.

Clauses possessing the status of operator clauses operate on clauses possessing the status of operand clauses. The question is how operand clauses can be chosen by operator clauses for operations upon them. In which way an operator clause can find the argumentation or the cause for operating upon an operand clause, to change it and to generate in a counter-informational way a new clause and to embed a new clause informationally, i.e., to connect it to some other clauses of $\hat{e}$?

The imagined faculty of clauses to be operators and operands of entity $\hat{e}$ demands a concrete concept, i.e. a real changing and arising of clauses through their particularization and universalization.

10.2. How is an informational expert entity ê informed by other expert entities?

Entity $\hat{e}$ is informed by other information through the arriving information. The question is how information arrives to $\hat{e}$ or where is the sensitivity of $\hat{e}$ for the arriving information located. Which partial information of $\hat{e}$ decides on the acceptability of the arriving information? The answer to these and similar questions is that there exist a general mechanism which qualifies each entity $\hat{e}$ within an informational expert system for the acceptance of and for the sensitivity against the information »arriving« to $\hat{e}$.

The question which has to be resolved is what is the nature (structure and organization) of the sensibility mechanism for the arriving information and how this information is processed (informed, counter-informed, and embedded) within $\hat{e}$. In fact, there does not exist a substantial difference between the so-called information of $\hat{e}$ and the arriving information in regard to sensitivity. Entity $\hat{e}$ has to sense (over a general mechanism) its own information (its informational perplexity of clauses) in a similar manner as it senses the arriving information. The question remains how to chose the arriving information as that which in regard to $\hat{e}$ is the sufficiently *relevant* information.

On the basis of sensing its own and the arriving information, $\hat{e}$ synthesizes new clauses. These clauses can be understood to be informed and counter-informed clauses which contents depends on the state of the informational set $\hat{e}$ and the sensed arriving information. This phase of the synthesis is informational and counter-informational. Now, the synthesized clauses have to be connected (informationally interweaved) with clauses belonging to the informational set $\hat{e}$. It means that these clauses become meaningful in the realm of entity $\hat{e}$. The act of connection is performed by the so-called operation of embedding which can be constructed as a general, common principle, valid for any type of entity $\hat{e}$.

It can be conceived that $\hat{e}$ includes some particular clauses which determine the possibilities of embedding of the arisen counter-information and the arriving information. To these particular, internal embedding clauses a general embedding mechanism (within the informational expert system) is on disposal by which the embedding of arisen and arriving information can be efficiently performed. It is worth to mention that the so-called embedding underlies the general principles of information. Embedding in a particular situation constitute embedding operand-clauses, embedding operator-clauses, and embedding operand-operator-clauses. In this way counter-informing

and embedding underlie the general (and hierarchically higher) principle of information.

10.3. How does an informational expert entity ê inform other expert entities?

It was already described in which way the entity *ê* can inform (also counter-inform and embed). The informational connectedness of *ê* does not concern merely *ê* but also other informational entities of type *ê*. Otherwise, it would be not possible to say that *ê* informs other informational entities by its own intention. It is to understand that the informational connectedness of informational entities (sets of type *ê*) within an informational expert system is mutually perplexed. Thus, a concrete entity *ê* can decide to which entities its informing and by its informing produced information will become accessible or to which entities such information will be mediated. Thus, in this sense, it is possible to say that entity *ê* informs other entities within an expert system.

To inform other entities can mean simultaneously to produce information intentionally for itself and for other entities in question. Informing is the synonym for informing, counter-informing, and embedding of information. Informing can mean that, for instance, the existing information keeps its form and new clauses are generated; parts of existing information are modified, changed, canceled and on this basis new information arises, etc. As mentioned before, this principle of informational arising is effective in informing, counter-informing, and embedding of information. Thus, it is necessary to analyze the possibilities of informing concerning counter-informing as well as the counter-informing following phase called embedding of information arisen by counter-informing.

10.4. How does an informational expert entity ê counter-inform and how is it counter-informed?

What kind of informational mechanism is needed for the implementation of counter-informing within an informational expert system?

By definition, counter-informing is an unforeseeable process of informing. How could the unpredictability of counter-informing as the generating of new, not ultimately in advance determined clauses of entity *ê* be constructively implemented? Probably, several generating mechanisms for this purpose can be chosen, applying the methods of random construction of clauses from given clauses and random choice of various possibilities. Evidently, the mechanism of counter-informing within an informational expert system could determine the quality (intelligence, diversity) of the system's inference mechanism.

The unpredictability of counter-informing can depend substantially on the unpredictability of arriving information. This information can cause the generation of *ê*'s clauses which in no way could be determined in advance. In this way the arisen clauses are the consequence of system's »understanding« of instantaneous situation. In this point of the problem, it becomes obvious how understanding of an informational expert system is the informational problem par excellence. Generated clauses are not only operand-clauses; they can be operator-clauses or operand-operator-clauses. as well. By counter-informing, system acquires not only new data but also new functionality.

On the other side, entity *ê* is counter-informed only to the extend by which the arisen counter-information and the arriving information can be embedded into *ê*. Prior to embedding of information, this information does not reach the so-called »horizon« of understanding of *ê*. It is, for instance, clear that *ê* counter-informs, however, it must become evidently clear how entity *ê* is counter-informed.

10.5. How does an informational expert entity ê embed information?

Informational embedding reflects in the most direct form the »understanding« of the existing informational expert system. New information—the arisen and the arriving one—has to be connected to the existing (given, source) information, so the new information can be understood by information which already exists in the system. Understanding means nothing else than informational connectedness of new information with the existing information. If this connectedness is weak, understanding of new information by existing information can be faulty or deficient. This does not mean that the deficiency in the process of informational embedding will not be reduced or sup-

pressed.

Embedding of information occurs by means of specific and general clauses of an informational expert system. Specific clauses belong to the entity $\hat{e}$ in question while general clauses for embedding are principles of the expert system as a whole. The set of embedding »rules« is variable; it is a consequence of the instantaneous state of the expert system. Embedding is not an informational exception and it underlies the general principles of information. The product of embedding is the so-called embedding information which determines the connectedness of some informational parts (clauses) with other informati nal parts (clauses).

10.6. What is the nature of embedding information into an informational expert entity ê?

The embedding information is the product of informational embedding and concerns the arisen or arrived information on the one side and the existing information of $\hat{e}$ and other informational entities on the other side. The embedding information is a specific information of clausal connectedness within $\hat{e}$ and outside of $\hat{e}$. This information can be passive as well as active, i.e., can have the form of operand clause as well as operator clause or operand-operator clause.

It is clear that the embedding information can connect as well as disconnect certain information. Disconnection means not only the annulling of certain embedding clauses but also their changing. Embedding of information has to be understood as informational connecting and disconnecting of information by means of embedding information. In this regard, the embedding information by itself can be connected and disconnected to some extent. Embedding information performs as regular information and underlies the general principles of information.

10.7. What is a complete informational cycle of an informational expert entity ê?

In an informational expert system, the so-called *informational cycle* is the basic principle on all levels of the system performance. Under a complete informational cycle of an entity $\hat{e}$ the process of informing, counter-informing, and embedding of information is understood, being closed into an open circle or loop of itemized components of informing. The immediate question of this formula is which are the informational levels of an informational expert system within which informational cycles occur.

The lowest informational level is the domain of a clause representing the facts of an application. The fact-clause $\hat{e}$ is the most elementary informational entity within the system. Thus, a clause $\hat{e}$ informs, counter-informs, and embeds information by its open cycle, producing information, counter-information, and embedding information. The function of the informational cycle is performed by $\hat{e}$ as informational entity and by means of system's general informational principles being in common for all entities of the system. In this case, the general informational principles have to be understood as informational procedures which functionality or ability of informational performance is memorized within the system. These principles perform as regular information, i.e., as clauses (operands, operators, operands-operators). In each concrete state of the system, clauses, i.e. $\hat{e}$-s, principles, etc. can be changed accordingly to the instantaneous situation.

A higher informational level is the domain of clauses which connect clauses as facts. Within the domain of connective clauses several categories can be distinguished. The first category are clauses which belong to the so-called »classical« rules of inference (for instance, an if-then statement, implication, or modus ponens) which connect the clauses representing facts of an application. These rules belong to the techniques of classical expert systems and can be preserved under specific conditions also within the informational expert systems. For these category of connective clauses their ultimate dependence on the factual situation or application is evident. These primitive forms of inference are applied in an informationally circular manner.

The second category of connective clauses belong to the class of the so-called modi informationis (IL-IV). An informational expert system can use in an informationally mechanic way the following informational modi: modus ponens, modus tollens, modus rectus, modus obliquus, modus procedendi, modus operandi, modus vivendi, modus possibilitatis, modus necessitatis, etc.

which are nothing else than rules of informational inference. Certainly, these rules are application dependent, however, they preserve certain general principled properties of informational arising. Within an informational cycle, different modi informationis can be applied.

The third category of connective clauses are semantic clauses determining semantic relations among clauses. These clauses serve as operand-clauses and are used in the processing of clauses by various »rules« of inference.

The highest group of clauses used within an informational cycle are informational principles on the system level being responsible for the informational nature of the discussed clauses of the informational expert system. This principles-clauses are active on all levels of the expert system as mechanisms of various informational cycles.

The hierarchy of informational cycles is determined by the rank of rules: the primitive, higher, and the highest ones. Beside of the hierarchy within an informational cycle there exist parallel and perplexed cycles (loops or subcycles). Thus, the parallelism of cycles within a cycle is the most powerful way of informing within an informational expert system.

10.8. An informational electronic dictionary

How could the search in an electronic dictionary be improved by the use of the discussed informational principles? What is the spontaneity and the circularity of search in a dictionary? What is the difference in quality of meaning or understanding of a searched notion in a dictionary?

The question we ask ourselves is how do we search for a meaning of a given term in an integrated, for instance, a multilingual dictionary (set of different dictionaries). Why we can obtain different or less or more exhaustive meanings of selected terms as the consequence of searching by different persons? Does an ultimate meaning of a term exist at all or can the understanding of a term be developed ad infinitum?

Necessarily, a dictionary does not mean a finite domain within which a term could be understood semantically and pragmatically to a definite extent. The meaning of a term depends on the intensity, intention, and kind of the act of understanding and on the decision of the searcher that he/she is satisfied with the meaning obtained and constructed as a result of searching in a dictionary. In fact, the definite meaning of a term does not exist, for meaning is a regular information and it arises through informing, counter-informing, and embedding of the available and the individually arisen information. The informational availability depends not only on information included in a dictionary, but also on the needs, circumstances, and at last but not least on a searcher's metaphysical orientation. This facts point to the belief that an electronic dictionary has to be organized as a dedicated informational expert system and not only as a classical expert system which is based predominantly (or even solely) on formal implicative logic and on (solely) traditional algorithmic procedures.

The use of an electronic dictionary is evidently not a mere routine operation, by which a user would be capable to find one or whichever of the possible meaning of the searched item. Such strategy of searching would not call for the ambitious development of highly automatized dictionary without some of informational principles of understanding in mind.

11. Misunderstanding as Information

... The organism creates the disease to recover itself. ...

G. Canguilhem (NP) 20

There could be said: the understanding creates the misunderstanding to understand itself. Or more precisely: understanding as information counter-informs misunderstanding to embed it informationally into understanding. What is the informational difference between understanding and misunderstanding as information? From the informational point of view of understanding, misunderstanding means another, informationally irrelevant or noisy form of understanding. To confront understanding and misunderstanding of a given informational entity means to investigate and evaluate the informational difference between understanding as information and misunderstanding as information.

As we can comprehend, there does not exist a reliable general principle for distinguishing the

essence of understanding and misunderstanding of a concrete informational entity. We have already recognized how the so-called breakdowns can interrupt informational cycles of a misunderstanding (an informational blindness), beginning a new course or cycle of informational arising within the cycle of »correct« understanding. However, a given understanding can arise more and more into a form of misunderstanding and this course of informational development can be broken down every now and again.

Understanding and misunderstanding of an informational entity can be two different informational processes arising around the same informational subject (information in question). They treat this informational subject differently, so, through the so-called breakdowns of understanding and misunderstanding, respectively, they can change from one form into another. Beside of this general principle of understanding there can exist forms of understanding which can be marked according to the social normativeness as disorders of understanding. In this case, disorder of understanding can mean a certain degree of misunderstanding and non-understanding.

Understanding and misunderstanding are two different views of a given information, which can have contradictory or semantically and pragmatically non-overlapping opinions, truths, beliefs, and/or comprehensions of information in question. Or, they do not have essential common, ageeable informational items concerning the informational entity in question. Also, understanding and misunderstanding usually disagree in the importance of informational arguments by which an informational entity will be evaluated, investigated, and/or comprehensively demonstrated. It is simply informationally impossible to lead understanding and misunderstanding to a common realistic or rational interrelation and senseful comparison, if they have too divergent, incomparable, and informationally strange, incompatible, inconsistent, and mutually unreliable nature. To understand or to misunderstand requires and demands the intention to be understood or to be misunderstood, respectively.

Understanding and misunderstanding are two informational poles of the same informational capability concerning the informational arising of understanding and misunderstanding as information. Understanding can be treated as misunderstanding and vice versa from the point of view of information which understands or misunderstands a certain information. If understanding concerns opinion, truth, belief and comprehension, misunderstanding concerns counter-opinion, untruth, counter-belief, and counter- comprehension. Thus, misunderstanding can be understood as a kind of understanding's counter-information, which to some extent can be or can to some extent not be informationally embedded into the existing understanding of information.

Living information is always on the way to understanding and misunderstanding simultaneously, for only a kind of contradiction enables the development and the actuality of the existing understanding and misunderstanding of some information. Thus, both understanding and misunderstanding perform as regular information and interact informationally in their own, however, mutually impacted informational manner. It may simply be said that neither understanding nor misunderstanding can have the finite opinion, truth, belief, comprehension, rationality, etc. tenanted, for these are only instantaneous informational items of relevance within an arising understanding and misunderstanding.

12. Disorders of Understanding

... Viewed semantically, the pathologic against the normal is marked not so much as by a *or* dis *as by* hiper *or* hipo, *respectively. Also, if we keep within an ontologic theory its calming confidence in the possibility to overcome an illness [disorder] technically, we are still very far apart from the thought that health and illness could be qualitative opposites, struggling powers. ...*

G. Canguilhem (NP) 22

In its broadest sense, a being's understanding is conditioned and impacted by the instantaneous state of its autopoietic system and particularly by the instantaneous state of its neural system. In medical sense, *disorder* is an alteration of the function or structure of an organ or the body as a whole. Understanding as information can be impacted by various kinds of disorder causing disorders of understanding.

Disorders of understanding can occur in forms of cultural information processes (for instance, as indoctrination, totalitarian ideology, individual cynicism, philosophic teachings, scientific theories, etc.) as well as in forms of »pathologic« processes (illness, mental disorders, insanity). For instance, a rigid ideology can be understood as communicated insanity or a shared delusion, which is unable to understand the wrongfulness of conduct.

Informationally, disorders of understanding in living beings occur also as consequences of various disorders, for instance, as acute brain, affective, aggressive behavior, conduct, dissociative, gnostic, histrionic personality, impulse, motility, neurotic, paranoid, psychosomatic, schizophrenic spectrum, speech, stress, thought, XXX disorder, etc. In these cases, disorders of understanding are the consequence of »pathologic« states, conditions, defects, abnormalities, particularities, etc.

Acute brain disorder is an organic mental disorder which is reversible, such as toxic-infections delirium. This disorder can impact the functioning of brain parts responsible for the degree of human understanding.

Any disorder typified primarily by disturbances of mood, including the manic-depressive psychosis and the neurotic-depressive disorder is called *affective disorder.*

Aggressive behavior disorder is the label for a group of *conduct disturbances* in children, characterized by repetitive and consistent antisocial activities.

Dissociative disorder is any of the syndromes whose central feature is sudden, temporary alternation in consciousness or identity (multiple personality and depersonalization).

A group of disorders involving abnormalities of perception or recognition is called *gnostic disorders.* These are abnormalities of associative thinking such as may be observed in schizophrenia, to disturbances in cortical functioning as seen in some types of Broca's or syntactical aphasia, and to deep, epicritic sensation in contradiction to protopathic sensation disturbances.

Histrionic or *hysterical personality disorder* is characterized by egocentricity, attention-seeking behavior, suggestibility, overdramatization, sexually provocative behavior, proneness to easy disappointment, and emotional lability.

Impulse disorder is characterized by ego syntonicity, a pleasurable component, minimal distortion of the original id impulse, and some degree of lack of control (pathologic gambling, kleptomania, pyromania, and other acts of destructibility).

Motility disorder is an abnormality of movement, particularly the abnormal postures and gestures displayed in catatonic schizophrenia and in pervasive developmental disorders.

Neurotic disorder (neurosis) is any of functional disorders of behavior caused by excessive anxiety or by behavior distorted by an exaggerated use of avoidance behaviors. In psychoanalysis, neurosis is the symptomatic expression of conflict between the id's sexual and aggressive impulses and the ego's need to cope with and adapt to reality.

Paranoid disorder (paranoidism) is a condition that is similar to paranoia (also called *intellectual monomania*) which comes from the Greek *paranoein* and means to think amiss or wrongly. Paranoia is a gradually developing delusional state or fixed delusional system with preservation of intelligence and orderly thinking. Most commonly the delusions are persecutory in nature, although they may also be of grandiose, religious, ideologic, erotomanical content, etc.

Psychosomatic disorder concerns structural, physiologic, or other organic changes in one or more body systems, whose origin, at least in part, is related to emotional factors.

Schizophrenic spectrum disorder (schizoidia) is a range of character and behavioral disorders that may be related genetically to schizophrenia. These disorders may include impulsive crimes, social isolation, alcohol abuse, eccentric reclusiveness, suspiciousness, jealousy, litigiousness, fanaticism, opinionated and narrow-minded pedantry, haughtiness, and unsociability.

Speech disorder means difficulty in producing the sounds of spoken language; it may occur as a developmental condition or as a result of various acquired neurologic disorders.

Stress disorder means transient situational personality disorder. *Post-traumatic* stress disorder is associated with serious traumatic events. Classic symptoms are re-experiencing the trauma in dreams, recurrent images and thoughts, a general sense of numbness and feelings of lack of involve-

ment with the real world, etc.

Thought disorder is understood to be any abnormality in the process of thinking or a disturbance seen in schizophrenia characterized by blocking of thought, thought deprivation, poverty of thought, or haphazard associations. A good example of this type of disorder is, for instance, a totalitarian ideology.

XXX disorder is a case of the so-called chromosomal disorder causing mental deficiency of subjects.

Let us step into similar kinds of abnormal understanding from the point of view which is offered by some principles of neural science (PNS). It is evident that informational principles are in commonsensical accordance with some principles of neural science and vice versa. In many respects, informational phenomenology is becoming inevitable in neural research, in explaining and interpreting the processes of mind.

Let us overview in short localization of higher functions and the disorders concerning the higher cortical functions relating language, thought, and affect. Hemispheric *brain asymmetries* and the cortical localization of higher cognitive and affective functions seems to be relevant in the process of human understanding. The two hemispheres are not fully symmetrical and differ in their capabilities. Split-brain experiments reveal important asymmetries and show that consciousness and self-awareness are not unitary. What functional advantages, if any, does lateralization confer? Anatomical asymmetry of the brain has been demonstrated in other animals and there are interesting sex differences in hemispheric dominance.

To some extent, even the most complex functions of the brain can be topographically localized. This is important in explaining syndromes, disorders, and diseases in specific regions and, certainly, information phenomena in the nervous system.

The next important questions within human understanding are natural language, disorders of language, and other localizable disorders of cognitive functioning. All *human languages* share four distinct features: creativity, structuring, meaningfulness, and interpersonal impact. The arising of human language originates in evolution of the brain, thus, the capability for human language is thought to be an innate cognitive skill as well as learned. Further, aphasias are disorders of human language that also interfere with other cognitive processing (informing). For instance, the aphasias can be understood on the basis of the Wernicke-Geschwind model of language. Aprosodias are disorders of the melodic intonation of language and its perception. Some disorders of reading and writing (alexias, agraphias, dyslexia, hyperlexia) can also be accounted for by the Wernicke-Geschwind model.

Disorders of thought can be thought (classified) as the schizophrenic syndromes. There is an important genetic (molecular-informational) component to schizophrenia.

Disorders of feeling are (information) processes classified as affective diseases (depression, mania). These disorders suggest a defect in the hypothalamus and a strong genetic predisposition, for instance, for major depression. A biogenic amine hypothesis of depression has been proposed and there are disordered neuroendocrine functions in depression.

Disorders appearing in human mind and/or human brain can substantially impact the so-called human understanding (social models of behavior) and influence the understanding arising in human communication.

13. Understanding as Information in Natural Languages, Sciences, and Cultures

What does understanding as information mean within the domain of a natural language, science, and culture? The trivial answer would be that understanding in natural language concerns the cultural environment in which a natural language arises in the form of discourse from its historical roots and in the everyday use. Science as a cultural form is impacted by semiotic capabilities of a natural language as well as culture of thought and labor. If language is embedded in cultural information as a special, communicative means, science as an informational form is embedded in natural language and from this language deduced formal language of the science in question. Thus understanding as information depends culturally, linguistically, and scientifically.

Cultural forms of information are, for instance, philosophy, art, history, technology, but also language and science (ID1, ID2). In this respect several natural languages, sciences, and cultures are informational phenomena which are informationally interconnected and perplexed and they constitute the cultural behavior and the mentality (truths, beliefs, styles of life) of individuals and populations through time within which this informational forms will be understood.

14. Conclusion of Part One

In this part of essay we have set the background of understanding as information from various aspects of information: philosophic, linguistic, neural, comprehensional, and informational. This cultural and individual background can now be used for the construction of a formal informational theory of understanding which from this philosophical basis can be axiomatized, deduced, and induced in an informationally formal (logical) way. Thus, in Part Two, we shall show this way of formalization by a special, informational logic of understanding, which will embrace the fundamental concepts of information (informing, counter-informing, and embedding), i.e. informational principles (POI), as well as the most complex informational forms of understanding.

We have seen how understanding as information can be built up as the most complex and also intelligent information. For this reason particular, i.e., intelligent informational forms has to be determined. The basic formalistic concept of describing intelligent informational entities was shown schematically in (IPF), for instance. The form of an intelligent process was an informationally perplexed and recursive formula or system of formulas. Thus, it will be possible to show some particular cases of parallel-cyclic scenarios of understanding, from the basic informational level to the most perplexed one.

We have recognized how understanding as information can be constructed as a complex informational entity, as an informational process delivering information as understanding which concerns other informational entities and also understanding as information itself. In this way, understanding performs informationally regular as any other information with information-owing capability of informing, counter-informing, and embedding of information. As we shall see, the formal-theoretic treatise of understanding as information will reveal and brighten the perspective of understanding as a horizon of possibilities being relevant for philosophy of understanding as well as for the design of understanding artifacts.

References

(MP) Aristotle, *Metaphysics*, The University of Michigan Press, Ann Arbor, Mi (1985).

(PH) Gadamer, H.-G., *Philosophical Hermeneutics*, University of California Press, Berkeley, Ca (1976).

(NP) Canguilhem, G., The Normal and the Pathologic (in Slovene), Studia Humanitatis, Ljubljana (1987).

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

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

(PNS) Kandel, E.R. and J.H. Schwartz, *Principles of Neural Science* (Second Edition), Elsevier Science P.C., New York (1985).

(POI) Železnikar, A.P., *Principles of Information*, Cybernetica **31** (1988) *2*, 99-122.

(ID1) Železnikar, A.P., *Information Determinations I*, Cybernetica **31** (1988) *3*, 181-213.

(ID2) Železnikar, A.P., *Information Determination II*, Cybernetica **32** (1989) *1*, 5-44.

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

(IPF) Železnikar A.P., *Informational Principles and Formalization* (in Slovene), Informatica **13** (1989) *3*, 21-40.

DESIGN OF HARDWARE WITH PROGRAMMABLE GATE ARRAYS

INFORMATICA 3/90

Peter Praprotnik
Iskra TELEKOM
Bogdan Dugonik
Tehniška Fakulteta Maribor

Keywords: SI-2000, Programmable Gate Arrays, Group Switch Module, Hardware Deisgn

Abstract: This application shows how pure hardware problems can be solved with modern software tools. There was a need in Iskra Telekom for the new Group Switch module for SI2000 digital telephone exchange. Because of the complexity and the number of functions the module has to provide, there was not enough room on the board for standard hardware components, so we decided to use Programmable Gate Arrays (PGA).The most interesting in this application is the direct hardware dependence on the software. In this paper some solutions are given.

In the case of using PGAs, the main problem is the development of programs. The complete development of contents, simulation, real time simulation and verification is made by using special software on fast Personal Computers or even better on a Sun. With such tools it is possible to manage one PGA in three or four weeks in the laboratory without involving big manufacturers of electronic components. The main advantage of PGA is that you just download another program in it and you already get different hardware on the board without changing any piece of the hardware. Another advantage is that this can be done during operation of the target system. It is obvious that such software development of hardware design will be applied more often in the future.

Introduction

There was a need to improve one part in the hardware of the SI 2000 digital telephone exchange. The old Group Switch Module which switched 32 subscriber modules has to be reorganized. Each of this modules handles up to 240 end users. Our new Group Switch Module can now manage four times more subscriber (analog or digital) modules with much better time characteristic. Our problem was to replace approximately 160 SSI and MSI chips on the old circuit board with Gate Arrays or with the full custom application specific IC without increasing production costs. At the end of the cost analysis, it was found out, that our problem could be solved only by using Programmable Gate Arrays.

SSI (Synchronization, Switch and Interface) is one of the centralized modules in the SI-2000 Digital Switching System and it is the heart of the Group Switch Unit, that is capable to handle 128 subscriber modules. SSI provides some vital functions (i.e. connections between modules and channel switching) necessary for the operation of the whole system. Because of the importance of these functions the reliability of the module has to be high. It is achieved by duplication of the vital functions, where failure would cause the fall of the whole system. It is possible to connect MLI (Module, Link and Interface) units on each SSI module. The module is performed with the standard HC-HCT MOS chips as well as with PALs and PGAs. The whole contents of the module is equivalent to 165 sixteen pin chips.

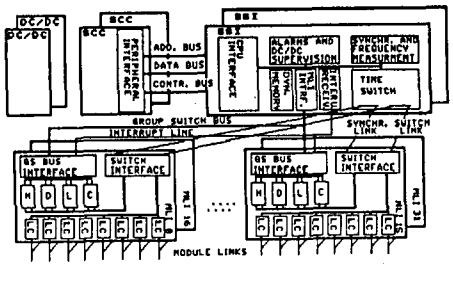


Fig.1. SI-2000 Group Switch module.

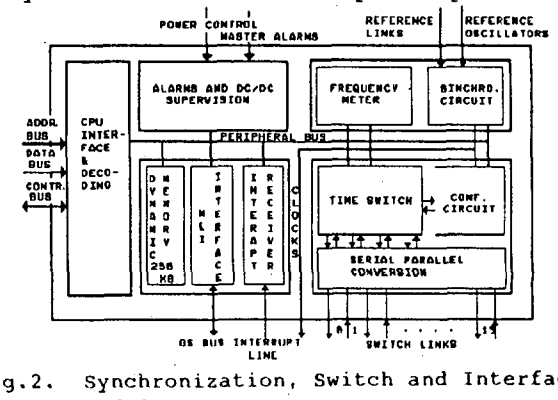


Fig.2. Synchronization, Switch and Interface module.

The standard SSI/MSI devices offer less density and consume more power than devices used in our design. They are usually manufactured in technologies with limited opportunity for further cost reductions. The PGA device, mentioned in our design, is a user programmable gate array that provides the same density and relatively short development times as gate arrays do. The PGA device combines the design and production benefits of a standard logic device with the system benefits, such as increased reliability, power savings, space savings and lower production costs, depending on whether SSI/MSI standard logic or ASIC device is used. The architecture of the PGA consists of an interior matrix of configurable logic blocks, user programmable interconnections, and I/O blocks which are partioned at the edge of the chip. The interconnections are located in channels between the rows and columns of configurable logic blocks and I/O blocks. Fig.3 shows the block diagram of the interior organization of the PGA.

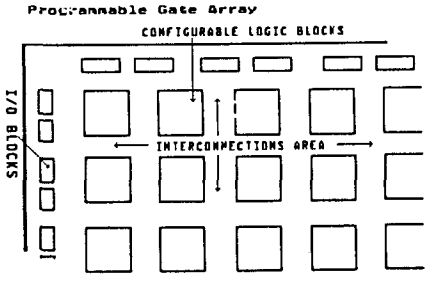


Fig.3. Inside organization of the Programmable Gate Array.

The PGA performance depends on the fixed delays of the logic and storage elements plus the interconnections delays. Each PGA device is identical until it is loaded with its specification. Our configuration bit stream is loaded automatically from a serial PROM when the power goes on. If the power goes down, the PGA loses the configuration. We require very high device density for SSI module. This means we have to put the devices very close together. We have packed our three PGA devices close together with no effect on the thermal coefficient for the expansion of the silicon.

The development support for the PGA device includes the complete software aided design entry, analysis, and verification. The development system also offers the complete basic configuration. Further activities to complete a PGA design include: schematic entry (supported with PGA library which includes common logic functions, 7400 series parts, latches,..), simulation, automatic placement and routing or automatic design implementation, timing calculations, and design optimization. The last stage in the design cycle is the in-circuit emulation.

Design cycle

The key to providing support for engineers who wish to develop a complex Programmable Gate Array is a CAD environment which enables a good application backup. Our decision was the software package OrCad that supports complex tasks to entry , to simulate and to test of the design. ORCAD/SDT is a drawing program which allows the designer to work faster with less chance of errors. It is not the simplest drawing program, but it manipulates with fully integrated software, found in the workstations. The program also supports a large library with components of standard elements, microprocessor series, TTLs, CMOS and for our requirements, very important LCA (LCA is an older term, PGA is a new one) libraries, series 2000 and 3000. We have made our own component library. This has two advantages. Firstly, when looking for a component it can be found very quickly. The second advantage is that the memory not occupied by unnecessary library parts. If you wish to design a chip using 3000 series PGA family, you should have up to 8 MB of system memory.

The most important fact is the modular way of designing. Every designer has his own favorite circuit designs being optimized by experience. The designs, made independent from each other, can be fetched from previously drawn circuits and placed into present design without any modification. In our case, a design team has worked closely together, but each designer made his own part of the design, independent from each other. At the end, the designs are put together in a modular way. The organization of our worksheets is made hierarchically. Designing in this modular way means that at the end every part of the design can be put into the present design without modifications.

The utility automatically annotates the old components. The hierarchical option is chosen instead of the single sheet drawing option to avoid the problems, appearing during the printing of the massive drawing.

Our first design is a part of a Synchronization, Switch and Interface module previously presented and shown in Fig.2 . The top level screen shows multiple sheets connected together (Fig.4).

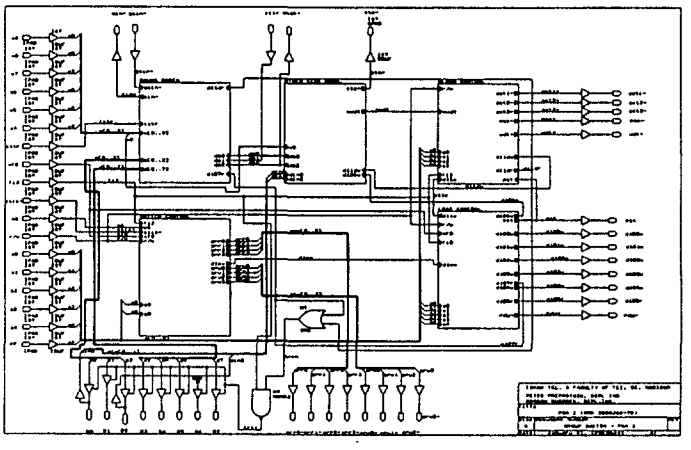


Fig.4. Complex drawing handled by partitioning in OrCad.

It presents the following five part designs: Round Robin, Alarm Control, Switch Control, Communication Control and LCA3 Control. The design, involved into the first hierarchical level is too complex to stay in one sheet. To cope with the complexity we have put our five designs into several part sheets at the second hierarchical level. Each of this designs is shown in Fig 5.

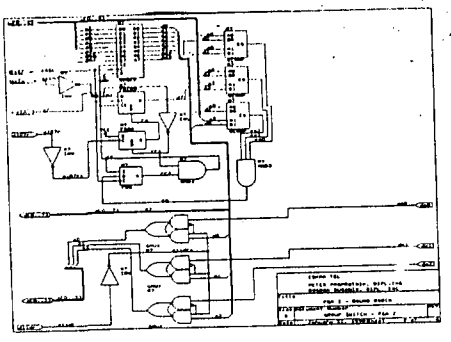


Fig.5. Detailed view of one of the five designs, globally presented in Fig.4.

As soon as the circuit is drawn, a complete part list with the list of component type, with the quantity name and the references is generated. Another command produces a list which describes every part of the diagram with all its connections to other pins. With by this command, all the interconnections in our designs are checked. Next we checked the electrical conditions. It is done by the Electrical Rules Checker. It gives us information about each pin of the part and it is used to verify if that connections conform with the electrical conditions allowed for the specific pin. For example, a bus may not be connected to a pin or two pins may not be connected together, if both of these pins are defined as output pins.

Once the designer has finished his design using the Schematic Design Tools, the design file has to be translated to the format, that is compatible with simulator and PGA design editors input. The format itself is the standard ASCII file and is called Netlist. Before the translation begins the software again automatically tests the schematic file on electrical and schematic errors. The Netlist File can also be used as the standard input for other software packages as PCB layout or Standard Cells design. For the simulation it is necessary that designer creates the stimulus for the circuit as well as the trace file where the results of the simulation are captured. The first step of simulation procedure is the functional or the logical simulation of the circuit. This type of the simulation is used primary for the logical verification of our design, which can be easily corrected by Schematic Design Software if a logical error is found. Of course if this is the case, the corrected design has to be translated again back to the Netlist, so that the simulation of the revised circuit can start again. This loop proceeds until designer is satisfied with the function of the project.

Before the second step of the simulation the final placement of our design into the Programmable Gate Array structure has to be done. The first stage is the translation from an error free Netlist File to PGA file. The translation from Netlist to PGA actually performs two operations: the logic reduction and the logic partitioning. The logic reduction is the process of deleting unused logic from a design, permitting liberal use of the macro library, without any penalty when all the functions within a macro are not used. The remaining logic is automatically partitioned into fragments that can be implemented inside individual Input Output Blocks and Configurable Logic Blocks of the PGA structure. The result is a file defined in terms of the IOB's and CLB's. This is already an actual placement of our design, however it is not convenient for use, because the placement is not optimal and connections between the blocks are not realized yet. Each CLB and IOB of the design is placed by assigning it to one of the discrete blocks within an PGA and is routed by specifying the programmable interconnect paths used to implement the connections between blocks. Since both logic blocks and programmable interconnection paths include signal delays, placement and routing choices have a major effect on the performance of the resulting implementation. It is possible to perform placement and routing interactively with the graphic based design editor, with software tool called ADI for Automatic Design Implementation or with any combination of the two.

The ADI determines the best placement for the design and then routes the nets that interconnect the blocs. An optimum placement of a design results in the shortest and the fastest interconnects with minimum possible delay. To achieve this ADI uses combination of two algorithms. Using a min-cut algorithm ADI places each block in its most desirable region of the die, then the simulated annealing algorithm completes the final placement of the block into its most near optimal location.

Another possibility to perform the placement is the graphic based design editor called Xact, which can be used interactively to edit, enter, place and route PGA designs. With the Xact design editor a user can manipulate the graphic view of the internal PGA resources directly defining the functions of CLB's and IOB's, their position in the matrix and specifying their interconnections.

More commonly both ADI and XACT design editor are used together to complete an PGA design. For most designs ADI will route all the nets in the design satisfactorily, however some designs, particularly those that use most of the internal capacity of the PGA may require completion of the routing by using interactive design editor. Xact design editor might also be used additionally to optimize critical timing paths. It is also possible to route critical portion of the design with Xact editor initially and then

use ADI to complete the placement and routing of the rest of the design.

When the place and route procedure is finished, the second step of the simulation can begin. This is the final simulation and considers the real time delay caused by placed blocks and theirs interconnections in the PGA. At this point the worst case time analysis can be done and in case of violation of the timing, the designer can improve the timing by changing the placement and interconnect paths with Xact editor, until the timing is satisfactorily.

After completion of the final simulation Design Rules Checker program is used to check for illegal configurations, such as interconnect networks with multiple sources. Another utility then generates the binary configuration program, that can be downloaded into the PGA, EPROM or serial PROM.

One major advantage of a user programmable, standard logic device such as the PGA is the ability to test the design immediately in the target application. In circuit debugging can be accomplished with in circuit verification tools such as various emulators or by programming an actual PGA device directly in the target hardware. Alternatively, the designer may debug hardware by using traditional test equipment, such as logic analysers and oscilloscopes. To estimate if a design will fit into a given PGA, the number of I/O pins, registers and the amount of logic used for the design should first be estimated, and these requirements should be matched to the number of IOB's and CLB's in the PGA respectively. Of course it should not be expected that all IOB's and CLB's of a given PGA can be utilized without some effort during placement and routing. The amount of PGA resources that can be used is dependent on the nature of the design, for example fully synchronous, logic intensive designs with low I/O intensity are the easiest to implement.

Conclusion

The task was finished in six weeks. As soon we have begun with the designing on PGAs, another team of the designers started with the designing of the new printed circuit board. The verification and the test on the new board have taken us three months respectivetily on some modifications.

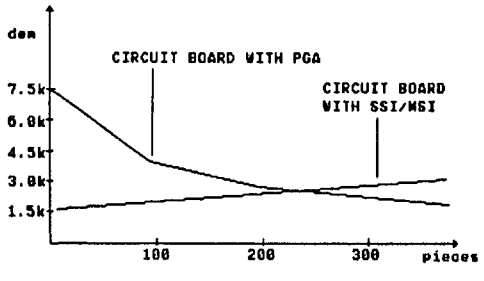


Fig.6. The approximate cost analysis for SSI board.

The costs of producing a new SSI module comparing to the old one, increased for about 6000 DEM per board. If we consider the SSI module ability to switch 128 subscriber moduls with much better time consideration, higher system security, lower power consumption, we realize that the costs for development of the new circuit can be amortized in production volumes over 270 SSI circuit boards per year. If the production exceeds the number of 1000 modules , we think we have done a good job.

References:

/1/ Fawcett B.:'User-programmable gate arrays: design methodology and development systems' Microprocessors and microsystems (5 June, 1989)

/2/ The programmable gate arrays data book AMD (1989/90)

/3/ B. Dugonik, P. Praprotnik: "Using Software Tools for modern Hardware design", Aplied Informatics '90, Insbruck (1990).

TRANSPUTERJI V SISTEMIH ZA DELO V REALNEM ČASU

INFORMATICA 3/90

Keywords: real-time systems, embedded microcomputer systems, multitasking, parallel processing, transputer, occam

Peter Kolbezen
Institut Jožef Stefan, Ljubljana
Peter Zaveršek
Gorenje Servis, T. Velenje

POVZETEK. Članek opisuje lastnosti in nekatere koncepte krmiljenja sistemov z računalnikom za delo v realnem času. Obravnava jih z vidika strojne in programske opreme. V takšne sisteme uvaja Inmosov transputer in programski jezik occam kot učinkovito metodo uporabe mikroprocesorja napram klasičnim mikroprocesorsko zasnovanim sistemom za delo v realnem času. Uporaba occama podpira izvajanje več opravil hkrati na dveh nivojih in porazdeljevanje časa na nizkem nivoju. Komunikacijski mehanizmi med asinhronimi occamovimi procesi morajo biti ustrezno prirejeni za procesiranje sinhroniziranih procesov v realnem času. Prispevek preučuje lastnosti strojne in programske opreme transputersko zasnovanih sistemov. Pokaže jih na primerih zametka takšnega sistema.

TRANSPUTERS FOR EMBEDDED REAL-TIME SYSTEMS. This paper describes the characteristcs and any control concepts of embeded real-time systems. It discusses the characteristcs and concepts in view of hardware and software. Further, the paper introduces the Inmos transputer and the programming language occam as a viable alternative to coventional methods of implementing microprocessors-based real-time systems. The present occam implementations support multitasking on two levels and time sharing on the low level. The communication mechanism between asynchronous occams processes must be matched with real-time event synchronized task processing. The hardware and software characteristics typically involved are assesed and these characteristcs are shown on the beginning of the such computer system.

1.UVOD

Krmiljenje procesov je najbolj pogosta uporaba mikroprocesorsko zasnovanih sistemov za delo v realnem času (RT- sistemov). Mikroprocesor, ki je vključen v takšen sistem, nadzoruje neke stalne zunanje aktivnosti. Strojna in programska oprema sta navadno specifični in odvisni od namembnosti sistema. Najvidnejša značilnost RT-sistema je dovolj hitro procesiranje podatkov, ki vstopajo v sistem naključno, in s tem pravočasno odzivanje sistema na okolje, kar zagotovlja pravilno delovanje celotnega sistema.

V našem prispevku bomo govorili o "sistemu za delo v realnem času", če bo izpolnjeval naslednja dva pogoja:

1. Vrstni red izvajanja opravil (računanja) je funkcija časa izvajanja ali je funkcija podatkov, ki prihajajo v računalnik (naključno) od nekih zunanjih dogodkov;
2. Rezultati delnih izračunov so lahko odvisni od vrednosti spremenljivke časa, ki jo zahteva izračun, ali od časa, ko se izvajanje prične.

Tako definirane sisteme je mogoče po klasifikaciji Civera /1/ razdeliti v natanko dve kategoriji:

1. V prvo kategorijo se uvrščajo sistemi, ki norajo imeti srednji čas izvajanja opravila - merjen preko določenega časovnega intervala - manjši od dovoljenega maksimuma.
2. V drugo kategorijo se uvrščajo sistemi, pri katerih mora biti izvajanje opravila v vsakem primeru (tudi ob nepredvidenih dogodkih) dokončano pred dovoljenim časovnim maksimumom.

Pri RT-sistemih se torej srečujemo z zahtevami kritičnega odzivnega časa. Zato mora imeti sistemska programska oprema večprocesno strukturo, ki omogoča, da se lahko vsak proces kot programska enota izvaja sočasno z drugimi procesi. Ti procesi imajo časovne omejitve in morajo običajno pri izvajanju neke namenske sistemske funkcije med seboj komunicirati in sinhronizirati svoje aktivnosti. Paralelno izvajanje procesov je lahko izvedeno s prepletanjem sestavljenih procesov na določenih časovnih intervalih (interleaving). Pri tem je uporabljen en sam procesor, ki s prepletanjem posameznih procesov ustvarja t.im. kvazi paralelizem. Tako tečejo RT-programi le navidezno vzporedno, v tako

imenovanem večopravilnem načinu (multi-tasking). Prepletajoče procese predstavljajo običajni sekvenčni programi, ki so med izvajanjem večkrat prekinjeni, da se lahko "sočasno" izvajajo tudi drugi programi. V primerjavi z običajnimi večopravilnimi programi pa vrstni red izvajanja RT-programov ni vnaprej določen. Določen je z zunanjimi signali, ki vstopajo v sistem naključno. V tistih segmentih programa, pri katerih se pričakuje zunanjo zahtevo po prekinitvi, sinhronizacija med opravili ni dopustna.

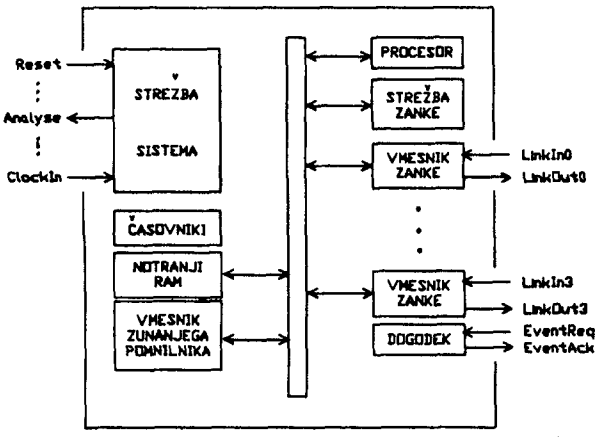
Stvarni paralelizem omogoča le večprocesorski sistem, pri katerem se izvajajo posamezni procesi ločeno, vendar sočasno, vsak na svojem procesorju.

Večina RT-sistemov vsebuje tudi dele opreme, ki ne dela v realnem času. To dejstvo ima določeno prednost, saj sta načrtovanje in implementacija programske opreme takšnih delov preprostejša.

Obravnava strojne in programske opreme mikroprocesorsko zasnovanih sistemov v realnem času bo pokazana na primerih zametka signalno informacijskega in nadzornega sistema za hotelsko službo in preskrbo. Sistem in metodologija načrtovanja sta zasnovana na Inmosovem transputerju in transputerskem konkurenčnem programskem jeziku occam. Namen tega prispevka je, da se na primeru omenjenega zametka pokaže, kako je obravnavana metodologija tako sestavljene opreme (transputer in occam) napram klasičnim metodologijam praktično uporabna in učinkovita.

2. Transputer in strojna oprema RT-sistema

Transputer je visoko zmogljiv računalnik na eni sami LSI-enoti; sestavljajo ga mikroprocesor, pomnilnik, štiri polno dupleksirane vhodno/izhodne zanke za povezovanje s perifernimi napravami in priključki za povezovanje z drugimi transputerji.



Slika 1. Zgradba transputerja

Dandanes je uporaba transputerjev že močno razširjena. O tem zgovorno priča vrsta proizvajalcev najrazličnejših transputerskih modulov. Poleg INMOSa, kot prvega proizvajalca te tehnologije, so še ParsyTec/ParaCom, MSC/Hemma, Mark Ware Associates, Gemini Computer Systems, in drugi. Module, ki so v glavnem splošno namembni, je mogoče poljubno povezovati v namenske, zmogljive, zanesljive, prilagodljive oziroma razširljive mikroračunalniške sisteme. Poleg funkcionalnosti, se moduli med seboj razlikujejo tudi po vodilih, ki omogočajo njihovo vgradnjo v sisteme z standardnimi vodili, kot so PC, Micro Channel, Q, VME, Multibus in še nekateri.

Transputer je zasnovan na tehnologiji RISC in je kot tak dovolj hiter. Neposredno podpira paralelizem, komunikacije in časovno odvisnost dosledno s koncepti konkurenčnosti komunikacijskih sekvenčnih procesov (CSP) /3/. Osnovne karakteristike transputerja kaže slika 2.

	MODEL		
PARAMETER	T212	T414	T800
Širina glavnega in medregistrskega vodila	16-bit	32-bit	32-bit
Kapaciteta RAM - pomnilnika na čip	2 KB	2 KB	4 KB
Direktno naslovljiv pomnilniški prostor	64 kB	4 GB	4 GB
Hitrost izvajanja	10 MIPS	10 MIPS	10 MIPS

Slika 2. Karakteristike transputerja

Transputerji podpirajo različne oblike in stopnje paralelnega procesiranja. S povezovanjem v večtransputerski sistem omogočajo stvarni paralelizem. V tem primeru so lahko povezani med seboj na več načinov. Glede na to so možne različne strukture večprocesorskih sistemov. Najosnovnejše so: linearna, matrična, kubna in drevesna struktura, njihova izbira pa je največkrat odvisna od namembnosti sistema.

Transputerji podpirajo tudi kvazi paralelno procesiranje. V takšnem primeru se več paralelnih procesov izvaja na enem samem transputerju s prepletanjem aktivnih rezidenčnih procesov na osnovnih časovnih intervalih. Za prepletanje skrbi mikrokodni dodeljevalnik, ki v mikrosekudah preklaplja posamezne procese in s tem ustvarja hiter kvazi paralelizem. Specifična odlika transputerskega večprocesorskega sistema je, da stopnja paralelizma in s tem učinkovitost procesiranja naraščata sorazmerno s številom transputerjev v sistemu.

Uporaba transputerja v paralelnih sistemih je preprosta. Na topologijo sistema vplivata predvsem dva faktorja: zmogljivost in cena sistema. Transputerska, jezikovno usmerjena arhitektura, primerna strojna oprema za dodeljevanje procesov, sihronizirano komuniciranje, zunanje in časovne programske prekinitve ter časovno odvisno procesiranje omogočajo učinkovite gradnike mikroprocesorskih sistemov, ki delajo v realnem času. Naštete posebnosti so v sistemu dosledno prisotne, zasnova jezika occam pa takšna, da je programiranje enostavno in da je predstavljena rešitev v tem jeziku zelo blizu predstavitvi problema.

3. Occam in programska oprema RT-sistema

Programski jezik occam je bil posebej razvit za delo v transputerski tehnologiji in podpira paralelno programiranje. Jezik je zasnovan neposredno na Hoare-ovim CSP - jezikovnem zapisu komuniciranja in konkurenčnosti /3/. Omogoča predstavitev funkcionalnih karakteristik RT-sistema in medsebojnih sistemskih aktivnosti v zgoščenem visoko-nivojskem zapisu.

3.1. Načrtovanje occam-programov

Programiranje v jeziku occam je nekoliko drugačno od programiranja v klasičnih sekvenčnih jezikih. Uporabljajo se preprosti koncepti paralelizma. Posamezne dele programa je možno zastaviti paralelno. So pa tudi deli programa, ki tečejo sekvenčno.

Takšni deli so npr. vhodno/izhodne operacije, ki so relativno manj pogoste. Zanje se uporabljajo običajna načela programiranja. Tipično se uporabljajo prilagojene tehnike, kot je "koračno očiščevanje" znotraj strukturiranega programskega okolja. Uporabo takšne tehnike omogoča na primer zložljiv urejevalnik.

Pri uporabi konceptov paralelizma pa se je treba najpreje otresti predsodkov, ki so navadno posledica programiranja v običajnih, sekvenčno naravnanih programskih jezikih. V prvih korakih načrtovanja je koristna raba že dobro vpeljanih pripomočkov. Mednje sodi diagram pretoka podatkov (Data Flow Diagram, DFD), s katerim je mogoče načrtovati ne le serijske, ampak tudi paralelne programe, saj je DFD že po svoji naravi inherentno paralelen.

V okolju occam-a uporabljamo DFD direktno, ker so tokovi podatkov enaki tokovom podatkov, ki tečejo skozi kanal. Podobno ustrezajo procesna vozlišča v DFD procesom v occam-programu. Zato DFD podaja strukturo kanalov in procesov, kot tudi tok podatkov, in s tem povezanost med procesi.

Uporaba DFD bo pokazana v naslednjih razdelkih predvsem za sistemsko analizo na tako imenovani nizko nivojski implementaciji diagramov.

Naslednji razdelek bo posvečen študiji jezikovnih posebnosti jezika occam, ki so pomembne za razvoj mikroprocesorske programske opreme za delo v realnem času. Študija je omejena na sisteme, katerih zasnova strojne opreme in arhitekture se opira na transputerski koncept. Programska oprema takšnih sistemov je nadalje obravnavana na primerih zametka signalno informacijskega in nadzornega sistema za hotelsko službo in preskrbo (HIS).

3.2. Zametek RT-sistema HIS; Zahteve, konfiguracija in značilnosti opreme

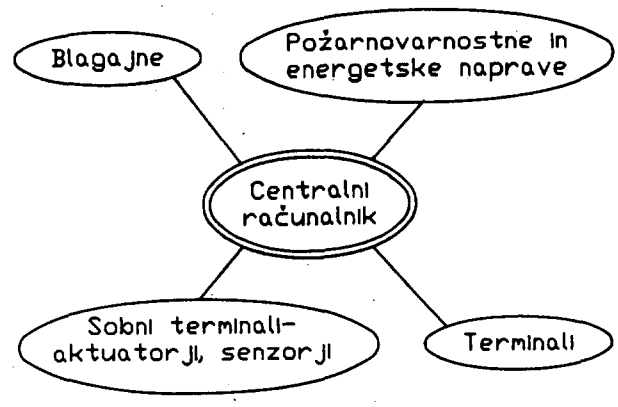
Značilnosti programske opreme sistema HIS za delo v realnem času so pogojene z asinhronim izvajanjem paralelnih procesov v sistemu. Oprema mora biti razbita na preproste, vase zaključene enote, ki se lahko izvajajo harmonično in paralelno.

Konkurenčno programiranje v jeziku occam omogoča modeliranje opravil na paralelne procese, ki komunicirajo med seboj preko sporočil. Za izmenjavo sporočil, s katerimi se procesi med seboj v celoti sinhronizirajo, se uporabljajo določeni enosmerni transputerski kanali. Ti kanali nimajo vmesniških pomnilnikov za shranjevanje sporočil. Zato se izvajanje procesa zadrži, dokler vhodno/izhodne zahteve niso izpolnjene. Na ta način je preskrbljeno za medprocesno sinhronizacijo in asinhrono izvajanje procesov na docela urejen način. Jezikovna opisa paralelizma in komuniciranja sta si podobna za enoprocesorski ali za večprocesorski sistem.

Značilnosti jezika occam so v naslednjih poglavjih pokazane z njegovo uporabo na primeru preprostega RT-okolja, ki ga predstavlja HIS. Osnovno procesno shemo HIS prikazuje slika 3a.

Centralni računalnik neprekinjeno testira vhodne informacije, ki prihajajo iz objekta naključno, in se nanje po potrebi tudi dovolj hitro odziva. To mu omogoča sposobnost paralelnega procesiranja podatkov (signalov), ki po lokalni komunikacijski mreži prihajajo iz sobnih terminalov oziroma naprav signalno-informacijskega podsistema ter iz blagajniškega, požarno-varnostnega in energetsko - klimatskega podsistema. Na centralno enoto je priključenih tudi več terminalov za interaktivno delo s sistemom. Kakšne so funkcije posameznih komponent sistema?

Signalno -informacijski, klimatski in požarno - varnostni podsistemi. V sobah in drugih prostorih hotelskega objekta so in-



Slika 3a. Osnovna shema zametka RT-sistema HIS: Okolje nadzornega sistema

stalirane posebne naprave z aktuatorji in senzorji, po želji pa tudi osebni računalniki. Preko lokalne mreže so te mikroprocesorske naprave povezane s centralnim računalnikom, kateremu posredujejo vrsto podatkov: o statusu sobe (SOS stikalo, kontrola ključavnice na magnetno kartico, protivlomna zaščita, klic sobarice, servisa, ...), o klimi in požarni zaščiti (aktuatorji, senzorji). Omenjene naprave v obratni smeri, torej od centralnega računalnika sprejemajo signale oz. sporočila za gosta (signal prisotnosti, bujenja, sporočila na recepciji), za nadzor klime in požarne-varnosti, itd. Podsistem omogoča tudi računalniško komuniciranje iz "vseh" sob objekta z drugimi računalniki, ki so priključeni v objektu na centralno enoto HIS preko lokalne mreže (LAN) ali izven njega tudi preko javnih mrež.

Blagajniški podsistem. Gostje in osebje ima namesto ključa magnetno kartico za vstop v prostore. To kartico je smiselno uporabiti tudi za registracijo prisotnosti in kot kreditno kartico znotraj hotelskega kompleksa (trgovina, bar, restavracija, športni objekti). Zato je potrebna sprotna evidenca o veljavnosti kartice, porabljenih finančnih sredstev gosta in dr. Na ta način je preskrbljeno za bolj varno in ažurno poslovanje, za gostu prijetnejše bivanje ter za brezgotovinsko naplačevanje in poravnavo računov v kateremkoli trenutku.

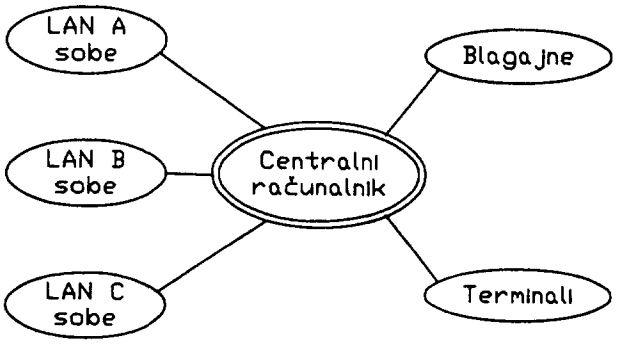
Terminali. Osrednji računalnik mora biti povezan z več terminali. Ti so npr. na recepciji za evidentiranje gostov in za prikaz zasedenosti sob, vodenje rezervacij ipd., pri gospodinjah, sobaricah in najrazličnejših servisih za evidenco stanja sob (čiščenje, menjava posteljnine, prisotnost, klici) in za druga opravila. Terminal naj bi bil prisoten tudi pri varnostniku in omogočal pregled nad stanjem protivlomnih, protipožarnih in podobnih naprav.

Komunikacijska mreža. Za realizacijo komunikacijske mreže je izbrana TV-napeljava, saj je danes prisotna že v vsakem objektu. Na ta način se bistveno zmanjšajo stroški celotnega sistema, način pa omogoča tudi lažjo vgradnjo sistema HIS v že obstoječi hotelski kompleks. Ob prisotnosti kabelskega TV (CATV) omrežja na širšem področju (mesto, turistično naselje, itd.) predstavlja tako omrežje še posebej zanimiv komunikacijski medij v večjem informacijskem sistemu.

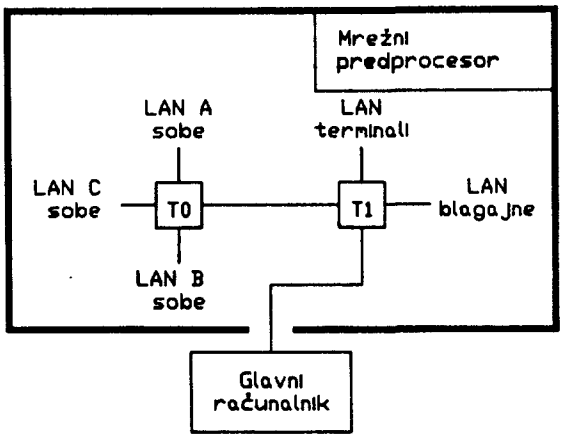
3.3. Vrstni red in časovni potek izvajanja

Zaradi lažje nadaljne obravnave prikazuje slika 3b paralelno procesno shemo (iz slike 3a) bolj poenostavljeno. Poenostavljena shema na sliki 3b ima le dva podsistema, hkrati pa je bližja realnemu svetu, saj je lokalna mreža razdeljena na več paralelnih mrež.

Centralni računalnik mora biti dovolj zmogljiv in primeren za delo



Slika 3b. Paralelna procesna shema sistema HIS



Slika 3c. Mrežni predprocesor

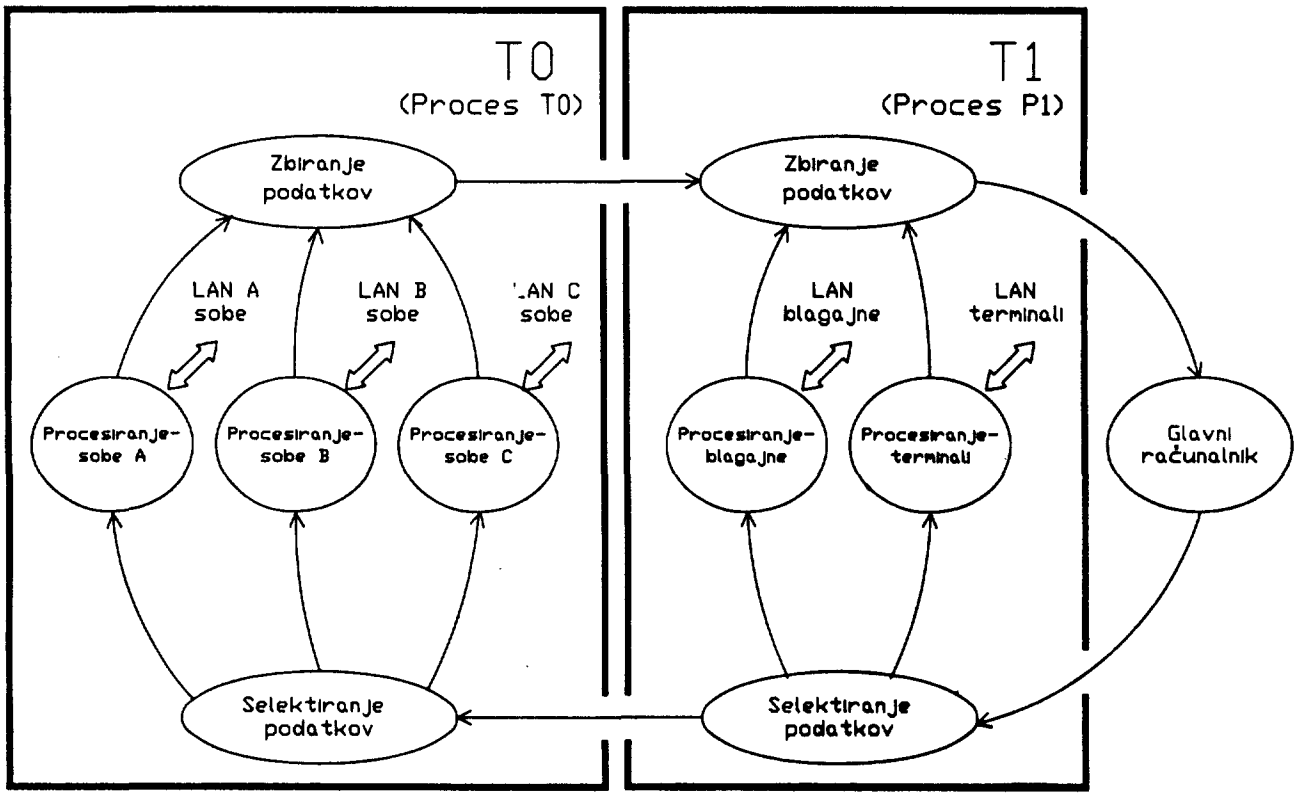
v realnem času. Zato je sestavljen iz glavnega računalnika in predprocesorja za neposredno vodenje podsistemov HIS. Predprocesor, ki opravlja tudi funkcijo krmilnika lokalnih mrež sistema, je realiziran s transputerji.

Predprocesor v transputerski izvedbi z dvema transputerskima modulama T0 in T1 je prikazan na sliki 3c. Iz slike je razvidno, da sedaj transputerji upravljajo celotno komunikacijo, od glavnega računalnika pa prevzemajo tudi del pomembnih funkcij podsistemov, kot so npr. kontrola magnetnih kartic, sortiranje podatkov, oddajanje signalov v mrežo za indikacijo odziva sistema, sortiranje podatkov iz sprejetega paketa, in podobno.

Če se procesi izvajajo na enem samem transputerju, potem število kanalov med njimi ni omejeno. To so t.im. programski kanali (soft channels). V primeru, da se procesi, ki med seboj komunicirajo, izvajajo na različnih transputerjih, je število kanalov omejeno na štiri fizične zanke (hard channels, strojni kanali). Če prihajajo preko ene zanke podatki iz več procesov, je potreben multipleksiran prenos podatkov. Ta način je možen in sprejemljiv, če hitrost prenosa podatkov ni kritična. Procesom, ki zahtevajo hitrejšo izmenjavo podatkov z okolico, pa mora biti zagotovljena posebna zanka. Iz slike 3c je razvidno, da imata oba modula (T0 in T1) v skladu s to zahtevo po štiri dvosmerne zunanje povezave z okolico.

Iz diagrama na sliki 4 je opazna skladnost povezav med procesi P0 in P1 v transputerskem okolju s povezavami med transputerji T0 in T1 na slīki 3c.

DFD lahko direktno preslikamo v occamski program. Vozlišča v



Slika 4. DFD transputerskega okolja

diagramu predstavljajo procese, povezave med vozlišči (tj. med procesi) pa predstavljajo komunikacijske poti za medprocesno izmenjavo podatkov. V occamu so komunikacijske poti direktno occam-kanali, po katerih si procesi med seboj izmenjujejo sporočila. Tako DFD natančno podaja število kanalov in povezavo kanalov s posameznimi procesi, s tem pa je v occamu podana tudi medsebojna povezanost procesov.

V jeziku occam posplošena procesna struktura in sinhronizacija procesov, zasnovana na sporočilih, omogočata hierarhično dekompozicijo programske opreme v množico komunikacijskiskih paralelnih procesov nad lokalno deklariranimi podatki.

Programi v jeziku occam so v osnovi sestavljeni iz treh primitivnih procesov, ki so označeni s spremenljivko ter vhodom in izhodom kanala. Pri tem se uporabljajo jezikovni konstrukti PAR, ALT in SEQ, ki določajo strukturirane procese. Tudi procesi zbiranja in selektiranja oz. predaje podatkov na transputerju, kot je razvidno iz DFD na sliki 4, imajo ustrezne značilnosti paralelnega, alternativnega ali serijskega izvajanja. Izpis 1 podaja occam-program procesa zbiranja in predajanja podatkov skupaj z glavnim programom. Prvi proces je imenovan 'producer', drugi pa 'consumer'.

```
--  zbiranje/predaja podatkov
--  definicija procesa zbiranja
PROC producer ( CHAN s.comms )
  CHAN OF INT input :
  PLACE input AT 2 :
  VAR x :
  WHILE TRUE
    SEQ
      input ? x
      s.comms ! x
:

--  definicija procesa predaje
PROC consumer ( CHAN d.comms )
  CHAN OF INT output :
  PLACE output AT 1 :
  VAR y :
  WHILE TRUE
    SEQ
      d.coms ? y
      output ! y
:

CHAN comms :
PAR
  producer ( comms )
  consumer ( comms )
```

Izpis 1. Program zbiranja in predajanja podatkov

Pri multipleksiranem zajemanju prihajajo podatki od distribuiranih procesov s pomočjo vektorskih kanalov d.c. Vrstni red, v katerem se pojavljajo, ni določen. Vhodi se zato izbirajo nedeterministično. Pri tem se uporablja konstrukt ALT, ki je zapisan takole:

```
ALT
  d.c.[0]?val
      :
  d.c.[1]?val
      :
  d.c.[2]?val
```

Deterministično izvajanje paralelnih in alternativnih komponentnih procesov na istem transputerju v časovno kritičnih odsekih kode se lahko določi z uporabo konstruktov PRI PAR in PRI ALT, ki označujejo prioritete izvajanja posameznih komponentnih procesov. Izpis 1 prikazuje program s konstruktom PRI ALT, ki predstavlja proces prikazovanja poziva (proces I) ob prekinitvi opravila "povpraševanje".

```
-- Proces I
--
WHILE TRUE
    CHAN OF BYTE povprasevanje.tipalo:
    CHAN OF INT lokacija.zaslon:
    INT parameter1.vrednost,....,parameterN.vrednost:
    BYTE povprasevanje:
    PRI ALT
      povprasevanje.tipalo ? povprasevanje
        IF
          povprasevanje = '1'
            SEQ
              parameter1.povprasevanje ! 'Y'
              parameter1.kanal ? parameter1.vrednost
              vrednost.zaslon ! parameter1.vrednost
            .
            .
            .
          povprasevanje = 'N'
            SEQ
              parameterN.povprasevanje ! 'Y'
              parameterN.kanal ? parameterN.vrednost
              vrednost.zaslon ! parameterN.vrednost
          TRUE
            SKIP
        TRUE & SKIP  --  prekinitev povprasevanja
                     --  po parametru1/.../parametruN
```

Izpis 2. Prikazovanje poziva na zaslonu

Vsak proces je mogoče opisati z osnovnimi procesnimi strukturami:

Poleg paralelizma je bistvena lastnost RT-programa časovna odvisnost posameznih aktivnosti, ki so s programom določene in morajo biti sinhronizirane z neko dimenzijo časa. S tem je zagotovljena časovno odvisna strežba komponent, ki so priključene na sistem. Lastnost, ki jo zato morajo imeti RT-jeziki je, da dopuščajo dostop do časovnih posebnosti, ki so povezane npr. z urnim mehanizmom procesorja, in da imajo konstrukte, ki omogočajo jedrnato specifikacijo časovnih operacij.

Programiranje v jeziku occam, ki je zasnovano za delo v realnem času, je glede na gornjo zahtevo mogoče podpreti s konstrukti, ki uporabljajo poimenovane objekte tipa TIMER. Uporabljajo se za odčitavanje transputerskega časovnika. Konstrukti so izraženi v obliki posebnih operacij vhodnih kanalov. Vhod iz transputerskega časovnika predstavlja integer vrednost. Glavne oblike časovnih operacij, ki jih določa jezik occam, so naslednje:

```
TIMER timer:
INT time, period:
--
--  proces casovnega vhoda
--
  timer ? time
        --
```

```
        -- vrednost spremenljivke
        -- casa, ki jo sproti doloca
        -- transputerski casovnik
--
--  proces casovnega izhoda
--
  timer ? AFTER time
        --
        -- vhod iz transputerskega
        -- casovnika je zadrzan,
        -- dokler njegova vsebina
        -- ne preseze doloceno
        -- vrednost; od tega
        -- trenutka dalje so
        -- operacije brez ucinka
--
-- zakasnitveni proces
--
SEQ
  timer ? time
  timer ? AFTER time PLUS period
        --
        -- vhod iz transputerskega
        -- casovnika je zadrzan,
        -- dokler njegova vsebina
        -- ne preseze doloceno
        -- vrednost (zakasnitev)
        -- (cas + perioda)
```

Izpis 8. Oblike časovnih operacij v occam-u

Časovni konstrukti jezika occam omogočajo predikatne RT - operacije, ki so preprosto določljive. Število časovnih objektov, ki jih program dopušča in uporabljajo isti transputerski časovnik, je neomejeno. Zato ima lahko vsak proces svoje časovne zahteve in uporablja lokalno deklarirane časovne objekte. Predstavitev sistemskih funkcionalnih delov in njihovih časovno odvisnih operacij je tako dovolj nazorna.

3.4. Programiranje na nižjem nivoju

Jeziki za programiranje sistemov, ki delajo v realnem času, morajo omogočati lažje programiranje tudi na nižjem nivoju in s tem možnost, da se programi čim lažje prilagajajo spremembam v konfiguraciji sistema. Ta zahteva izhaja iz dejstva, da se strojna oprema, ki je pridružena RT-sistemom, pogosto menja in je pogosto nestandardna. Klasična rešitev je, da se v takšnih primerih v visoko-nivojski program vključujejo strojni kod ali rutine v zbirnem jeziku.

Jezik occam je za transputersko arhitekturo zaprt. Zato se v splošnem nizko-nivojsko programiranje transputerskih sistemov ukvarja s preslikavami na ciljno sistemsko strojno opremo z naslednjimi specifikacijami:

1. Prirejanje virov
2. Proces - transputersko dodeljevanje
3. Deklariranje vrat; Preslikave pomnilnika na vhodno / izhodna vrata

(1) Prirejanje deklariranih procesnih kanalov zankam strojne opreme vmesnika in spremenljivk, kot tudi zunanjih vhodno / izhodnih vrat, fizičnim pomnilniškim naslovom, omogoča konstrukt PLACE .. AT in je izražen takole:

```
PLACE parameter.tipalo AT 0:
    -- dodelitev kanala
    -- parameter.tipalo
    -- k vmesniku zanke 0

PLACE status.register AT #1001:
    -- dodelitev zunanjih vrat
    -- status.register k
    -- pomniliski lokaciji 4097
```

Poimenska prireditev v programski opremi bo v času izvajanja omogočala dostop do specifične strojne opreme. Vsekakor morajo biti vse prireditve poenotene; zanki vmesnika je lahko prirejen en sam kanal.

(2) Dodeljevanje paralelnih procesov posameznim transputerjem v večtransputerskem sistemu omogoča konstrukt PLACED PAR, ki nadomešča konstrukt PAR pri ustreznem tekstualnem nivoju v programih. Temu sledi procesorski stavek (PROCESSOR statement), ki vsebuje številko transputerja in nato pridružen proces, ki se na transputerju izvaja.

Konstrukt PLACED PAR za procesorje T, ki so označeni z zaporednimi številkami 0, 1 in 2, ter za njim prirejene procese P0, P1 in P2, je:

```
PLACED PAR
  PROCESSOR 0   -- transputer T0
    SEQ
      ...       -- deklaracije
      ... P0    -- opis procesa 'sobe (A,B,C),
                -- povezava s T1'
  PROCESSOR 1   -- transputer T1
    SEQ
      ...       -- deklaracije
      ... P1    -- opis procesa 'terminali,
                -- blagajne, povezava s T0
                -- in glavnim racunalnikom'
```

Zgoraj opisano proces-transputersko konfiguracijo in komunikacijske poti v sistemu prikazuje slika 3c.

Iz gornjega programskega konstrukta je razvidno, da imamo dva procesa (P0, P1), ki se nahajata vsak na svojem procesorju T (T0, T1). V deklaraciji opišemo potrebne kanale, razporeditev strojnih kanalov po linkih in spremenljivke. Enako velja tudi za deklaracije v naslednji primerih zapisa posameznih procesov:

Pogled v proces P0:

```
-- proces P0
PAR
  ... hotelska mreza
  ... povezava s T1
```

Transputer T0 podpira hotelsko mrežo, ki je razdeljena na tri paralelne dele, in je namenjena komunikaciji s sobnimi terminali in drugimi napravami (kot so aktuatorji, senzorji), s terminali za sobarice in za ostale servisne in nadzorne službe. Povezava s T1 je namenjena sprejemanju sporočil, ki jih daje proces P1 (na transputerju T1), in po ustrezni obdelavi na T0 za pošiljanje podatkov nazaj k T1. In podobno, za sprejemanje in oddajanje sporočil v primerih, ko se obdelujejo podatki iz T0 na T1.

Načelna oblika okrnjenega programa za lokalno mrežo s štiremi vzporednimi vejami (LAN A, B in C) je naslednja:

```
-- glavna mreza
PAR [i = 0 FOR 3]     -- za tri veje
  SEQ                 -- zaporedno
    ...               -- deklaracije
    WHILE NOT end     -- ista veja za sobe
      SEQ             -- in sobarice
        SEQ [k = 0 FOR st.sob]
                      -- za vse sobe in
          ... sobni racunalnik
                      --
        SEQ [k = 0 FOR st.sobaric]
                      -- za vse sobarice
                      -- v veji
          ... terminal sobarice
                      --
```

V posameznih vejah se najprej sekvenčno naslavlja in procesira sobne računalnike, nato še terminale sobaric.

Proces T1 je podobno zastavljen:

```
-- proces T1
PAR
  ... blagajne
  ... terminali
  ... povezava s T0
  ... povezava z glavnim racunalnikom
```

Kvazi paralelno se procesirajo mreže za blagajne, terminale (recepcija, varnostnik, ...) ter za komunikacije med procesorjem T0 in centralnim računalnikom.

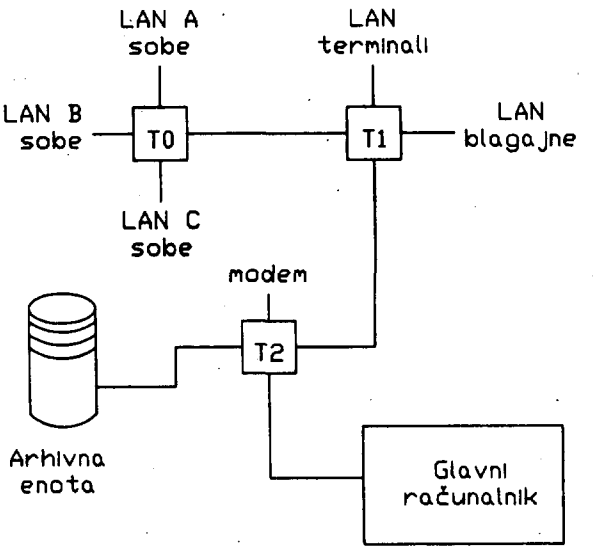
Za terminale, ki so na isti mreži, se postopek odvija sekvenčno:

```
-- terminali
SEQ
  ...
  WHILE NOT end
    SEQ  -- na isti mrezi
      ... recepcija
        --
      ... varnostna sluzba
        --
```

V primeru, da se število povezav z okoljem poveča, ali da dano polje procesorjev ne more dovolj hitro opraviti svoje naloge, je mogoče konfiguracijo sistema po potrebi razširiti. Mnogokrat se poišče takšna rešitev, da lahko ostanejo že napisane programske rešitve popolnoma nespremenjene. Takšna rešitev je možna pri razširitvi sistema z arhivno ali rezervno pomnilniško enoto in modemom za povezavo z zunanjim svetom v konfiguraciji, ki jo kaže slika 5.

Pri razširitvi na sliki 5 se v sistem vključuje dodatna procesorska enota (T2). Vsi procesi, ki se izvajajo na T0 in T1, ostanejo skoraj v celoti nespremenjeni. Konstruktu PLACED PAR je dodan procesor T2 z ustreznim procesom P2, nekoliko pa se spremeni tudi proces P1.

```
PLACED PAR
  PROCESSOR 0   -- transputer T0
```



Slika 5. Razširjena konfiguracija HIS

```
  SEQ
    ...          -- deklaracije
    ... P0       -- opis procesov 'sobe (A,B,C)
                 -- in povezave s T1'
  PROCESSOR 1    -- transputer T1
    SEQ
      ...        -- deklaracije
      ... P1     -- opis procesov 'terminali,
                 -- blagajne in povezav s T0
                 -- in T2'
  PROCESSOR 2    -- transputer T2
    SEQ
      ...        -- deklaracije
      ... P2     -- opis procesov 'povezav
                 -- s T0, T1, arhivno enoto
                 -- in glavnim racunalnikom'
```

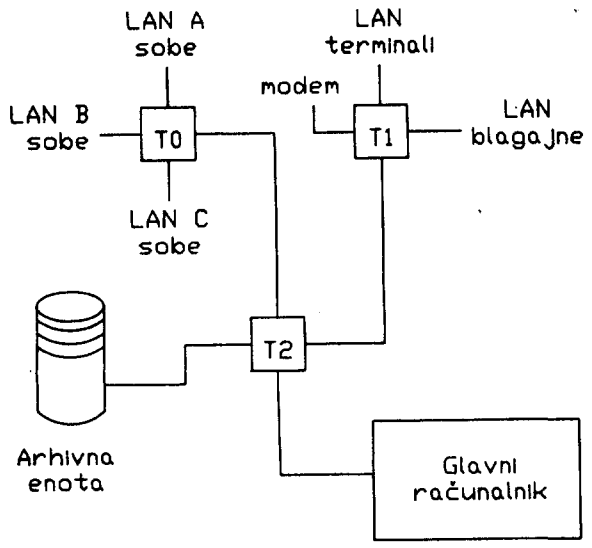
V primeru na sliki 6, kjer ima razširjeni sistem HIS (iz slike 5) spremenjeno topologijo, so potrebne že večje spremembe.

Kot je razvidno iz gornjih izvajanj, je s kombinacijo konstrukta PLACED PAR in konstrukta PROCESSOR možno direktno dodeljevanje posameznih procesov določenim transputerjem. S tem dodeljevanjem in skupaj s prireditvenimi konstrukti PLACE .. AT .. je določen celoten dostop do glavnih virov večtransputerskega sistema.

Obravnavani konstrukti jezika occam omogočajo razvoj paralelnih occam programov, ki tečejo na enem samem transputerju. Odlika takšnih programov je, da se lahko hitro - največkrat le z minimalnim reprogramiranjem - prenašajo iz sistema, za katerega so bili načrtovani, na sisteme z drugačno topologijo.

(3) Priključitev perifernih naprav (terminalov, tiskalnikov, ali v našem primeru tudi priključitev transputerja T2 na glavni računalnik) za prenos podatkov v obliki sporočil, je potrebno določiti vhodno/izhodna vrata takole:

```
PORT OF BYTE external:
     -- z 'external' so deklarirane
     -- vhodno/izhodna vrata tipa BYTE
```



Slika 6. Razširjen sistem HIS s spremenjeno topologijo

```
PLACE external AT #No:
     -- priredi vhodno/izhodnim vratom
     -- zunanje naprave pomnilniski
     -- prostor v T z naslovom No
```

Pomnilniške preslikave vhodno/izhodnih vrat se uporabljajo podobno kot preslikave occam kanalov pri prenosu podatkov. Toda - v nasprotju s kanalsko zasnovanim komuniciranjem-pri pomnilniških preslikavah vhodno/izhodnih vrat ni nobenega sinhronizacijskega mehanizma. Zato morajo biti vmesniške naprave, ki skrbijo za pomnilniško preslikovanje prenosov, programirane tako, da so upoštevane operacijske karakteristike perifernih naprav. To omogočajo jezikovni konstrukti v occamu, ki olajšujejo nizko - nivojsko upravljanje z bitno usmerjenimi logičnimi operacijami in s pomiki ali rotacijami bitnih nizov v levo ali desno. Ustrezne kode kažejo tipično asinhrono naravo pomnilniških preslikav vhodov / izhodov.

Occamovi jezikovni pripomočki za neposredno prirejevanje virov, prilagodljivo dodeljevanje procesov in neomejeno konvencionalno pomnilniško preslikovanje vhodov/izhodov bistveno olajšujejo načrtovanje RT-sistemov. Pri tem je pomembno poudariti, da se pri uporabi jezika occam ni potrebno spuščati na nižji nivo programiranja, kot je običajno pri drugih jezikih.

3.5. Obdelava napak

Posebej za sisteme, ki delajo v realnem času, je pomembno, da njihova programska oprema obravnava tudi napake. Takšna oprema zagotavlja sistemsko zanesljivost in trdoživost do napak, kar je še posebej pomembno za sisteme z računalnikom (embedded systems). Ob morebitni napaki se mora sistem po obdelavi napake vsakokrat hitro vrniti nazaj v pravilno stanje in normalno nadaljevati delo.

Occam ne podpira formalnih mehanizmov za obravnavo napak, medtem ko jih ADA podpira. Occamovo programsko opremo za odkrivanje napak v času izvajanja programa je mogoče opredeliti v dve skupini:

1. Detektiranje napak s prevajalnikom
2. Detektiranje napak pri izvajanju programa

Pri programski opremi iz prve grupe, se napake odkrivajo na več načinov: s strojno opremo, s preverjanjem instrukcij, ki se vključujejo s prevajalnikom, s transputerskimi zastavicami za napake in z izhodi, na pinu katerega se ob napaki pojavi signal. Napake, ki se lahko odkrivajo so: aritmetična prekoračitev (obsega), prekoračitev polja in deljenje z 0. Sistem je prirejen za obdelavo takšnih napak po eni od naslednjih poti:

- Proces z napako se ustavi, ostali procesi normalno tečejo dalje.
- Transputer, pri katerem se pojavi napaka, se ustavi, ali se ustavi večprocesorski sistem v celoti, in zaščiti notranje stanje sistema za nadaljne analize.
- Vključi se univerzalni proces obdelave napake. Ta pričakuje vhod, ki je pogojen s pinom EventReq na zahtevo pina Error. Pri tem se uporablja strojni kanal, ki dovoljuje vhod signala na pinu EventReq in preko katerega poteka pošiljanje sinhronih sporočil za procesiranje napak. Proces se izvaja podobno kot drugi occam-ovi procesi in je podvržen istim pravilom. Zato mora biti occam-kanal združen s pinom EventReq; deklaracija takšnega kanala, uporabljenega za obdelavo napak, je opisana v naslednjem zapisu:

```
PLACE event AT No:
...
WHILE TRUE
  SEQ
    event ? signal
      -- vhod je zakasnen, dokler
      -- ni pogojen s pinom EvenReq
      -- na zahtevo pina Error

         Proces obdelave podatka
```

  Proces, v katerem se pojavi napaka, se zadrži v stanju mirovanja. Posledica napake se odpravlja tako, da drugi procesi prevzamejo aktivnosti tistih procesov, ki postanejo nefunkcionalni. Zaradi centalizirane narave procesiranja napake, morajo biti dodatni koraki zelo nezahtevni za izvajanje ali dodeljeni drugim procesom tako, da se osnovni proces obdelave napake lahko nadaljuje in hitro odzove morebitni ponovni napaki.

Pri programski opremi iz druge grupe se napake odkrivajo z vgrajenim, eksplicitno programiranim preverjanjem s pravimi vrednostmi in rezultati. Tako se odkrivajo napačni podatki, zatajeni pogoji in podobno.

Večja zanesljivost komunikacij, ki uporabljajo transputerske vmesniške zanke, je dosežena s podporo številnih standardnih occam - procedur. Za uspešen prenos preko vmesniške zanke skrbita standardni proceduri InputOrFail in OutputOrFail. Proceduri pokažeta, če se je prenos, ki se je aktiviral, izvršil ali ne. Strojna oprema uporabljene zanke, na kateri se je med komuniciranjem pojavila napaka, mora biti ponovno inicializirana. Inicializacijo kanalov na obeh koncih zanke izvrši standardna procedura Reinitialise.

Možnost obdelave napak in trdoživost proti napakam je lastnost transputerskih sistemov, ki običajno zahteva zapleteno diagnostiko napak in ustrezno, dokaj zahtevno programsko opremo.

Visoka stopnja zanesljivosti, sposobnost reševanja vprašanja redundance strojne opreme in dinamične rekonfigurabilnosti sistema zagotavljajo pravilno delovanje tudi v primerih morebitnih kritičnih napak v strojni opremi. V splošnem vplivajo na strukturo transputerske strojne in programske opreme, ki omogočata obdelavo napak, značilnosti sistema, ki so pogojene z njegovo uporabo.

4. Zaključek

Bistven pomen prispevka naj bi bil, da se na konkretnem primeru zametka signalno - informacijskega in nadzornega sistema za velike hotelske oziroma turistične objekte pokažejo nekatere posebnosti in prednosti uporabe transputerske programske in strojne tehnologije pri snovanju RT-sistemov z vgrajenimi računalniki.

Uporaba sistemov v realnem času je navadno zasnovana na običajnih računalnikih. Ti so sestavljeni iz enega samega ali večih procesorjev, ki si dodeljujejo pomnilnik in vodila. Z dodeljevanjem virov se pojavljajo zapleteni problemi, ki morajo biti rešeni, praviloma pa vodijo k manjši zmogljivosti sistema.

Transputersko zasnovana oprema, ki kaže v visoko zmogljivost in modularnost, je posebej primerna za asinhrono izvajanje opravil v RT-sistemih, saj omogoča resnično in kvazi paralelno procesiranje. Možnost lahkega in učinkovitega oblikovanja večprocesorskih transputerskih sistemov, pri katerih predstavlja dodeljevanje pomnilnika in vodil manjši napor, daje transputerju bistveno prednost pred običajnimi računalniškimi sistemi, ki se uporabljajo v RT-sistemih. Kljub temu, da so v transputerskih sistemih medprocesorske komunikacije omejene s štirimi fizičnimi povezavami, obstajajo velike možnosti pri izbiri optimalne računalniške topologije in v strategiji uporabe ustrezne programske opreme. Oboje skupaj lahko odločilno vpliva na zmogljivost sistema.

Jezik occam ima preprosto, regularno sintakso, ki omogoča jasen in naraven razvoj programske opreme paralelnih transputerskih sistemov. Jezik je neposredno uporaben tako za programiranje RT-sistemov, kot za predstavitev paralelnih procesov in časovnih potekov, za dostop do nizkega nivoja stroja in za prilagodljivo priključevanje perifernih naprav. Pomembno je tudi, da omogoča programiranje, ki minimizira prekomeren čas izvajanja in skrbi za optimalnejšo izrabljenost pomnilniškega prostora. V jeziku occam zasnovano programsko opremo je mogoče dopolnjevati s konstrukti jezika C, Pascala in Fortrana. Ti jeziki namreč podpirajo nekatere dodatne programske konstrukte in podatkovne strukture, ki so lahko ugodnejše od onih, ki jih nudi occam. Vidna prednost jezika occam napram ostalim tovrstnim jezikom pa je vsekakor ta, da omogoča razvoj programov, ki opisujejo tudi v sistemu uporabljene komponente strojne opreme. Zato je jezik uporabljiv tako za formalni opis rezultatov načrtovanja, kot za opis dejanske končne programske opreme sistema. Na ta način je možno bistveno zmanjšati ceno in čas načrtovanja.

Kadar se transputer in occam uporabljata vzajemno, je učinkovitost implementacije transputerja velika. Tako načrtovani RT-sistemi se po svojih zmogljivostih uvrščajo med najzmogljivejše "embedded" krmilnike, ki se uporabljajo za reševanje zahtevnih nalog procesiranja slik in signalov. Zaradi omenjenih lastnosti in cene transputerske tehnologije, ki stalno pada, predstavljajo transputerji pomembno alternativo pri izbiri mikroprocesorjev za sisteme, ki delajo v realnem času.

5.Literatura

/1/ Civera,P.,Del Corso,D. and Gregoretti,F., "Microcomputer systems in real-time applications" in Tzafestas,S.G.(ed.), Microprocessors in Signal Processing, Measurement and Control, Reidel

/2/ Inmos Ltd. Transputer Reference Manual (1986)

/3/ Hoare,C.A.R., "Communicating sequential processes", Communications ACM, Vol.21, No.8 (1978), pp.666-677

/4/ Burns,A., Programming in Occam 2, Addison-Wesley (1988)

/5/ Hull,M.E.C., and Zarea-Aliabadi,A., "Embedded microcomputer systems - a comparative study of two implementation methods", J. of Microcomputer Aplications, Vol.10 (1987), pp.255-281

/6/ Bennett,S., Real-Time Computer Control: an introduction, Prentice Hall (1988)

/7/ Kerridge,J., Occam programming: a practical approach, Blackwell Scientific Publications (1987)

/8/ Mihovilović,B.,Kolbezen,P.,Šilc,J., "A paradigm of transputer system implementation", Informatica 4/87, (1987) pp.54-58

/9/ Jereb,B., Pipan,L., Klofutar,A., "Transputers", Informatica 1/89 (1989) pp.43-47

SIMULACIJA IN VREDNOTENJE PROGRAMSKE OPREME ZA VEČPROCESORSKI RAČUNALNIK

INFORMATICA 3/90

Keywords: parallelism, multiprocessor system, interprocess communication, simulation, granularity

Bojan Čukić
Fakulteta za elektrotehniko in računalništvo, Ljubljana

P O V Z E T E K

Razvoj vse hitrejših in zmogljivejših računalnikov gre danes v smeri izkoriščanja principov paralelizma pri snovanju novih računalniških arhitektur. Nove arhitekture pogojujejo tudi spremembe principov programiranja. Večprocesorski računalniki se danes programirajo na dva načina. Prvi način je programiranje v standardnih sekvenčnih programskih jezikih. Tak program se prevede s posebnim, vzporednim prevajalnikom, ki z več ali manj uspeha prilagodi kodo za vzporedno izvajanje. Drugi način, ki zahteva popolnoma nov pristop k snovanju algoritmov, je programiranje z eksplicitno vzporednimi jeziki.

V članku je predstavljen koncept za simuliranje izvajanja programov pisanih v eksplicitno vzporednem programskem jeziku PARSYS Pascal za večprocesorski računalnik. PARSYS Pascal je razširjeni ISO Pascal z "ukazi" za kreiranje procesov, medprocesno komunikacijo, skupnimi in lokalnimi spremenljivkami itd. "Ukazi" so v bistvu klici procedur in funkcij simulatorja,ki se v času prevajanja dodajo vsakemu PARSYS Pascal programu. Poleg primitivov programskega jezika vsebuje simulator tudi procedure in funkcije, s katerimi simulira izvajanje vzporednega programa na večprocesorskem sistemu.

Časovni rezultati simulacij so dobra osnova za učenje principov dobrega in učinkovitega vzporednega programiranja.

ABSTRACT: SIMULATION AND PERFORMANCE EVALUATION OF PARALLEL SOFTWARE OF MULTIPROCESSOR SYSTEM

The development of ever faster and more capable computers is aimed at the application of the principles of the parallelism in the construction of new computer architectures. New computer architectures cause in turn changes in the programming principles. The multiprocessing systems are programed now in two ways. The first involves standard sequential programming languages. Such a program is compiled by a parallel compiler which with varying success adapts a program code to parallel execution. The second way, requiring a completly new approach to the creation of algorithm, involves a programming with explicit parallel language.

The article presents a concept of the simulated execution of a program which is written in exsplicit parallel programming language PARSYS Pascal for multiprocessor. The PARSYS Pascal is an ISO Pascal enlarged with "statements" for the process creation, interprocess communication, local and shared variables, etc. The "statments" are actually calls of procedures or functions which during the compile time are added to each PARSYS Pascal program. Apart from the primitives of the programming language the simulator contains procedures and functions by which it simulates the execution of the parallel program on the multiprocessing system.

The results of the simulations provide good basis for those who wish to learn the principles of a good and efficient parallel programming.

1. U V O D

Čeprav smo priča sila naglemu razvoju računalništva in informatike, vidimo, da je hitrost naraščanja potreb po močnejših računalnikih večja, kot pa razvoj tehnologije. Rešitve potreb po močnejših sistemih iščemo na različne načine in sicer:

- skozi novo tehnologijo: stopnja integracije vezij je vedno večja, uvajajo se novi materiali kot na primer GaAs
- preko novih programskih prijemov: metode umetne inteligence, ekspertni sistemi, programska orodja
- z novimi arhitekturami : paralelne arhitekture v računalništvu omogočajo paralelno izvajanje nalog.

Računalniški sistemi z novimi arhitekturami imajo zelo pomembno vlogo pri razvoju novih konceptov in rešitev v računalništvu. Nekateri paralelni sistemi, kot na primer računalnik

Butterfly, IBM RP3, Cedar in drugi so pokazali izredne možnosti paralelnih računalnikov.

Tuje konzultantske firme napovedujejo da bo sredi 90. let 40% trzišča zmogljivih računalnikov temeljilo na principih paralelnega procesiranja.

Cilj mojega dela je predstavljanje sprememb v načinu programiranja, ki so posledica novih računalniških arhitektur. Očitno je da bo programiranje paralelnih računalnikov k malu postalo del vsakdanjega življenja vseh, ki se ukvarjajo z računalništvom.

2. PROGRAMIRANJE PARALELNIH RAČUNALNIKOV

Do danes je razvitih več pristopov k programiranju računalnikov zasnovanih na paralelnih arhitekturah. Razlog za to je, da so cilji posameznih pristopov različni.

Za pregled obstoječih vzporednih programskih jezikov je zato nujno izbrati kriterij za razlikovanje pristopov k programiranju. Razlikovali bomo:

- jezike z eksplicitnim izražanjem vzporednosti in
- jezike z implicitno vzporednostjo.

Jeziki iz prve skupine so zasnovani na programerjevi določitvi vzporednosti. Jezik PARSYS Pascal, ki je predmet obravnavanja v naslednjem poglavju, pripada tej skupini jezikov. To je razlog nekoliko podrobnejše obravnave eksplicitno vzporednih jezikov.

Drugi skupini pripadajo jeziki zasnovani na principih avtomatičnega razpoznavanja (detekcije) paralelizma v času prevajanja.

Prvi in drugi pristop imata, kot je razvidno iz slike 1, izhodišče v avtomatičnem razpoznavanju mikroparalelizma v sekvenčnih programih. Kaj to pomeni, bo razvidno iz naslednjega primera. Recimo, da v nekem programu instrukciji LOAD sledi instrukcija ADD. Če ADD ne uporablja ponornega ("target") registra instrukcije LOAD, potem je dovoljeno prekrivanje nekaj zadnjih ciklov instrukcije LOAD in prvih ciklov instrukcije ADD. Opisano razpoznavanje mikroparalelizma se opravlja v času izvajanja (run-time) programa, možno pa je ne glede na izvorni programski jezik. Na žalost so pohitritve, dosežene na ta način, relativno majhne.

2.1 Eksplicitno izražanje vzporednosti

Če želimo izračunati poti desetih neodvisnih raket, je očitno, da moremo do rezultata priti desetkrat hitreje, če namesto desetih zaporednih izračunov izvajamo po en izračun na desetih paralelnih procesorjih hkrati.

Programiranje z jeziki iz skupine eksplicitno paralelnih jezikov, kot je razvidno tudi iz navedenega primera, predpostavlja možnost naravne razčlenitve (dekompozicije) problema, ki ga rešujemo. Porazdelitev jezikov znotraj skupine eksplicitno paralelnih temelji na različnih možnostih za izražanje paralelizma. Prikazana je na sliki 2.

2.1.1 Vzporedni sekvenčni programi

Najenostavnejši pristop k eksplicitnemu paralelizmu je istočasna uporaba več sekvenčnih programov. Ti programi navadno izmenjujejo sporočila samo preko datotečnega sistema. Zato je cena teh povezav izredno visoka.

Istočasna uporaba več sekvenčnih programov se izplača samo, če paralelno izvajamo velike programe, ki redko izmenjujejo sporočila.

Dober primer uporabe tega principa je DEC VAX VMS mreža.

2.1.2 Nizko nivojska eksplicitna vzporednost

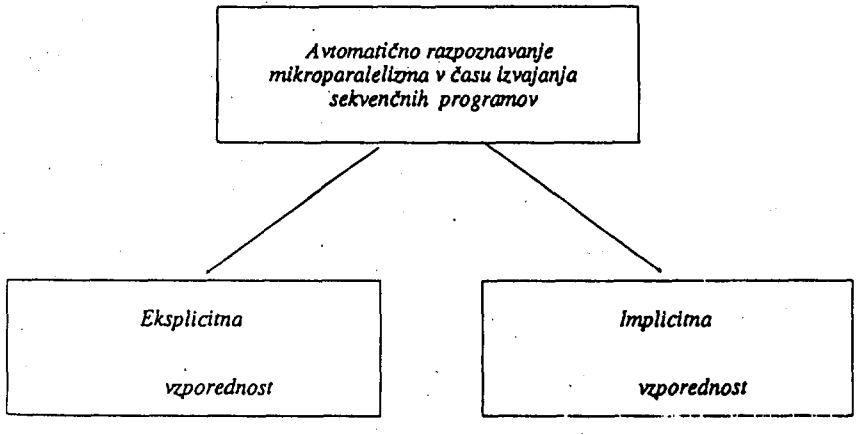
Visoka cena povezovanja in sinhronizacije programov pri predhodni tehniki je omogočala samo grobo-zrnati ("large-grain") paralelizem. Številni sekvenčni jeziki pa so spremenjeni tako, da omogočajo tudi drobno-zrnati ("fine-grain") paralelizem. Razširjeni so s funkcijami, kot so semaforji in join/fork operacije ter različnimi načini za izmenjavo sporočil (npr. buferji za sporočila). Enostavnost teh rešitev povzroča časovno potratnost in neefikasnost kompleksnih algoritmov. Po drugi strani pa ima mala napaka pri uporabi dodanih funkcij pogosto za posledico smrtni objem ("deadlock").

Uspešni primeri izkoriščanja tega pristopa so BBN Uniform Programming System in nekatere verzije Berkley in System V UNIXa za paralelne računalnike.

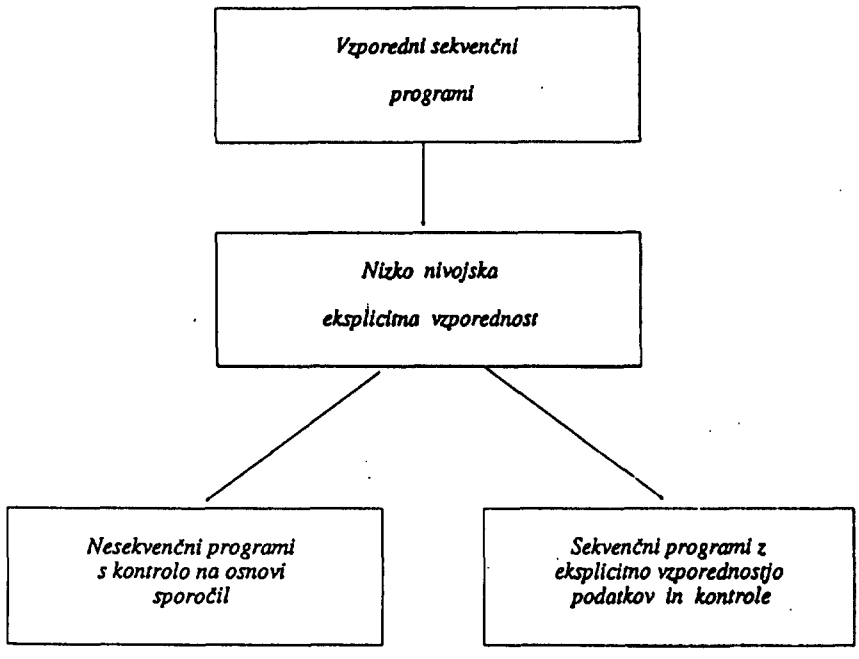
2.1.3 Nesekvenčni programi s kontrolo na osnovi sporočil

Številni pristopi k paralelnemu izvajanju so nastali na osnovi visoko nivojskih formalizmov. Verjetno najbolj znani formalizem je razvil Hoare leta 1978. To je CSP (Communicating Sequential Processes). Programi napisani v jezikih, ki so nastali iz teh formalizmov, pripadajo skupini nesekvenčnih programov s kontrolo na osnovi sporočil.

Slabost tega pristopa je slaba prilagodljivost jezikov različnim vrstam računalnikov. Sprememba računalnika more imeti za posledico občutno zmanjšanje učinkovitosti programa (na primer, če uporabljamo izmenjavo sporočil na vektorskih računalnikih). Zadovoljiva učinkovitost programa je ozko povezana s tipom računal-



Slika 1. Izhodišćna porazdelitev pristopov k vzporednemu programiranju



Slika 2. Tipi eksplicitne vzporednosti

niške arhitekture. Predstavniki jezikov tega tipa so: Ada, OCCAM in LINDA.

2.1.4 Sekvenčni programi z eksplicitno vzporednostjo podatkov in kontrole

Poleg samega podajanja modela vzporednosti (kot v prejšnjem pristopu), omogočajo jeziki tega razreda tudi modeliranje okolja, v katerem se program izvaja. Enake težave se za programerja pojavljajo pri poskusih optimalnega razvrščanja (alociranja) podatkov v pomnilnik, kot tudi pri definiranju vzporedne kontrole v programu (človek je vajen zaporednega načina razmišljanja in izvajanja operacij).

Programski jezik Actus omogoča naslednjo specifikacijo:

```
var A : array [1 : 4, 1 .. 5] of integer ;
```

Navedeni programski stavek pomeni, da je v pomnilniku matrika A organizirana tako, da so paralelni podatki razvrščeni po prvem indeksu. Z drugimi besedami, dvopičje določa indeks, po katerem so podatki razdeljeni med procesnimi elementi v matričnem računalniku, oziroma stalno shranjeni v vektorskih registrih nekega vektorskega računalnika.

Tudi nekateri drugi jeziki podpirajo podobne (vektorski orientirane) konstrukte in zato pripadajo skupini vzporednih jezikov, ki jih opisuje to poglavje. To so npr. Parallel Pascal, Vector C, Parallel C, Concurent C itd.

2.2 Implicitna vzporednost

Možnost neposrednega definiranja vzporednosti v sekvenčnih jezikih ne obstaja. S tem odpadejo tudi problemi, ki jih programerju povzročajo sinhronizacija izvajanja, izmenjava sporočil itd. Če je takšen sekvenčni jezik izhodišče za programiranje paralelnega računalnika je nemogoče, da bi programer pri izvajanju povzročil smrtni objem (deadlock). To je seveda res samo, če prevajalnik, ki transformira kodo v vzporedno formo, uporablja transformacije, ki obdržijo pravilnost delovanja. Vzporedni program bo tako proizvedel enake rezultate kot sekvenčni. Izvorni program je sedaj čitljiv (za razliko od eksplicitno vzporednih programov), iskanje napak pa ne povzroča več prevelikih težav.

Obstajajo še nekatere prednosti programiranja s sekvenčnimi jeziki. Program je prenosljiv na različne paralelne računalnike ne glede na njihovo arhitekturo s tem, da je povsod približno enako učinkovit (odvisno od kvalitete prevajalnika). Še več, program je prenosljiv tudi na enoprocesorske računalnike.

Prvi poskusi avtomatične paralelizacije sekvenčnega programskega jezika so bili narejeni s standardno verzijo jezika FORTRAN.

Nekatere tehnike paralelizacije, predvsem tiste za paralelizacijo zank, so se izkazale kot zelo dobre. Konstrukte, podobne DO zanki, je možno v FORTRANu paralelizirati skoraj idealno, za katerikoli računalnik. Čeprav je doseženi nivo paralelizacije zadovoljiv za večji del današnjih paralelnih računalnikov, z razvojem naslednje generacije to ne bo več držalo. Poleg tega so bili rezultati paralelizacije drugih sekvenčnih jezikov poleg FORTRANa slabi, tako da so danes paralelni prevajalniki za te jezike razmeroma redki.

V zadnjih letih so razvite metode (npr. metoda "prečiščevenja jezikov" - "rafined languages approach") za paralelizacijo velikega števila drugih jezikov. Te metode so dale dobre rezultate pri paralelizaciji programov pisanih v programskih jezikih C, FORTRAN, LISP, Pascal, Prolog itd. To je trenutno področje obsežnih raziskav, pričakovati pa je bistven napredek v naslednjih nekaj letih.

Poseben pristop vzporednosti znotraj skupine implicitno vzporednih jezikov se opazi pri jezikih s podatkovnim potekom ("data flow languages"). Ti jeziki so dostopni na računalnikih specifičnih arhitektur. Računalniki s podatkovnim potekom delujejo na principu izvajanja instrukcije v času dostopnosti njenih operandov. Jeziki namenjeni tem računalnikom so sestavljeni iz različnih vrednosti, izrazov in funkcij, namesto spremenljivk, stavkov in procedur. Ti jeziki onemogočajo pojave kakršnihkoli stranskih učinkov (side effects), s tem pa je proces avtomatičnega razpoznavanja vzporednosti poenostavljen. Uporaba implicitne vzporednosti in implicitne sinhronizacije pomeni, da v jezikih s podatkovnim potekom ne obstajajo možnosti za eksplicitno definiranje vzporednosti, npr. procesov ali monitorjev.

Primer jezika s podatkovnim potekom je VAL. Uporabnikom ponuja funkcije brez stranskih učinkov, kar omogoča preverjanje sintaktične, kot tudi semantične pravilnosti programa v času prevajanja. Slabost VALa so vhodno izhodne operacije. Te operacije niso realizirane v funkcijah jezika in za njihovo izvajanje je odgovoren operacijski sistem ciljnega računalnika, kar pa je možen izvor nezaželjenih pojavov.

Kljub obstoječim pomankljivostim dosedanjih rešitev so računalniki s podatkovnim potekom in njihovi jeziki področje raziskav, od katerih se v računalništvu veliko pričakuje.

Poleg vseh omenjenih prednosti implicitnega pristopa vzporednosti pred eksplicitnim programska oprema vseh paralelnih superračunalnikov še vedno vsebuje eksplicitne paralelne jezike. Razlogov je verjetno več. Eden od njih pa je gotovo podoben razlogu za definiranje asemblerskega jezika pri vseh modernih računalnikih.

3. PARSYS PASCAL

Programski jezik PARSYS Pascal je razvit kot sestavni del simulatorja paralalne programske opreme. Glede na porazdelitveno shemo paralelnih programskih jezikov, predstavljeno v predhodnem poglavju, se v programih pisanih v PARSYS Pascalu vzporednost izraža eksplicitno. Torej programer določa dele programa, ki se bodo istočasno izvajali. Bolj natančno, pa PARSYS Pascal pripada skupini nesekvenčnih jezikov s kontrolo na osnovi sporočil.

PARSYS Pascal je nadmnožica standardne verzije programskega jezika Pascal, natančneje Microsoft Pascala, ki je instaliran na računalniku Triglav pod operacijskim sistemom XENIX in podpira ISO standarde. Pascal je razširjen s primitivi, ki izražajo (navidezno) vzporednost. Definicije teh primitivov se nahajajo v simulatorju, ki se v času prevajanja (compile time) doda vsakemu PARSYS Pascal programu. To omogoča izvajanje programov v enonaslovnem prostoru, brez skokov v operacijski sistem in iz tega izhajajočih časovnih izgub. Program, pisan v jeziku PARSYS Pascal, ima strukturo podobno navadnim pascalskim programom, programerju pa je omogočeno kreiranje algoritmov z več vzporednimi procesi, ki izmenjujejo sporočila med seboj. Procesi imajo sintakso pascalske procedure s pomenom zaporedja ukazov, ki se izvaja vzporedno z drugimi zaporedji ukazov. Procesi nastajajo dinamično, v času izvajanja programa, končujejo pa se v trenutku, ko se izvede zadnji definirani stavek. Navidezni izbor procesorja, na katerem se bo proces izvajal, je dolžnost simulatorja in ni načina, da bi programer na to vplival. S pošiljanjem sporočil skozi "kanale" sta doseženi sinhronizacija in komunikacija med procesi. Kot bo razloženo kasneje, so kanali v bistvu poštni predali (mailbox), v katerih procesi puščajo sporočila in iz njih ta sporočila jemljejo. Procesi morejo komunicirati še na drug način, z uporabo skupnih spremenljivk. Poleg primitivov, ki omogočajo izvedbo navedenih možnosti, simulator vsebuje tudi primitive, ki pomagajo programerju, da se seznani s trenutnim stanjem procesov v obdelavi, npr. kateri procesi so trenutno aktivni, katera sporočila so poslana, niso pa še prebrana (uporabljena) in podobno. Prisotnost teh primitivov v simulatorju pomeni možnost njihove uporabe kot "ukazov" PARSYS Pascala.

Podroben opis vsega navedenega je podan v podpoglavjih, ki sledijo.

3.1 Spremenljivke

Pri programiranju sekvenčnih, enoprocesorskih računalnikov se spremenljivke eksplicitno deklarirajo na začetku programa pri prevajalnikih, ali pred prvo uporabo pri interpreterjih (implicitno). Pri izvajanju se nad njimi korak po koraku izvršujejo ukazi. Izvajanje teče zaporedno, spreminjanje vrednosti spremenljivk se odvija po znanem, vnaprej določenem vrstnem redu. Vrednost spremenljivke je možno spremeniti samo z enim izmed dveh načinov: prirejanjem vrednosti oziroma branjem. Če programer v sekvenci ukazov ni uporabil spremenljivke, more biti prepričan, da ima le-ta isto vrednost po izvajanju teh ukazov, kot jo je imela pred tem.

Pri vzporednem izvajanju več zaporedij (sekvenc) ukazov predhodna trditev ne drži nujno. Če vzporedni procesi uporabijo spremenljivko in spremenijo njeno vrednost, to povzroča nepričakovano in morda tudi neponovljivo vedenje programa. To je odvisno samo od (vnaprej neznanega) zaporedja časovnih intervalov, v katerih bo posamezen proces uporabil spremenljivko. To povzroča obstoj dveh različnih tipov spremenljivk v vzporednih programskih jezikih, glede na njihovo dostopnost.

Prvi tip so lokalne spremenljivke. Uporabljajo se samo v procesu, na začetku katerega so deklarirane. Za posamezni proces je že rečeno, da predstavlja navadno zaporedje ukazov, "obnašanje" lokalnih spremenljivk v njem pa je enako, kot pri spremenljivkah v sekvenčnih programih.

Drugi tip spremenljivk predstavljajo tiste spremenljivke, ki jih lahko uporablja in tudi spreminja več procesov. Spremenljivke tega tipa imenujemo skupne (shared), uporabljamo pa jih pri medprocesni komunikaciji.

PARSYS Pascal sledi navedenim razlogom in omogoča deklariranje dveh tipov spremenljivk glede na njihovo dostopnost. Poleg rezervirane besede VAR, ki se uporablja pri vseh verzijah Pascala, nudi PARSYS Pascal uporabo rezerviranih besed LOCAL (za lokalne spremenljivke) in SHARED (za globalno dostopne spremenljivke). LOCAL in SHARED se uporabljajo samo pri deklariranju spremenljivk v glavnem programu zato, ker, če je spremenljivka deklarirana v neki proceduri, funkciji, ali pri PARSYS Pascal procesu, to že samo po sebi pomeni, da je dostopna samo znotraj te procedure, funkcije ali procesa. To pa ustreza pogojem za (glede na dostopnost) lokalni tip spremenljivke. Če spremenljivko v glavnem programu deklariramo kot LOCAL, to pomeni, da je lokalno dostopna samo znotraj glavnega programa. Iz opisa simulatorja bo razvidno, da glavni program pri PARSYS Pascalu postane samo en proces v simulatorju, zato bo pojem "lokalna spremenljivka glavnega programa" razumljiv pojem. Spremenljivke deklarirane z besedo SHARED so globalno dostopne vsem procesom, proceduram in funkcijam znotraj programa in, kot je razvidno iz uvodne razprave tega podpoglavja, predstavljajo stalno nevarnost za pravilnost programov manj izkušenih avtorjev paralelnih algoritmov.

3.2 Procesi

V sekvenčnih programskih jezikih je iniciranje izvajanja programske kode naloga mehanizmov operacijskega sistema. Poleg ukazov operacijskega sistema, namenjenih iniciranju izvajanja programa, nekateri sistemi dovoljujejo "zbujanje" (in izvajanje) programa s strani drugega programa, ki se trenutno izvaja. Pri paralelnih programskih jezikih se drugi program "zbuja" kot poseben proces (proces ali posel/"task" v smislu operacijskega sistema) pod kontrolo operacijskega sistema. Kontrola se takoj po zbujanju novega procesa vrne originalnemu procesu (očetu). Procesa od tega trenutka naprej živita skupaj.

Pri paralelnih programskih jezikih programer v svojem programu uporablja ukaze za zbujanje (kreiranje, razmnoževanje) procesov, podobne ukazom operacijskega sistema. Zbujeni proces začenja "življenje" z izvajanjem svojega "glavnega programa" (proces more tudi imeti svoje procese...). Zbujanje novega procesa ima nekatere značilnosti. To so:

- možnost prenosa množice parametrov zbujenemu procesu in

- možnosti za komunikacijo in usklajevanje (sinhronizacijo) vseh "živečih" procesov.

Očitno je, da pri (asinhronih) paralelnih programskih jezikih postajajo procesi, njihovo zbujanje in "sožitje" sestavni del programiranja. Vsi zbujeni procesi morajo po opravljenjem delu tudi zaspati. To se doseže z izvrševanjem zadnjega ukaza v procesu. Ko se proces na ta način konča (in zaspi), postanejo vse njegove lokalne spremenljivke nedostopne. Končani proces se ne more reaktivirati. Če se s posebnim ukazom zbudi proces, ki ima isto programsko kodo kot že speči, ali pa še delujoči proces, je to pomensko novi proces, z novimi lokalnimi spremenljivkami in neodvisnim delovanjem.

Celotni paralelni program je zaključen, če so vsi njegovi procesi zaspali. Po končanem delovanju programa so vse lokalne, pa tudi skupne (globalne) spremenljivke nedostopne. Pri ponovnem izvajanju programa se vse spremenljivke ponovno inicijalizirajo, neodvisno od vrednosti, ki so jih imele v prejšnjem "življenskem ciklu" programa.

V navadnem Pascalu je program sestavljen iz glavnega programa ter množice podprogramov (procedur in funkcij). Edini povezavi med podprogrami in glavnim programom sta seznam parametrov in delovanje nad skupnimi spremenljivkami. Kot je že rečeno, PARSYS Pascal ohranja osnovno zgradbo pascalskih programov, le da se poleg procedur in funkcij pojavljajo še procesi. Predprocesor razpoznava proces po njegovem prvem stavku, ki je po sintaktičnih pravilih enak začetnem stavku procedur, če se namesto rezervirane besede PROCEDURE, uporablja beseda PROCESS. Poglejmo nekaj primerov:

```
PROCESS Producer;

PROCESS Cell( x, y);

PROCESS Suma(ProcID);
```

Podobni stavki se nahajajo na začetku vsakega procesa. Iz primerov je razvidna tudi specifikacija seznama parametrov, ki se prenašajo procesu pri njegovem zbujanju.

Za zbujanje procesov je v PARSYS Pascal vpeljan primitiv CREATEPROC. Primeri uporabe ukaza CREATEPROC, ki kreirajo procese z začetnimi stavki, navedenimi v prejšnem primeru so:

```
CREATEPROC Producer;

CREATEPROC Cell ( i, j);

CREATEPROC Suma ( z );
```

Semantična pravila za uporabo stavka CREATEPROC so enaka semantičnim pravilom pri klicu procedure v Pascalu. Število parametrov v CREATEPROC stavku in prvem stavku procesa mora biti enako, ujemati se morajo tudi tipi parametrov itd.

3.3 Medprocesna komunikacija

V paralelnih programskih jezikih obstajajo različni mehanizmi za izmenjavo sporočil med procesi. V uvodu poglavja o PARSYS Pascalu sta omenjena dva načina za medprocesno komunikacijo. Eden je uporaba skupnih spremenljivk. Ta način je nevaren za pravilno delovanje paralelnega algoritma zato, ker pri paralelnem izvajanju vrstni red ukazov, ki spreminjajo vrednost skupne (globalno dostopne) spremenljivke, ni v naprej znan (ukazi se nahajajo v različnih procesih). PARSYS Pascal pa nima vgrajenih splošno znanih mehanizmov za zavarovanje spremenljivk pred istočasno uporabo s strani več procesov (npr. semaforje ali monitorje). Zato se je priporočljivo izogibati tovrstni medprocesni komunikaciji.

Bolj varen in priporočljiv način za izmenjavo sporočil med več procesi je uporaba "poštnih predalov" (mailbox). "Poštni predali" pri PARSYS Pascalu predstavljajo poseben podatkovni tip. To pomeni, da morajo biti podatkovni objekti (spremenljivke), namenjeni medprocesni komunikaciji, na začetku vsakega paralelnega programa eksplicitno deklarirani. Logično je tudi to, da so "poštni predali", oziroma spremenljivke tipa SIGNAL, globalno dostopni tako glavnemu programu kot tudi vsem procesom. Deklariranje teh spremenljivk (v uvodu smo jim rekli tudi "kanali") v skupini skupnih spremenljivk (rezervirana beseda SHARED) omogoča, da morejo preko njih komunicirati vsi procesi. Spremenljivke tipa SIGNAL deklariramo na enak način, kot spremenljivke kateregakoli drugega tipa, na primer integer, real, boolean... Oglejmo si nekaj primerov:

```
SHARED

  may_consume, may_produce: SIGNAL ;
  A : array [1 .. 20 ] of SIGNAL ;
```

Spremenljivke may_consume in may_produce so navadne spremenljivke tipa SIGNAL, enodimenzionalna matrika (vektor) A, pa je sestavljena iz 20 "poštnih predalov".

Spremenljivke tipa SIGNAL se uporabljajo za prenos obvestila o nekem dogodku med procesi. Pododek, ki ga je potrebno sporočiti ostalim procesom je najpogosteje dostopnost neke informacije. Proces pošlje sporočilo drugemu procesu, ki to sporočilo pričakuje in sprejme. Na ta način je dosežena sinhronizacija dveh procesov. Prenos se izvaja z uporabo ukazov SEND in RECEIVE. Ukaz SEND prepiše sporočilo pošiljatelja v spremenljivko tipa SIGNAL. Iz te spremenljivke sporočilo prevzame nek drug proces, ki izvaja ukaz RECEIVE nad istim "poštnim predalom". Pri tem je bistveno poudariti, da proces, ki pošilja sporočilo, načeloma ne ve, kdo bo sporočilo sprejel. Tudi proces - sprejemnik sporočila ne ve, kdo je pošiljatelj. Združuje jih samo ista spremenljivka tipa SIGNAL. Za semantično pravilnost izmenjave sporočil med procesi PARSYS Pascala torej skrbi izključno programer.

Ukaz SEND se izvrši tudi, če v spremenljivki tipa SIGNAL že obstaja neko neprebrano sporočilo. V tem primeru se novo sporočilo samo doda v "kanal". Sintaksa ukaza SEND pri PARSYS Pascalu je predstavljena z Bacuss Naurovo formo.

```
<SendStmt> ::= SEND (<Sgnl>,<Dta>);

    <Sgnl> ::= <Identifier>

     <Dta> ::= <Identifier> | <IntNum>
```

Primeri:

```
SEND ( may_consume, i );

SEND ( A [i] , 10 );
```

Prvi argument ukaza SEND je spremenljivka tipa SIGNAL, drugi pa je celo število ali spremenljivka celoštevilčnega tipa (integer). Tudi ta omejitev PARSYS Pascala naj bi bila v bližji prihodnosti odpravljena.

Ukaz RECEIVE se uspešno izvrši samo, če je podatkovni objekt tipa SIGNAL poln, oziroma, če je kateri od procesov že poslal potrebno sporočilo procesu, ki izvaja ukaz RECEIVE za nadaljevanje dela. Če sporočilo do trenutka izvajanja ukaza za sprejem ni poslano, se proces sprejemnik ustavi, ker je nadaljevanje dela brez informacije, ki je del sporočila, nemogoče. Če pa v "poštnem predalu" čaka več sporočil, proces sprejemnik nakljućno izbere eno izmed njih. Razumevanje tega principa nakljućne, popolnoma nedeterminirane izbire sporočila je eden bistvenih pogojev za sestavljanje pravilnih paralelnih algoritmov. Opis dejanske izvedbe ukazov SEND in RECEIVE bo predstavljen pri opisu

simulatorja paralelne programske opreme v poglavju 4.

Primeri uporabe ukaza receive:

```
RECEIVE ( may_consume , Index ) ;

RECEIVE ( A [i] , Value ) ;
```

Podani primeri izkazujejo enega od možnih načinov za sprejem sporočil med primeri za ukaz SEND. Index in Value pri PARSYS Pascalu morata biti spremenljivki tipa integer, A i in may_consume morata biti tipa signal.

4. SIMULACIJA PARALELNE PROGRAMSKE OPREME

Paralelno procesiranje omogoča velike časovne pridobitve pri izvajanju velike skupine aplikacij. Navedena trditev je splošno veljavna, postavlja pa se vprašanje, kako oceniti te pridobitve pred dejanskim izvajanjem programa na nekem večprocesorskem ali paralelnem računalniku. Prvi način za ocenjevanje časovnih pridobitev je izpeljava natančnega matematičnega modela. Slaba stran tega pristopa je, da vsaka aplikacija potrebuje poseben pristop k modeliranju, natančen matematični model za zahtevne aplikacije pa je dokaj težko izdelati.

Drugi pristop, ki je uporabljen tudi v tem delu, je izgradnja simulatorja za vzporedne programe, ki obenem tudi vrednoti časovne pridobitve nastale kot posledica paralelnega procesiranja. Simulator ni vezan samo na eno, natančno specificirano paralelno arhitekturo. Omogoča pa, da se v času izvajanja določajo arhitekturni parametri značilni za posamezne računalnike (na primer število procesorjev). Z manjšimi spremembami bi bilo brez težav mogoče izboljšati simulator tako, da bolj precizno vrednoti izvajanje vzporednih programov na arhitekturi striktno določenih parametrov. Simulacijski program v času izvajanja nekega vzporednega programa (napisanega v že predstavljenem jeziku PARSYS Pascal) zbira številne parametre, kot so na primer koeficient uporabe posameznih procesnih elementov in podatki o medprocesni komunikaciji. Število procesorskih elementov določimo na začetku izvajanja, po potrebi ga v različnih tekih simulatorja spreminjamo in iz zbranih parametrov sklepamo o povezanosti parametrov algoritma in parametrov sistema. Na podoben način sklepamo tudi na osnovi podatkov o času izvajanja posameznih procesov itd.

Algoritem za simulacijo sloni na maksimalnem izkoriščanju vzporednosti pri izvajanju programov. Že v prvem podpoglavju bo razvidno, da algoritem za dodeljevanje procesorjev posameznim procesom izbira novi, do tega trenutka ne uporabljeni procesor samo v primeru, če so vsi ostali procesorji v trenutku dodeljevanja ("scheduling") zasedeni. Zato je očitno, da pri določenih aplikacijah ne bodo uporabljeni vsi "ponujeni" procesorski elementi. Z drugimi besedami povedano, število uporabljenih procesorjev predstavlja največje število istočasnih aktivnosti, ki jih zahteva testirani vzporedni program. Pri opisu PARSYS Pascala je rečeno, da se za sinhronizacijo procesov uporablja tehnika izmenjave sporočil. Razlog za ta izbor je bil prilagojenost te tehnike večprocesorskim računalnikom s skupnim pomnilnikom in računalniškim mrežam, ter njena razširjenost. Dovoljena je tudi uporaba skupnih spremenljivk. Procesi komunicirajo skozi kanale. Več procesov more izkoriščati en kanal za pošiljanje svojih sporočil. Enaka je situacija tudi na izhodni strani kanalov, kjer več procesov pričakuje sporočila iz istega kanala. Ampak eno sporočilo more prebrati samo en proces, oziroma z enim sporočilom je "zbujen" samo en proces med tistimi, ki so sporočilo pričakovali iz določenega kanala.

Simulator na grobo opisanih karakteristik je razvit v visoko nivojskem programskem jeziku, izvaja pa se v enoprogramskem okolju. Razvoj je potekal na računalnikih Delta Partner (operacijski sistem CP/M) in Triglav pod operacijskim sistemom XENIX. Zato je uporabljen programski jezik Pascal, ki je dostopen na obeh računalnikih. V kratkem času bo simulator dostopen na IBM PC AT in kompatibilnih računalnikih pod operacijskim sistemom DOS.

Simulator deluje na enoprocesorskih, opisuje pa dogajanja v večprocesorskih računalnikih. Zato je nujno pred detaljnim opisom posameznih delov simulatorja, opisati algoritem za simulacijo in dokazati njegovo pravilnost.

4.1 Simulacijski algoritem

Predpostavlja se, da moramo simulirati izvajanje množice procesov na računalniku z omejenim številom procesorjev. Vsak proces se sme izvajati na kateremkoli procesorskem elementu. Če ima simulirani sistem samo en procesor, potem se istočasno izvaja največ en proces, problemov pri simulaciji pa ni. Za simulacijo istočasnih dogodkov v sistemu z večimi procesorji moramo specificirati koncept "vodenja časa" v simulatorju.

V spremenljivki T vsakega procesa (lokalno) je shranjena ura. Procesi se izbirajo za izvajanje en za drugim, in njihova ura se postavi na čas, v katerem naj bi se proces začel izvajati v realnem večprocesorsken sistemu. Vsakič, ko se izvrši ena operacija, se uri prišteje količina časa, potrebna za izvajanje te operacije na simuliranem računalniku. Sinhronizaciji med procesi (komunikacija in kreiranje) je pridružen čas T, (trenutno stanje spremenljivke) v katerem je izvedena.

Procesi

Za vsak proces i je shranjena informacija o njegovem statusu ("ready" ali "waiting"). Spremenljivka BT(i) (Block Time) predstavlja čas, ko je proces zadnjič prišel v stanje "waiting". RT(i) (Ready Time) je spremenljivka, v kateri je shranjen čas, ko je proces postal pripravljen (če je v stanju "ready"). RT(i) se po navadi postavi na čas dogodka, ki zbudi proces i, BT(i) pa je dejansko spodnja meja za RT(i) (dogodek, ki se zgodi pred BT(i) povzroči, da proces postane "ponovno pripravljen" v trenutku BT(i), torej proces v tem primeru dejansko sploh ni bil blokiran in ustavljen). Vsi pripravljeni ("ready") procesi so člani vrste, ki je urejena po naraščajoči vrednosti spremenljivke RT(i). Proces kreiran v trenutku T postane pripravljen na naslednji način:

```
RT(i) := T;                                   (1.1)
BT(i) := T;
```

Dogodek v trenutku T, na katerega je čakal čakajoči (blokirani, "waiting") proces i, povzroči naslednjo spremembo časovnih spremenljivk:

```
RT(i) := max ( T, BT(i) );    (1.2)
```

Če proces i ni bil v stanju "waiting", potem se RT(i) spremeni v skladu z izrazom (1.2) samo, če je bil prejšnji RT(i) večji od T (RT(i) > T). Proces se glede na nastalo spremembo RT(i) postavi v vrsto pripravljenih dogodkov.

Procesorji

Vsak procesor <k> je opisan z dvema vrednostima. ST(k) (Stop Time) je trenutek, v katerem je procesor zadnjič bil ustavljen; WT(k) (Working Time) je akumuliran čas delovanja procesorskega elementa. WT(k) se uporablja samo za vodenje statistike o procesorjih.

Simulacijski algoritem

Naj bo j indeks pripravljenega ("ready") procesa in Kj množica procesorjev, ki so v tem trenutku prosti. TSj bo prvi trenutek (najnižji čas), v katerem se more proces j začeti izvajati. Potem velja

$$TSj = \max (RT(j) , ST(k)) \quad (2)$$

pri čemer je k procesor iz množice Kj, za katerega velja:

- k je najnižji indeks pri katerem velja $ST(k) < RT(j)$ (2.1)

ali, če pogoj (2.1) ne more biti izpolnjen

- $ST(k) < ST(h)$, za vsak h element množice Kj (2.2)

Proces i je izbran za izvajanje, če velja
- $TSi <= TSj$ za vse j : (proces j je v stanju "ready"). (3)

Vrednost spremenljivke T se pri ponovnem aktiviranju procesa postavi na TSi. Izbrani (navidezni) procesor, na katerem se proces i izvaja, ustreza pogojem (2.1) in (2.2). V času izvajanja procesa (execution time) se T inkrementira na že opisani način in ima tako funkcijo lokalne, realne ure (večprocesorskega sistema).

Ko se proces ustavi (blokira) ali konča v trenutku Tb, morajo biti izvedene naslednje spremembe vrednosti spremenljivk:

- $BT(i) := Tb$ (4.1)
  pomnjenje vrednosti ure v trenutku ustavljanja procesa,

- $ST(k) := TB$ (4.2)
  pomnjenje trenutka, v katerem je procesor postal prost,

- $WT(k) := WT(k) + (Tb - Ts)$ (4.3)
  zbiranje delovnega časa posameznega procesorja.

V slučaju, da proces začenja svoje simulirano življenje (izvajanje) v trenutku Ts, opisani algoritem zagotavlja, da so vsi dogodki pred Ts že simulirani.

Dokaz:

Predpostavili bomo, da je prvi proces postal pripravljen za izvajanje ("ready") v trenutku TØ in da so vsi dogodki pred TØ simulirani pravilno. To je gotovo prav na začetku vsake simulacije, ko je množica procesov že kreirana, noben od njih pa se še ne izvaja. Proces i se izbere za izvajanje po pravilu (3), TSi po definiciji ne more biti manjši od TØ. Dodatni dogodki, ki bi spremenili stanje simulacije se ne morejo zgoditi pred TSi zato, ker ni procesa, pripravljenega za izvajanje, ali ni procesorja, na katerem bi se pripravljeni proces izvajal. Proces i se začenja z izvajanjem v trenutku TSi in do trenutka Tb, v katerem se bo končal, izvede vse zahtevane operacije. V trenutku Tb je proces končan ali pa čaka dogodek/sporočilo na nekem kanalu in zato spremeni stanje ("waiting"). Tretja možnost je, da je ustavljeni proces takoj postavljen nazaj v vrsto pripravljenih procesov, zato ker je bil dogodek, potreben za nadaljevanje dela procesa, generiran pred Tb.

Predpostavimo, da je TSj trenutek, v katerem se je začel izvajati naslednji proces. TSj je po pravilu (3) večji ali enak TSi (TSi <= TSj), proces i pa je edini proces, ki je mogel generirati dogodke med TSi (TØ) in TSj. Potem so vsa dogajanja pred TSj pravilno izvedena, stanje simuliranega sistema pa je enako stanju opisanem na začetku tega dokaza. Če z istim premislekom nadaljujemo do trenutka, ko so vsi procesi končani, potem je pravilnost simulacijskega algoritma dokazana.

4.2 Opis procedur simulatorja

V skupini procedur, ki izvajajo "ukaze" PARSYS Pascala, sta do sedaj največkrat omenjeni proceduri SEND in RECEIVE. To sta tudi najbolj pomembni proceduri za razumevanje implementacije simulacijskega algoritma. Zato bosta opisani v posebnih poglavjih (4.2.1 in 4.2.2).

4.2.1 Procedura send

Klic procedure send vsebuje dva parametra. Prvi parameter je "poštni predal" (mailbox), na katerega naj se sporočilo pošlje, drugi parameter pa je kazalec na sporočilo (message), ki se pošilja. V proceduri send so opredeljene vse akcije, ki se izvedejo pri oddajanju sporočila v izbrani "kanal". Procedura send deluje na osnovi naslednjega algoritma.

1. Preveri, ali obstaja čakajoči proces v polju mailbox.pcbs
2. Če obstaja, potem nadaljuj s korakom 3.1, sicer skoči na 4.1

3.1. Če proces prihaja v mailbox ravno v tem trenutku, ga počakaj

3.2. Kazalec delay v procesnem kontrolnem bloku postavi na sporočilo

3.3. Kazalec, ki je v polju čakajočih procesov (mailbox.pcbs) kazal na izbrani proces, postavi na NIL

3.4. Pravilno določi čas za nadaljevanje delovanja procesa (spremenljivka t_ready se postavi v skladu s postopkom opisanim v poglavju 4.1)

3.5. Pokliči jedro simulatorja, da postavi proces v vrsto "pripravljenih" procesov" (polje ready)

3.6. Skoči na korak 5

4.1. Poišči prvo prosto lokacijo v polju sporočil (mailbox.msgs)

4.2. Če je polje sporočil polno, potem počakaj

4.3. Na prosto lokacijo v vrsti sporočil postavi sporočilo (message)

4.4. Postavi vrednost spremenljivke message.t_send na čas, v katerem je sporočilo poslano

4.5. Sortiraj polje sporočil v "kanalu" (mailbox.msgs) po naraščajoči vrednosti spremenljivke message.t_send

5. Konec.

4.2.2 Funkcija receive

Naloge funkcije (ali "ukaza") receive so bile v predhodnih poglavjih pogosto opisane. Zato bom sedaj podal samo način njene izvedbe. Funkcija receive je tipa boolean in ima samo en parameter. To je "kanal", v katerem proces, ki želi sprejeti sporočilo, to sporočilo išče. Kot v primeru procedure send, bom opisal algoritem delovanja funkcije receive.

1. Preveri, ali obstaja sporočilo, ki čaka v "poštnem predalu" (mailboxu)
2. Če obstaja, nadaljuj s korakom 3.1, sicer skoči na 4.1

3.1. Če sporočilo prihaja ravno v tem trenutku, ga počakaj

3.2. Vsebino sporočila prepiši direktno v spremenljivko CPU_receive_reg (zato, ker se proces, ki izvaja funkcijo receive v tem trenutku procesira)

3.3. Kazalec, ki je v polju čakajočih sporočil mailbox.msgs) kazal na izbrani proces, se postavi na NIL

3.4. Vrednost funkcije receive postane TRUE

3.5. Skoči na 5

4.1. Postavi proces, ki izvaja funkcijo receive v stanje "WILL_SUSPEND" (to je oznaka, da bo proces v bližnji prihodnosti verjetno zaključen ali blokiran)

4.2. Preveri še enkrat, če obstaja sporočilo

4.3. Če je med prvim in drugim preverjanjem prišlo sporočilo, potem nadaljuj s korakom 4.3.1, sicer skoči na 4.4

4.3.1. Postavi proces v stanje "RUNNING"

4.3.2. Skoči na 3.1

4.4. Poišči indeks proste lokacije v polju mailbox.pcbs

4.5. V prosto lokacijo postavi kazalec na kontrolni blok procesa

4.6. Na novo določi vrednosti časovnim spremenljivkam v procesnem kontrolnem bloku (na način, ki je opisan v poglavju 5.1)

4.7. Uredi polje čakajočih procesov po naraščajoči vrednosti spremenljivke t_block

4.8. Pokliči jedro simulatorja, ki spremeni stanje procesa v "SUSPEND" in na procesorju opravi spremembo konteksta (context switch)

4.9. Vrednost funkcije receive postavi na FALSE

5. Konec

5. VREDNOTENJE VZPOREDNIH PROGRAMOV

Tema zadnjega poglavja je vrednotenje različnih vzporednih algoritmov. Vrednotenje programske opreme je zelo širok pojem. Zato je potrebno opredeliti pristop k vrednotenju, ki bo uporabljen v tem poglavju. Vzporedni programi bodo vrednoteni na osnovi časovnih pridobitev, ki izhajajo iz spremembe (zvišanja) števila procesorjev, na katerih se program izvaja, oziroma bolj splošno, iz prilagojenosti algoritma določeni računalniški konfiguraciji.

Poleg spreminjanja števila procesnih elementov, kar omogoča simulator vzporednih programov opisan v predhodnem poglavju, bo izvedeno tudi eksperimentalno ugotavljanje pohitritev izvajanja vzporednih programov v odvisnosti od velikosti (granularnosti) posameznih procesov vzporednega algoritma.

Pri ocenjevanju časovnih pridobitev, ki izhajajo iz simuliranega izvajanja vzporednega programa na različnem številu procesnih elementov, je potrebno določiti primerno mero. Za namene tega dela bo zadoščal faktor pohitritve SU (Speed Up), ki je definiran na naslednji način:

$$SU\ i,j = T\ i\ /\ T\ j \qquad (5)$$

Faktor pohitritve je kvocient časa (simuliranega) izvajanja programa na i-tih in izvajanja programa na j-tih procesorjih. V podpoglavjih, ki sledijo, bo vedno veljala naslednja relacija:

$$j = 2 * i \qquad (6)$$

Zato je SU i,j zmeraj med 1 in 2. Z dvakratnim zvišanjem števila procesnih elementov se hitrost izvajanja nikakor ne more znižati. Z druge strani pa je največja možna pohitritev izvajanja, sicer samo teoretično dostopna, dvakratna, zato je SU i,j z gornje strani omejen z 2.

5.1 Enostavni algoritem za vzporedno sortiranje

Sekvenčni programi, ki uporabljajo različne metode za sortiranje polj, imajo časovno zahtevnost med $O(n)*\log_2(n)$ in $O(n^2)$.

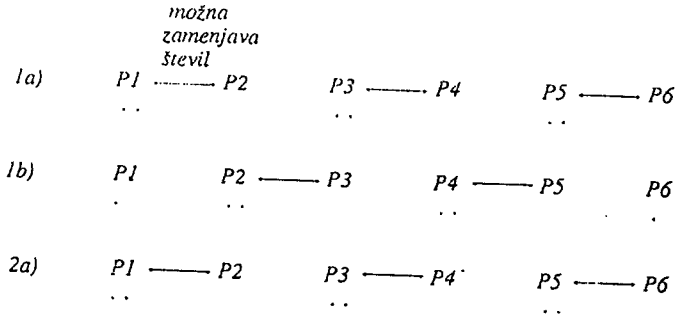
Vzporedni algoritem, ki sledi, pa ima časovno zahtevnost $n^2 / 2$, torej reda O(n). Ime mu je sodo - liho urejanje (Odd - Even Sort) in spominja na sekvenčno urejanje z mehurčki (Bubble Sort).

Osnovni cikel urejanja po metodi sodo - liho je naslednji: vsak proces v začetku delovanja programa dobi identifikator. Vsi procesi z lihim identifikatorjem (zaporednim številom, TaskID) dobijo tudi dve "žogici". To je oznaka, da morejo zamenjati število iz polja A (polje, ki ga je potrebno sortirati), ki ima indeks TaskID, s številom A [TaskID+1] , če je A [taskID] < A[TaskID+1]. Potem proces z lihim identifikatorjem pošlje (SEND) svoji žogici sosednjim (sodim) procesom. Vsak proces s sodim identifikatorjem ima 2 liha soseda, tako da dobi dve žogici in more zamenjati število A TaskID z A TaskID+1 . Ko je ta operacija končana, se žogici spet pošljeta (lihim) sosedom, ki tako nadaljujeta delo. Če neki proces nima dveh sosedov (prvi in zadnji) potem žogico soseda, ki manjka, pošilja sam sebi. V naslednjem koraku pa ne more biti aktiven, ker ima samo eno žogico, za aktivnost pa potrebuje dve. Za sortiranje polja bo potrebno največ n/2 (navzgor omejeno celo število) opisanih korakov, ki so prikazani tudi na sliki 3.

Program Odd_Even_Sort, ki deluje na opisani način je napisan v programskem jeziku PARSYS Pascal. Algoritem uredi 16 števil (MAXID = 16), časi simuliranega izvajanja za različno število procesnih elementov pa so naslednji :

#PE	T	SU
1	464	
		1.97
2	235	
		1.94
4	121	
		1.83
8	66	
		1.38
16	48	

#PE je število procesnih elementov pri simuliranem izvajanju programa, T so časi teh izvajanj, SU pa je faktor pohitritve izračunan po formuli (1). Pri spreminjanju števila procesorjev od 1 do 8 je SU skoraj idealen. Manjša rast hitrosti izvajanja pri spreminjanju #PE od 8 na 16 je tudi razumljiva, ker je iz opisa algoritma razvidno, da se istočasno izvaja n / 2 zamenjav števil v polju A. V primeru, ko imamo 16 procesorjev, je velik tudi vpliv časa za spreminjanje konteksta (context switch time) na



Slika 3. Prikaz algoritma za vzporedno sortiranje

vsakem procesorju, zato je faktor SU manjši od predhodnjih.

8,16

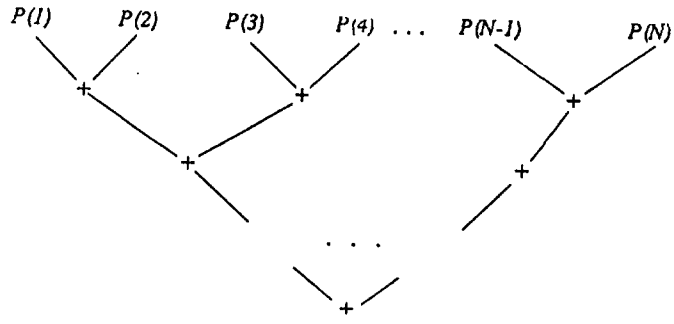
5.2 Algoritem za seštevanje v logaritmičnem času

V primeru algoritma za sortiranje smo si ogledali samo vpliv ene komponente računalniške konfiguracije na čas izvajanja vzporednega programa. Sedaj bo predstavljen algoritem za seštevanje množice števil v času proporcionalnem dvojiškem logaritmu števila sumandov. Pri sekvenčnih algoritmih za seštevanje n števil je potreban čas reda O(n) (vsoti predhodnjih se prišteje naslednje število). Ideja vzporednega seštevanja n števil sloni na lastnostih komutativnosti in asociativnosti seštevanja. Kako N števil seštejemo v času proporcionalnem log (n)
2
nam kaže slika 4.

Za demonstracijo je izbran primer seštevanje rezultatov 17 neodvisnih procesov. Vsak proces v samem začetku svojega življenja kreira še dva procesa - otroka (in jima dodeli zaporedne številke 2*IDD in 2*IDD + 1), ki od tega trenutka dalje živita neodvisno od očeta. Oče, preden zaključi svoje delovanje, pričakuje od otrok rezultate njihovega procesiranja, ki jih sešteje s svojim rezultatom in to pošlje svojemu očetu (dedu). Skupna časovna zahtevnost tega algoritma je večja od načrtovane logaritmične zaradi medsebojnega kreiranja procesov. Časovni rezultati simuliranega izvajanja osnovne inačice tega programa so naslednji:

*PE	T	SU
1	255	
		1.88
2	136	
		Y.70
4	80	
		1.25
8	60	
		1.25
16	50	
		1.33
32	41	

Zanimivo in obenem tudi razumljivo je, da je dosežen isti časovni rezultat simuliranega izvajanja kot pri 32 procesorjih že pri procesiranju na računalniški konfiguraciji s 17 procesorji zato, ker program kreira 17 različnih procesov. Vsak proces zase pa je zelo majhen. Razen kreiranja dveh procesov - otrok, samo še sprejeme njihove rezultate in jih sešteje z svojo vrednostjo. Čas, v katerem procesi čakajo na sporočila, je bistveno večji od časa dejanskega



Slika 4. Vzporedno seštevanje

izvajanja procesne kode. Zato so v vse procese dodana "bremena". Izkazalo se je, da so za večje posamezne procese faktorji pohitritve SU bistveno večji kot pri majhnih procesih. V naslednjem primeru bo "breme" prazna for zanka velikosti 100000 kroženj. Že znana tabela, ki vsebuje čase simuliranega izvajanja in faktorje pohitritve za obremenjeni program je podana v nadaljevanju:

*PE	T	SU
1	1723	
		1.93
2	889	
		1.76
4	503	
		1.63
8	308	
		1.43
16	214	
		1.51
32	142	

Vsi faktorji pohitritve so se zvišali. Časovne pridobitve vzporednega programa pri procesih večje granularnosti so večje. Ta sklep je še bolj prepričljiv, če pogledamo časovne rezultate simuliranega izvajanja vzporednega programa pri še 5 krat večjem bremenu kot v prejšnjem eksperimentu. Breme je sedaj prazna for zanka velikosti 500000 kroženj.

*PE	T	SU
1	7126	
		1.90
2	3750	
		1.79
4	2085	
		1.65
8	1258	
		1.48
16	849	
		1.83
32	426	

Predstavljeni rezultati pomenijo, da je za večprocesorske računalnike potrebno pisati take algoritme, v katerih je procesiranje razdeljeno na več procesov optimalne velikosti. Kolikšna pa je optimalna velikost, v mojem delu ni ugotovljeno. Iskazalo se je samo, da so časovne pridobitve vzporednih algoritmov večje, če so posamezni procesi vzporednega programa večji.

6. S K L E P

V članku je opisan simulator večprocesorskih sistemov skupaj s simulacijskim algoritmom za njegovo izvajanje na enoprocesorskem računalniku. Simulator je sestavljen iz množice procedur in funkcij, ki omogočajo programiranje v programskem jeziku Pascal, razširjenem s konstrukti za podpiranje (navidezno) vzporednega procesiranja. To so v prvi vrsti procesi, deli programske kode, ki se morejo izvajati vzporedno, kreirajo pa se dinamično v času izvajanja programov in mehanizmi za medprocesno komunikacijo (kanali in skupne spremenljivke).

Sproti s simuliranjem izvajanja vzporednih programov simulator zbira statistične parametre, ki omogočajo vrednotenje časovnega obnašanja večprocesorskih sistemov, neomejenih po številu procesnih elementov in procesov.

Časovno obnašanje simuliranih programov je pokazalo, da mora biti pri pisanju učinkovitih vzporednih algoritmov posebnapozornost namenjena porazdelitvi procesiranja po procesih. Pri tem je potrebno v čim večji meri slediti možnostim za naravno dekompozicijo reševanega problema in program sestaviti iz ne prevelikega števila, medseboj čim bolj neodvisnih procesov optimalne velikosti.

Na osnovi spreminjanja velikosti procesov pri vrednotenju vzporednih programov sem sklepal, da so časovne pridobitve pri paralelnem procesiranju večje, če so posamezni procesi večji. Predpostavljam pa, da obstaja meja za velikost procesov, preko katere procesov glede na hitrost izvajanja ni vredno povečevati. Določanje te meje je možna pot za nadaljevanje začetih raziskav.

Simulator vzporedne programske opreme je možno izboljšati na več načinov, na primer z grafičnim paketom, ki bi prikazoval dogajanja v simuliranem večprocesorskem računalniku ali z dodajanjem procedur, ki bi omogočale uporabniku, da sam določa hitrosti izvajanja posameznih ukazov in hitrosti dostopa do spremenljivk (izkušnje kažejo, da je dostop do spremenljivk v skupnem pomnilniku 3 do 4 krat počasnejši od lokalnega dostopa). To bi omogočilo kvalitetno simuliranje delovanja računalnikov točno določenih arhitektur. Rezultate takih simulacij bi bilo možno izkoristiti pri napovedovanju performans paralelnih računalnikov v razvoju. Modularna zgradba simulatorja predstavlja dobro izhodišče za nadaljevanje njegovega razvoja v eni od omenjenih smeri.

SEZNAM UPORABLJENIH VIROV

1. R. H. Dietz : The Refined - Language Approach To Compiling For Parallel Supercomputers, Purdue University, International Report, April 1987, (str 1 - 24, 104 - 118)

2. R. H. Perrot : Parallel Programming, Addison - Wesley Publishing Company, GB, 1987.

3. J. Miklosko, V. E. Kotov : Algorithms, Software and Hardware of Parallel Computers, VEDA, Bratislava, 1984.

4. Andrew S. Tanenbaum : Operating Systems, Design and Implementation, Prentice Hall Englewood Cliffs, NJ, 1987.

5. L. Rizzo : Simulation and performance evaluation of parallel software on multiprocessor systems, Microprocessors and Microsystems, Vol 13, No 1, January/February 1989, (str. 39 - 46)

6. P. O. Frederickson, R. E. Jones, B. T. Smith: Synchronization and control of parallel algorithms, Parallel Computing 2, North-Holland, 1985, (str. 255 - 264)

7. C. Ghezzi : Concurency in programming languages : A survey, Parallel Computing 2, North-Holland, 1985, (str 229 - 241)

INFORMATICA 3/90

RAČUNALNIŠKI SISTEM »SPHINX« ZA PREPOZNAVANJE KONTINUALNOG GOVORA NEODVISNIH GOVORNIKA SA VELIKIM REČNIKOM

Marijan M. Miletić
Lab. za računarsko razumevanje jezika,
Institut »Jožef Stefan«, Ljubljana

Keywords: continuous speech, Markov models, recognition, SPHINX

Sadržaj: U članku je opisan sistem SPHINX koji je razvio Kai-Fu Lee na Carnegie-Mellon Univerzitetu. Ovo je trenutno najbolji sistem za prepoznavanje kontinualnog govora neodvisnih govornika sa velikim rečnikom engleskog jezika. Baziran je na diskretnim skrivenim Markovljevim modelima i statističkim osobinama parova fonema te lako prenosljiv na ostale jezike.

Abstract: This article describes SPHINX system developed by Kai-Fu Lee on Carnegie-Mellon University. This is the best English language system for a large vocabulary speaker indepedent continuous speech recognition. It is based on discrete hidden Markov models and statistical properties of phone pairs and therefore easily transferable to other languages.

1. Uvod

Sistemi za prepoznavanje govora su kompleksni sistemi sa bazirani na digitalnoj obradi signala, prepoznavanju uzoraka, veštačkoj intelegenciji, statistici, lingvistici i fonetici. Uspešno razvijeni govorni sistemi postigli su dobre performanse zahvaljujući uvodjenju sledečih ograničenja:

1. poznat govornik,
2. izolovane reči,
3. mali rečnik i
4. jednostavna gramatika.

U sledečoj tabeli dat je prikaz najpoznatijih sistema:

GOVORNI SISTEM	NEODV. GOVOR.	KONT. GOVOR	VELIKI RECNIK	PRIR. GRAM.
NTT	NE	NE	NE	NE
DRAGON	NE	DA	NE	DA
HEARSAY	NE	DA	DA	NE
HARPY	NE	DA	DA	NE
BELL 82	DA	NE	NE	NE
FEATURE	DA	NE	NE	NE
TANGORA	NE	NE	DA	DA
BYBLOS	NE	DA	DA	NE
BELL 88	DA	DA	NE	NE

Prepoznavanje neodvisnih govornika je najteži zadatak jer su parametarske reprezentacije govora veoma odvisne od pojedinih govornika sa manjom grupom izrazito problematičnih govornika. Performanse sistema prepoznavanja opadaju za 10-40% pri radu sa nepoznatim govornicima. Treniranje sistema za poznatog govornika može se pokazati nepouzdano u situacijama kada se glas menja usled promene položaja mikrofona ili bolesti, zamora i napetosti.

Prepoznavanje kontinualnog govora je znatno otežano nepouzdanošću utvrdjivanja početka i kraja pojedinih reči. Ograničena dinamika govornog trakta dovodi do velike medju-odvisnosti fonema, a u kontinualnom govoru i celih reči.

Artikulacija kontinualnog govora je različita od izgovoranja izolonanih reči. Isprekidan govor je neprirodan način komuniciranja za čoveka i primenljiv je samo u situacijama koje donose velike beneficije direktnog diktiranja, naprimer opis rentgenskih snimaka. Porast grešaka kod obrade kontinualnog govora ide do 15%, a vreme potrebno za procesiranje je trostruko. Brzina isprekidanog govora je oko 70 reči u minuti, dok kontinualan govornik postiže oko 170 engleskih reči u minuti.

Veliki rečnici imaju uglavnom preko 1000 reči i tad broj konfuznih reči dramatično raste. Mali rečnici mogu se u celini modelirati za svaku reč posebno. U velikim rečnicima koristimo manje jedinice govora što dovodi i do desetostrukog povečanja grešaka u prepoznavanju. Pretraživanje celog rečnika postaje praktično nemoguče.

Gramatika dodatno komplikuje upotrebu rečnika. Najkompleksniji IBM sistem TAGORA koristi rečnik od 20000 reči sa gramatikom baziranom na verovatnoči pojava grupa od tri reči.
Informaciona kompleksnost merena entropijom u svakoj tački odlučivanja za ovaj sistem iznosi 200. Statistički podaci dobijeni su obradom 230 miliona reči engleskog teksta. Nažalost, u našim jezicima poziciona gramatika ima malu praktičnu vrednost. Prirodna gramatika je poželjna kod direktnog diktiranja računaru.

Za prevazilaženje gornjih ograničenja potrebni su sledeći elementi:

1. Složeni ali razumljivi modeli govora,
2. Uvodjenje ekspertnog znanja o akustici i fonetici,
3. Mogučnost efikasnog automatskog učenja govora i
4. Prevazilaženje razlika izmedju govornika.

2. SPHINX sistem

Sistem je namenjen upravljanju resursima mornarice i može se generalno posmatrati kao pretraživanje baze podataka. Rečnik ima 997 reči i korišćene su gramatike sa parovima reči u jednostavnim listama redosleda informativne kompleksnosti 60 ili statistička verovatnočom složenosti 20.

Korišćena je TI-MIT baza podataka sa 6300 rečenica snimljenih sa 630 govornika. Rečenice su izabrane sa što boljim fonemskim balansom. Većina govornika / 70% / su odrasli muškarci. Korišćeno je 48 fonetskih labela efektivno razdeljenih u 39 odvojenih grupa.

Govor je digitaliziran sa 16 kHz i naglašen sa filtrom prenosne karakteristike 1 - 0.97 / z. Hammingov prozor širine 20 ms primenjen je svakih 10 ms. Autokorelacionom metodom dobijeno je 14 LPC koeficienata. Izveden je skup od 12 kepstralnih koeficienata koji su bilinearno tranformirani u "mel" skalu. Iz ovih podataka generisane su tri kodne tablice od po 256 prototipnih vektora. Opseg govornih podataka smanjen je za faktor 48. Nagib govora dobijen je poredjenjem koeficienata za vremensku razliku od 40 ms. Energija govora i diferencijalna energija čine treću grupu.

Problemi evaluacije, dekodiranja i učenja su osnova u skrivenim Markovljevim modelima. Za evaluaciju je korišćen "unapred" algoritam, za dekodiranje Viterbi, a za učenje Baum-Welch "napred-nazad" algoritam. Pri impletaciji modela velika pažnja posvećena je inicijalizaciji parametara, nultim i povezanim tranzicijama.

Skriveni Markovljev model sa 7 stanja, 12 tranzicija i 3 izlazne funkcije verovatnoće gustine / pdf-probability density function / kreiran je za svih 48 fona. Tri tabele daju 256x3x3 = 2304 pdf.

Markovljevi parametarski modeli korišćeni su za sve izvore znanja: fone, reči i rečenice. Dva stohastička procesa omogućavaju modeliranje akustičnih fenomena i vremenskih izobličenja. Efikasni algoritmi razvijeni su za tačnu procenu parametara. Svaka iteracija prilikom treniranja modela dovodi do konačnog poboljšanja parametara. Efikasno stohastično pretraživanje zahtijeva znatno manje prostora i vremena od ostalih strategija. Postignuta je tačnost od 26,58 i 75% sa respektivnom gramatikom kompleksnosti 997,60 i 20.

Bilinearna transformacija povećava frekventnu neodvisnost skladno sa znanjem o čovekovoj percepciji. Diferencijalni koeficienti su lako uočljivi kod spektrograma. Promena energije je dobra indikacija prosodike. Uvodjenjem ovih fiksnih koeficienata povećana je tačnost na 40,81 i 87%.

Fonološka znanja o engleskom jeziku iskorišćena su za optimizaciju topologije pojedinih fona. To je rezultiralo u daljnem povećanju tačnosti na 50,84 i 90%.

Osnovna jedinica govora u SPHINX sistemu je generalizovani trofon zavisan od levog i desnog konteksta. Algoritmi grupisanja korišćeni su za identificiranje sličnih konteksta. Eksplicitno definisanje modela fona u 42 funkcionalne reči pomaže pri prepoznavanju konfuznih elemenata. Pomoću ovih metoda dobijena je krajnja tačnost od 71,93 i 96% koju je autor ovog članka kao problematičan govornik engleskog jezika lično verificirao.

3. Zaključak

Rezultati poredjenja sa sličnim sistemima su:

SISTEM	RECNIK	SLOZEN.	TACNOST	RECI
HEARSAY	1011	4.5	86 %	-
HARPY	1011	4.5	97	-
LASER	1000	24	91.1	88.6%
BYBLOS	997	60	94.8	92.5
BYBLOS	997	997	70.1	67.6
SPICOS	917	58	-	86.3
TI	997	997	55.5	44.3
SRI	997	997	43.6	40.4
SPHINX	997	997	74.0	71.2
SPHINX	997	60	94.2	93.3
SPHINX	997	20	96.0	95.6

U koloni za tačne reči kao greške se uzimaju i ubačene reči.

Na Carnegie-Mellon Univerzitetu sa dugom tradicijom istraživanja u prepoznavanju govora došli su do zaključka da ukoliko se raspolaže sa dovoljno podataka o govoru moguće je razviti jake algoritme čija se tačnost uvek popravlja sa automatskim učenjem.

Nažalost, trenutna situacija sa nepostojećim bazama govornih podataka u Jugoslaviji ne daje mogućnosti za ovakav pristup ni jednom našem jeziku.

5. Literatura

Kai-Fu Lee PHd thesis

Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX system

CMU April 1988

KRITERIJI ZA DOLOČANJE SISTEMSKE PROGRAMSKE OPREME OSEBNIH RAČUNALNIKOV ZA KRMILJENJE INDUSTRIJSKIH PROCESOV

INFORMATICA 3/90

Keywords: SCADA program, NETBIOS compatible network, distributed system, file server, X-Windows technology

Dragan Mrdaković
Odsek za energetiko in vodenje procesov
Institut »Jožef Stefan«, Ljubljana

Osebni računalniki postajajo vse bolj popularni za krmiljenje industrijskih procesov. V članku so določeni kriteriji za izbiro njihove sistemske programske opreme za ta namen. Na njihovi osnovi je izbran programski paket, ki je uporabljen v aplikacijah za kombinirano proizvodnjo električne energije in informatizacijo energetskih sistemov.

SYSTEM SOFTWARE FOR PROCESS CONTROL IN INDUSTRY - Personal computers are becoming more and more popular for process control in industry. In this work criteria for the choice of their system software are determined. On their basis the SCADA program, which is used in applications for combined heat and power production and informatization of internal energy systems, is chosen.

1.0. UVOD

Vse večja popularnost osebnih računalnikov vpliva tudi na področje njihove uporabe. Poleg pisarniških aplikacij, postaja vse bolj aktualna rešitev problemov v industriji. Na svetovnem tržišču obstaja velika izbira sistemske programske opreme in kvalitetnih vhodno-izhodnih modulov, ki osebni računalnik spremenijo v močno procesorsko postajo za vodenje industrijskih procesov.
Sistemska programska oprema je najbolj izpostavljen del sistema. Od nje je odvisen uporabniški vpogled v aplikacijo. Mišljenja smo, da je pogosto pametnejše koncentrirati moči na tehnologijo kot na razvoj samega orodja. V Institutu smo že razvili SCADA ((Supervisory Control And Data Aquisition) sistem (hardware in software) na osnovi PC računalnikov /1/. Toda, hitre spremembe na področju PC tehnologije so nas prepričale, da je težko obdržati korak s svetovnimi dosežki. Zahtevajo stalno razvojno jedro v skupini, kar je velika finančna obremenitev v primeru relativno omejenega tržišča. Zato smo se odločili, da uporabljamo orodja svetovno priznanih izdelovalcev in da z njihovo pomočjo implementiramo svoja tehnološka znanja na področju vodenja in informatizacije kombinirane proizvodnje električne energije.

Pri izbiri sistemske programske opreme je potrebno postaviti določene kriterije za medsebojno primerjanje posameznih izdelkov.

2.0. KRITERIJI ZA IZBIRO SISTEMSKE PROGRAMSKE OPREME

Pri izbiri sistemske programske opreme predlagamo sledeče kriterije /2,3/:

- enostavnost programiranja. Obvezna je možnost programiranja s pomočjo blokov, ki omogočajo izdelavo regulacijskih shem in brez velikega programerskega znanja. Poleg tega je pomembna izbira samih blokov kot možnosti, ki jih ponujajo (v primeru ponovitve enake regulacijske sheme in podobno),

- možnost izdelave lastnih algoritmov v C jeziku. Velika verjetnost je, da s pomočjo blokov ne bomo rešili vse probleme pri vodenju procesov. Zato mora obstajati možnost dodelave paketa, oziroma njegovo prilagajanje lastnim potrebam. V nasprotnem primeru so nujne pogoste intervencije proizvajalca, kar je časovno in ekonomsko nesprejemljivo,

- "prijaznost" (user friendly) do uporabnika. Program mora biti organiziran v obliki menijev, ki omogočajo enostavno izbiro posameznih funkcij,

- možnost spremembe vrednosti posameznih parametrov v samem delu oziroma sprotno prilagajanje nepričakovanim dogodkom,

- enostavna izdelava prikazov za animacijo procesa. Uporaba miške je obvezna,

- modularna zgradba. Ta omogoča izbiro posameznih zaželenih konfiguracij,

- hitrost dela. Po navadi je maksimalna hitrost vzorčenja v razponu med 0,1 in 1 sec. Vprašanje hitrosti je povezano z vprašanjem možnosti uporabe koprocesorskih modulov, ki imajo zmogljivost enokartičnih računalnikov kot so RIC, ARTIC in podobne. Te omogočjo tudi povečanje zanesljivosti delovanja, razširitev pomnilnika ter povečanje števila komunikacijskih kanalov,

- uporaba standardnih DOS funkcij. Nedvomno je, da je DOS operacijski sistem najbolj razširen na osebnih računalnikih in da je veliko že razvitega softwara po sprejemljivih cenah, ki so uporabni tudi za industrijske aplikacije (primer je LOTUS 123 in podobni paketi). Zato mora obstajati tako omenovano "DOS okno", v katerem lahko kličemo njegove standarne funkcije (date, time in podobno) ali pa zaganjamo zaželene aplikacije in analiziramo neto sprejete podatke,

- konfiguriranje sistema (izdelava podatkovne baze) mora biti enostavno. Zaželena je tudi možnost zaganjanja različnih baz na istem sistemu,

- število kontrolnih točk. Izbiramo paket z ozirom na število točk, ki jih podpira, pri čemer se zavedamo lastnih potreb,

- možnost povezovanja z velikimi računalniki, ki so zanimivi za simulacije in sprotne analize sprejetih podatkov iz procesa ter za njihovo arhiviranje,

- možnost delovanja v mreži. Vodenje prostorsko porazdeljenih procesov je vse bolj aktualno in paket mora podpirati NETBIOS standard /4/,

- možnost priključitve različne merno regulacijske periferije (OPTO 22, Burr Brown in podobno),

- cena programa oziroma razmerje cena/zmogljivost. Aplikacija, ki jo želimo delati je tisti presodni faktor, ki odloča, če je cena sistemskega softwara sprejemljiva ali ne.

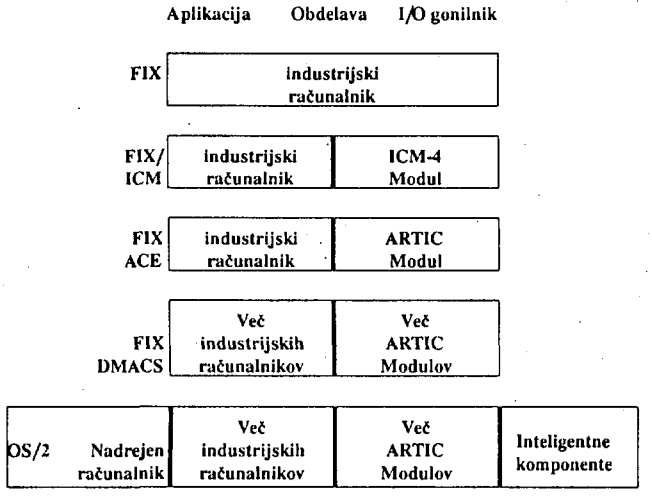
3.0. FIX SISTEMSKA PROGRAMSKA OPREMA

Z ozirom na naštete kriterije za izbiro sistemske programske opreme smo izbrali programski paket FIX za vodenje kombinirane proizvodnje električne energije in informatizacijo internih energetskihsistemov.

FIX sistemska programska oprema vsebuje module, ki delijo skupno podatkovno bazo in delajo pod večopravilnim jedrom. SAC (vzorčenje-scan, alarmiranje, vodenje-control) paket je zadolžen za vzdrževanje baze in je z njo in gonilnikom za mernoregulacijsko opremo po zagonu sistema naložen v dinamični pomnilnik. Še dva paketa predstavljata osnovo FIX-a. To sta DRAW in VIEW. DRAW omogoča izdelavo prikazov in jih povezuje z bazo podatkov. VIEW je zadolžen za komunikacijo z operaterjem (podatki se s SAC-om spravljajo v bazo, DRAW jih prenaša v ustrezen prikaz in VIEW jih posreduje operaterju). VIEW omogoča operaterju tudi nastavitev posameznih parametrov (npr. tarčo PID algoritma in podobno).

Večopravilno jedro se imenuje SHELL. Programom, katerim določa čas izvajanja, SHELL omogoča pristop DOS uslugam kot so I/O komunikacije, kar je zelo pomembno za uporabniške programe, ki delajo vzporedno z FIX-om. SHELL vsebuje dva glavna programa GO in MENU. GO omogoča nalaganje programa v pomnilnik in jih sproža po prioritetni shemi. MENU prikazuje uporabniku glavni meni, ki ga lahko poljubno spreminjamo in prilagajamo govornemu področju (npr. poslovenimo nadpise). Tretji program je okno v popularne DOS funkcije, kot so nastavitev časa in datuma in podobno. Zaradi povečevanja zanesljivosti delovanja računalniškega sistema SHELL preprečuje delovanje naslednjih sekvenc: Ctrl-Alt-Del (tako imenovani vroči zagon sistema), Ctrl-2 ter Ctrl-NumLock. Na sliki 3.1. je prikazan razvoj tega programskega paketa.

Osnovna verzija sistemske programske opreme se imenuje FIX. Obstaja možnost njegovega nakupa v različnih verzijah in to:



Slika 3.1. Razvoj programskega paketa FIX

- poenostavljena (oskubljena) verzija, ki se imenuje SPECIFIX. Nima možnosti uporabe statističnega modula, modula za izdelavo poročil, programskega bloka, koprocesorskih plošč. Obstaja tudi omejitev števila uporabljenih posameznih blokov,

- delovna verzija. Omogoča izvajanje vseh funkcij in sprotno spreminjanje posameznih parametrov. Možnost spreminjanja konfiguracije sistema in izdelave prikazov za animacijo procesa ne obstaja,

- razvojna verzija. Na prvi pogled identična kot delovna verzija s tem, da odklanja njene pomanjkljivosti.

- Za povečevanje hitrosti komuniciranja se uporablja inteligentni komunikacijski modul ICM-4, ki ima procesor INTEL 8088 za podporo do 4 RS 232 kanalov.

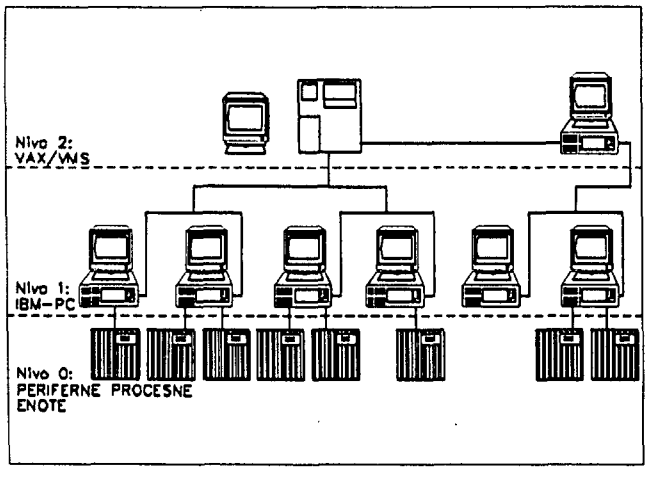
Za zahtevne aplikacije na samostojni računalniški postaji je najbolj primeren FIX ACE (application and control engine), ki uporablja zelo močan koprocesorski modul RIC ali ARTIC za shranjevanje baze, I/O gonilnika ter SAC programa. Hitrost delovanja (vzorčenja in vodenja) se lahko poveča na 1/10 sec. Poleg tega se tudi dobi dodatni pomnilniški prostor v DOS obsegu 0-640 KB za nove aplikacije (LOTUS 123 in podobno). Obstajajo različne verzije ARTIC (RIC) modula, ki se razlikujejo po tipu vodila (ISA ali Micro Channel), številu portov in velikosti dodatnega pomnilnika.

FIX DMACS (distributed manufacturing automation and control software) je prvi programski paket v svetu na osnovi osebnih računalnikov za vodenje porazdeljenih procesov. DMACS je NETBIOS kompatibilna mreža, ki zagotavlja prenose podatkov za delo v realnem času. To pomeni, da ni potreben prenos datotek preko serverja (ga ne potrebuje), kar je osnovna lastnost pisarniških mrež, ampak uporablja direktne klice NETBIOS programa. Komunikacija prek serverja lahko vnaša nesprejemljive zakasnitve. Ena postaja ima možnost izmenjave sporočil z največ 16 poljubnih postaj povezanih v mrežo. Hitrost prenosa je odvisna od uporabljene materialne opreme. DMACS je uspešno testiran v naslednjih mrežah: Token-ring, IBM PC Net, Ungermann Bass Net One, 3Com Ethernet in poljubni Ethernet ali Token passing, ki uporablja Novell Netbios program. Za najbolj zahtevne aplikacije je možna uporaba več koprocesorskih modulov RIC(ARTIC) ter EMS standard 3.2 ali več. DMACS je izdelan iz večih nivojev, kar zagotavlja maksimalno zmogljivost in fleksibilnost. To pomeni, da ne instaliramo na vsaki postaji kompleten programski paket, ampak le tisti del, ki ga potrebujemo, odvisno od funkcije postaje v mreži (npr. konfiguracijska postaja, kontrolna postaja, postaja za prikaze ali poljubna kombinacija naštetih). DMACS ima samo razvojno verzijo (delovna ne obstoja), ker mora imeti najmanj eno konfiguracijsko postajo.

DMACS je podobno kot FIX odprte zgradbe, ki omogoča uporabniku izdelavo lastnih algoritmov vodenja. Mreža je popolnoma transparentna kar pomeni, da uporabnik pristopa k podatkom na poljubni lokaciji v porazdeljenem sistemu brez vpogleda v mrežne operacije. DMACS lahko izmenjuje podatke z drugimi mrežnimi programskimi paketi (kot je IBM PC LAN), ki podpirajo NETBIOS standard. To pomeni, da imamo možnost uporabe namenskih obdelovalcev podatkovnih datotek značilnih za pisarniške mreže. Tako lahko integriramo porazdeljen sistem za informatizacijo (nadzor in vodenje) poljubnega procesa (tehnološkega ali energetskega) v enoten informacijski sistem podjetja. To je izrednega pomena, ker posamezni segmenti podjetja potrebujejo dokaj intenzivno izmenjavo določenih informacij.

Napovedan je tudi DMACS verzija VMS, ki bo integriral VAX/VMS zgradbo v DMACS preko DECnet mreže. Tako lahko dobimo kvalitetno orodje za CIM (computer integrated manufacturing). Verzija VMS podpira tehnologijo X-Okn, ki je razvita na MIT Institutu (Massachusetts Institute of Technology). Njena osnovna značilnost je standardizacija uporabniškega vpogleda v aplikacijo. Uporabnik, ki obvlada eno aplikacijo, se dokaj enostavno nauči naslednjo. Poleg tega omogoča tehnologija X-Okn

sočasno prikazovanje podatkov iz različnih aplikacij, kar olajša obdelavo podatkov na sistemskem nivoju (lahko naredimo tudi korelacijo med posameznimi mernimi točkami v sistemu). Na sliki 3.2. je prikazana ena možna varianta DMACS programa za VAX/VMS zgradbo.



Slika 3.2. DMACS za VAX/VMS zgradbo

Informacijski sistem smo organizirali v treh nivojih. Nivo 0 vsebuje kontrolne postaje kot so PLC kontrolerji, ki izvajajo direktno kontrolo procesa. Nivo 1 so PC/PS kompatibilni sistemi (AT 286 in zmogljivejši, model 50 ali več). Njihova naloga je izvajanje nadzornih funkcij kot so alarmiranje in sekvenčna kontrola ter izdelava sporočil. Nivo 2 je rezerviran za VAX (11/700, 8000 ali Micro VAX II). Njegova naloga je obdelava podatkov na sistemskem nivoju, izvajanje zahtevnejših proračunov in simulacijskih programov, arhiviranje podatkov in podobno.

Obstaja tudi verzija FIX-a, ki dela pod operacijskim sistemom OS/2 in uporablja vse njegove zmogljivosti (delo v oknih, komunikacija z velikimi računalniki kot je VAX in podobno).

4.0. ZAKLJUČEK

Pri izbiri sistemske programske opreme morajo biti postavljeni ustrezni kriteriji, na osnovi katerih se odločamo, katera je najbolj primerna za pričakovane aplikacije. Za vodenje kombinirane proizvodnje in informatizacijo internih energetskih sistemov smo izbrali programski paket FIX. Prve aplikacije so potrdile pravilnost narejene izbire in opravičenost postavljenih kriterijev.

5.0. LITERATURA

/1./ *D. Mrdaković, M. Tomšič, M. Vidmar:* Osnovne karakteristike distribuirane višemikroračunarske mreže DMS 860, Tehnika, 10, 1985,

/2./ *D. Mrdaković, P. Krajnik:* Izhodišča in smernice za vodenje in nadziranje prostorno porazdeljenega procesa v Tovarni zdravil Novo Mesto, IJS R-2 delovno poročilo, nov. 1988,

/3./ *D. Mrdaković, P. Krajnik:* Sistemska programska oprema za vodenje procesa v industriji, rokopis za MIPRO 90,

/4./ *D. Mrdaković, P. Krajnik:* Zasnova rahlo sklopljenega sistema za vodenje industrijskih procesov, Informatica št. 3, 1988.

RAZVOJ IN OPIS DVO-NIVOJSKEGA MODELA MORFOLOŠKE ANALIZE IN SINTEZE

INFORMATICA 3/90

Tomaž Erjavec
Laboratorij za računalniško
razumevanje jezika
Instutut »Jožef Stefan«, Ljubljana
Jamova 39, 61111 Ljubljana

Keywords: morphological analysis, synthesis, two-level model

POVZETEK: Članek obravnava dvo-nivojski, jezikovno neodvisni morfološki model Kimma Koskenniemia, ki je že implementiran na večih računalnikih in operativen za preko deset različnih jezikov. Predstavljena je uporabnost, razvoj in zgradba tega modela, ter podan opis dvo-nivojske gramatike.

ABSTRACT: THE DEVELOPMENT AND DESCRIPTION OF THE TWO-LEVEL MODEL OF MORPHOLOGICAL ANALYSIS AND SYNTHESIS. The paper discusses the two-level language independent morphological model of Kimmo Koskenniemi. This model has already been implemented on several different computers and is operational for over ten languages. The scope, evolution and structure of the model are presented, followed by the description of the two-level grammar.

1. Uvod

Računalniki vse bolj postajajo tudi orodje za procesiranje jezikovnih podatkov, od uporabe računalnika kot izboljšanega pisalnega stroja pa vse do računalniškega prevajanja besedil. Programi za računalniško obdelavo naravnega jezika se v različni meri ukvarjajo s raznimi plastmi jezikovne analize. Morfologija je najnižja raven tekstualne analize besedil, zaradi česar imajo morfološko komponento skorajda vsi programi za obdelavo naravnega jezika. To velja toliko bolj za morfološko bogate jezike, kot je slovenščina. Morfologija ali oblikoslovje je tisti del slovnice (Top84), ki se ukvarja z notranjo zgradbo besed in njihovo funkcionalno vlogo v stavku. Pristop, ki ga uporablja večina programov, je tim. morfološka analiza, ki vhodno besedo iz obdelovanega teksta razstavi v njene posamezne dele - morfeme (npr. "obkrožiti" v "ob-kroži-ti") in tej besedi pripiše njene morfo-sintaktične lastnosti (npr. glagol, nedoločnik, itd.).

Glavna naloga programov za morfološko analizo ali (in) sintezo je tako preslikava tekstualne (površinske) oblike besed v njihovo leksikalno (globinsko) obliko. Leksikalna oblika besed (njihova morfemska struktura) je zajeta v slovarju, po možnosti s pripadajočimi morfološkimi - sintaktičnimi - semantičnimi informacijami. Tekstualna ali površinska oblika besed pa je tista, ki se pojavi v pisanem besedilu. Tako:

lipama --> lip-a / samostalnik spol=ženski
število=dvojina sklon=dajalnik

2. Prepisovalna pravila in končni avtomati

Direktno preslikavo med tekstualnim in leksikalnim nivojem onemogočajo fono-morfološke in morfološke alternacije, torej "nepravilnosti", ki se pojavljajo pri pregibanju raznih besed (npr. "telesa --> telo-a").

S formalizmom *generativne fonologije*, razvitim v 60-ih letih, lahko opišemo morfologijo kateregakoli jezika, tako da konstruiramo množico *prepisovalnih pravil* (rewriting rules). S temi pravili leksikalno reprezentacijo korak za korakom transformiramo, dokler ne dobimo površinske (tekstualne) reprezentacije. Pravila za prepisovanje imajo naslednjo obliko:

x-->y / a__b *(1)*

x naj se prepiše v y, če je v kontekstu a__b, torej:

axb --> ayb *(1.1)*

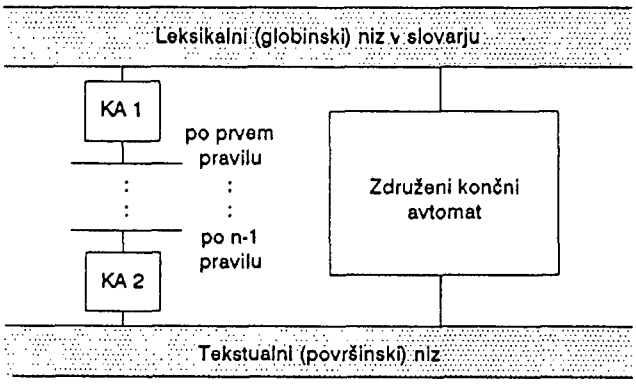
Generativni formalizem deluje samo v eni smeri (leksikalna --> tekstualna beseda), razen tega pa je računsko zahteven, saj imajo taka prepisovalna pravila v splošnem moč Turingovih strojev. To dvoje je zadosten razlog, da so ga uporabljali le redki morfološki programi.

Martin Kay in Ronald Kaplan (Kay82) sta ugotovila, da je prepisovalna pravila možno realizirati s *končnimi avtomati*, pod pogojem, da se pravila ne aplicirajo rekurzivno. Ker v morfologiji ni pravil rekurzivnega tipa, je bilo to omejitev smiselno sprejeti, saj so končni

avtomati implementacijsko enostavni in učinkoviti ter delujejo v obeh smereh (analiza, sinteza).

Takšen avtomat (pravilo) bi torej primerjal dva sosednja nivoja generativnega sestava: nivo neposredno pred apliciranjem pravila in nivo po njegovem apliciranju. Celotna morfološka "sintaksa" bi tako bila kaskada takšnih avtomatov.

Kaskade končnih avtomatov kot take ni možno implementirati, vendar pa obstaja algoritem, ki kaskado vseh avtomatov združi v enega (sl. 1). Ta, večji, avtomat je funkcionalno identičen kaskadi, čeprav v njem ni več mogoče identificirati posameznh pravil.



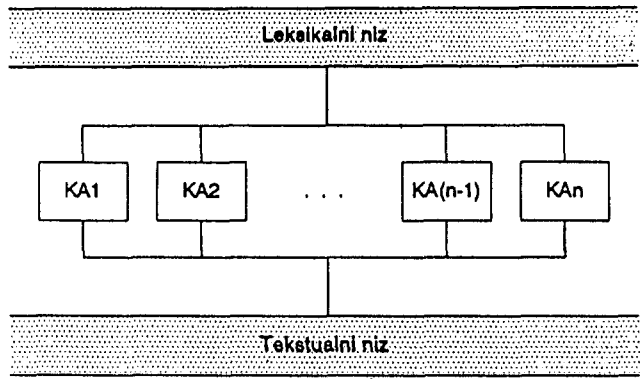
Slika 1. - Model Kay & Kaplan

Ta rešitev je na prvi pogled idealna, saj je rezultirajoči avtomat operativen, učinkovit in dvosmeren. Vendar pa ima pristop Kay & Kaplan hudo pomankljivost: velikost rezultirajočega avtomata je za opis jezikov z bogato morfologijo prevelika.

3. Dvo-nivojski model

Implementacijsko rešitev zgornjega problema ter novo teorijo in formalizem za opisovanje besedotvorja (pregibanja, derivacije in združevanja besed) je razvil Kimmo Koskenniemi (Kos84, Kos85) s svojim *dvo-nivojskim morfološkim modelom*. Dvo-nivojski model se od Kay & Kaplan modela razlikuje predvsem v opustitvi kaskade pravil in s tem tudi potrebi po združevanju avtomatov. Vsa pravila tu operirajo *neposredno* med leksikalnim in tekstualnim nizom. Pravila so zaradi tega nujno drugačna od abstrakno-fonoloških. Predvsem so omejena na variacije znotraj enega segmenta; vlogo pravil, ki smo se jim s tem odpovedali, prevzame sistem slovarja.

V dvo-nivojskem modelu sta torej samo dve reprezentaciji: leksikalna (globinska) reprezentacija in tekstualna (površinska) reprezentacija. Vsa pravila delujejo *paralelno* in testirajo ujemanje leksikalnega in tekstualnega niza, znak za znak (sl. 2). Pravila so realizirana kot transduktorji, ki so identični končnim avtomatom, le da kot vhod jemljejo pare simbolov (v našem primeru par "leksikalni niz : tekstualni niz"). Transduktorji so dvosmerni, tako da lahko tekstualni niz generirajo iz leksikalnega ali ga analizirajo v leksikalnega.



Slika 2. - Model Koskenniemi

Pravila določajo *legalne korespondence* leksikalnega in tekstualnega znaka v določenem kontekstu. Uporabljajo se za pokrivanje fono-morfoloških alternacij, ki nastanejo pri spajanju posameznih morfemov v besede. V grobem imajo pravila obliko:

$$a : b <=> x_1 : x_2 \_ y_1 : y_2 \qquad (2)$$

Leksikalni "a" se realizira kot tekstualni "b", če in samo če je njun levi kontekst par "x_1:x_2" in desni kontekst par "y_1:y_2". To pravilo bi tako ugotovilo korespondenco:

Leksikalni niz: . . . x_1 *a* y_1 . . .
Površinski niz: . . . x_2 *b* y_2 . . .

Morfološki analizator oz. sintetizator, ki uporablja dvo-nivojska pravila, torej ugotavlja ali je leksikalni niz (beseda, kot je zapisana v slovarju) v pravilni korespondenci s tekstualnim ali površinskim nizom. Če programu podamo nedefiniran tekstualni niz, bo kot "stranski učinek" niz sintetiziran; iz leksikalne predstavitve bo nastala površinska. V obratnem primeru, ko pustimo nedoločen leksikalni niz, se bo program obnašal kot morfološki analizator - tekstualno besedo bo analiziral v leksikalno. Za analizo potrebujemo poleg samih pravil še slovar legalnih leksikalnih besed, zapisan v primerni obliki.

4. Primer dvo-nivojske gramatike

Kot primer podajmo majhen a popolen primer dvo-nivojske gramatike povzete po (Dar87). Gramatika, oz. datoteka s pravili sestoji iz štirih delov z naslovi "Abeceda", "Množice", "Definicije" in "Pravila".

```
Abeceda
 A:a O:o U:y
 a b c d e f g h I j k l m n o p q r s t u v x y w z ;
```

```
Množice
Soglasnik = b c d f g h j k l m n p q r s t v w x z ;
ZadnjiSam = a o u ;

Definicije
NeSprednjiKontekst =
 :Soglasnik* ( :ZadnjiSam | :i | :e )* :Soglasnik* ;

Pravila
 "Samoglas-sklad"
 Vx:Vy <=> :ZadnjiSam NeSprednjiKontekst* __ ;
           kjer Vx v (A O U)
                Vy v ZadnjiSam
           vezane;
```

Ta gramatika vsebuje eno samo pravilo z imenom "Samoglas-sklad". Pravilo zahteva realizacijo leksikalnega A, O in U kot a, o in u po pojavitvi površinskega a, o ali u, pod pogojem, da je katerikoli vmesni površinski samoglasnik bodisi zadnji samoglasnik ali pa i oz. e. Na osnovi tega primera si bolj natančno poglejmo format dvo-nivojskih gramatik.

V dvo-nivojski gramatiki ločimo dve *abecedi*: *leksikalno* abecedo in *površinsko* abecedo. Nekateri znaki so prisotni samo v eni od obeh abeced, večina znakov pa pripada obema abecedama. Abeceda v pravilih je sestavljena iz seznama *veljavnih parov*. Ti pari so zapisani v obliki "*x:y*", kjer je "x" element leksikalne abecede, "y" pa element površinske abecede, dvopičje pa se uporablja kot ločilo. Veljavni pari so vsi pari, ki so eksplicitno določeni z abecedo (npr. "N:n", ne pa ":m"), ali pa dovoljeni z definicijo ali pravilom. Namen abecede na začetku gramatike je, da deklarira popoln seznam obeh abeced in delen seznam veljavnih parov. Znak brez dvopičja na katerikoli strani predstavlja par identičnih znakov.

Pri interpretaciji pravil privzamemo še poseben par znakov. Leksikalni del tega para je znak za besedno mejo, površinski del pa "prazen" znak. Sistemsko sta ta dva znaka nastavljena na "#" za besedno mejo in "0" za prazen znak. Seznamu veljavnih parov se seveda avtomatično doda "#:0".

Del gramatike z *množicami* vsebuje (če seveda obstaja) deklaracije, ki povezujejo imena s seznami znakov abecede. Množice imajo enako vlogo kot distinktivne oznake v drugih fonoloških formalizmih. Z uvedbo množic pravila pridobijo na moči generalizacije, saj lahko v pravilu uporabimo ime množice namesto eksplicitnega naštevanja znakov. Ime množice lahko uporabimo bodisi na leksikalni ali pa na površinski strani para, npr. "Samoglasnik:0". Interpretacija imen množic v takih parih je odvisna od tega, na kateri strani para stojijo. Če na leksikalni strani para uporabimo množico, ki vsebuje tako leksikalne kot površinske pare, bodo upoštevani samo tisti elementi množice, ki pripadajo leksikalni abecedi. Podobno velja za površinske množice, če pa v pravilu uporabimo ime množice brez podpičja, se bo tak zapis interpretiral kot par.

Naslednji pomožni del gramatike so tim. *definicije*, ki omogočajo še en način za izražanje posplošitev. Kontekst, ki ga deli več pravil, je lahko definiran samo enkrat, nato pa se nanj sklicujemo z imenom. Tako kot pri množicah, je tudi del z definicijami sestavljen iz seznama deklaracij; razlika je v tem, da so definicije regularni izrazi. Dovoljene operacije v izrazih definicij (in tudi pravil) so *konkatenacija, disjunkcija, Kleenejeva zvezda, opcionalnost* in *plus*. Definicija "NeSprednji-Kontekst" označuje vse nize, ki se začenjajo z pod-nizom nič ali večih znakov, ki imajo na leksikalni strani samoglasnik, nato sledi nič ali več znakov, ki imajo na leksikalni strani bodisi element iz množice "ZadnjiSam" ali "i" ali "e" in nato še poljubno ponavljanj znakov s soglasnikom na leksikalni strani.

Zadnji razdelek gramatike so *pravila*. Dvo-nivojsko pravilo sestoji iz imena v narekovajih, čemur sledi *korespondenca, operator pravila*, poljubno število *okolij*, in (če se v pravilu uporabljajo spremenljivke) *prireditvenega dela*. Okolje je sestavljeno iz *levega* in *desnega konteksta* (oba sta lahko prazna), ki ju loči "__". Obstajajo štirje operatorji pravil: "<=>", "=>", "<=" in "/<=". Njihov pomen bomo ilustrirali z nekaj enostavnimi zgledi. Za začetek enostaven zgled pravila leve implikacije:

```
"Ozvočenje 1"
 k:g <= Samoglasnik__Samoglasnik ;
```

Vsako dvo-nivojsko pravilo se ukvarja z določeno površinsko realizacijo nekega leksikalnega znaka v določenem okolju; v našem primeru je to realizacija "k" med dvema samoglasnikoma. Operator "<=" zahteva realizacijo "k" kot "g" v tem okolju. Z drugimi besedami se pravilo glasi: "k" *mora biti* realiziran kot "g" med dvema samoglasnikoma. Pravilo ne omejuje realizacije "k" v drugih okoljih; dopušča torej realizacijo "k" kot "g" kjerkoli. Če bi uporabnik hotel zagotoviti, da se bodo samo med-samoglasniški "k"-ji realizirali kot "g", bi zapisal pravilo z operatorjem "=>":

```
"Ozvočenje 2"
 k:g => Samoglasnik__Samoglasnik ;
```

Pravilo "Ozvočenje 2" je ekvivalentno zahtevi "k" je realiziran kot "g" *samo* med dvema samoglasnikoma. To pravilo ne zahteva, da so med-samoglasniški "k"-ji realizirani kot "g"; pravilo to samo dopušča in omejuje takšno realizacijo na to specifično okolje.

Pravili 1 in 2 lahko združimo v eno samo pravilo z operatorjem "<=>":

```
"Ozvočenje 3"
 k:g <=> Samoglasnik__Samoglasnik ;
```

Z drugimi besedami: "k" je realiziran kot "g" *vedno in samo* med dvema samoglasnikoma. Takšno združevanje pravil je lahko koristno, ni pa nujno, saj je skupen učinek pravil 1 in 2 enak učinku zgornjega pravila. Glavna prednost "=>" operatorja je v tem, da dopušča

neko realizacijo leksikalnega znaka v določenem okolju ne da bi zahteval, da mora biti v danem kontekstu vedno tako realiziran.

Četrti operator "/<=" omogoča določanje prepovedi:

```
"Ozvočenje 4"
 k:g /<= Soglasnik_Samoglasnik ;
```

Pravilo določa, da "k" *nikoli* ni realiziran kot "g" med soglasnikom in samoglasnikom. Operator prepovedi je koristen v situacijah, v katerih je laže opisati okolja v katerih se nekaj ne dogaja, kot pa našteti pozitivne kontekste.

Pravilo ima lahko tudi več okolij:

```
"Ozvočenje 5"
 k:g <=> Samoglasnik_          ;
         _Samoglasnik ;
```

Pravilo 5 zahteva realizacijo "k" kot "g" pred in po samoglasniku, in to samo v teh dveh okoljih. To pravilo je kompatibilno s pravilom 1 - pravilo 1 celo naredi redundantno - ne bi pa pravilno delovalo skupaj s pravili 2, 3 in 4. Ozvočenje 5 zahteva pred-samoglasniško realizacijo "k" kot "g", ne glede na to, kateri znak se nahaja pred njim in je tako lahko v nasprotju z ozvočenjem 4, ki prepoveduje tako realizacijo pred soglasnikom. Ravno tako je v nasprotju s praviloma 2 in 3, ki omejujeta ozvočenje "k" na med-samoglasniške kontekste.

Kot zadnji primer zvrsti pravil, ki jih lahko izrazimo v dvo-nivojskem formalizmu, si poglejmo še pravilo, ki realizira "k" kot "g" med dvema enakima samoglasnikoma. Tak tip pravila lahko zapišemo s pomočjo spremenljivk:

```
"Ozvočenje 6"
 k:g <=> Vx_Vx ; kjer Vx v Samoglasnik ;
```

Znak "Vx" je tu uporabljen kot spremenljivka; spremenljivki priredimo vrednost v "kjer" delu pravila, ki mora biti od telesa pravila ločen s podpičjem. To pravilo je ekvivalentno pravilu z večimi okolji: "a_a", "e_e", itd. za vsak samoglasnik. Spremenljivke ne prispevajo k moči formalizma, omogočajo pa zapis precejšnjega števila pravil na bolj koncizen način. Več spremenljiv lahko med seboj tudi "povežemo". V našem primeru vidimo primer uporabe vezave, ki zahteva, da lahko spremenljivke zavzamejo samo po indeksu enake vrednosti;tako se paru "Vx:Vy" lahko priredijo samo vrednosti "A:a", "O:o" in "U:u".

5. Implementacije

Dvo-nivojski model je doživel že številne implementacije, začenši z pascalsko za Burroughs B7800 in kasneje za DEC-20 in IBM-PC, pet različnih implementacij v verzijah LISP-a in še varianto v prolog-u (DEC-20, Bear & Pereira). Implementacijo modela v Interlisp-D za Xerox 1100 delovne postaje (ena zadnjih) sestavljata (Dal87) dva programa:

* DKIMMO se lahko uporablja za testiranje morfoloških pravil nad določenim slovarjem oziroma kot morfološki "front-end" za druge programe s področja obdelave naravnega jezika. Deluje lahko kot morfološki analizator ali sintetizator. Uporablja transduktorje, ki jih generira TWOL.

* TWOL je prevajalnik pravil v transduktorje, ki se hranijo v tabelarični obliki. Omogoča popolnoma interaktivno okolje (okna / miška) za razvijanje pravil nekega jezika. Pri dodajanju novih pravil v bazo (ali spreminjanju obstoječih) pazi na možne logične konflikte z že obstoječimi pravili.

Z dvo-nivojskim formalizmom je operativno opisanih (oz. je delo na njih še v teku) že lepo število jezikov: finščina, angleščina, nemščina, flamščina, japonščina, romunščina, francoščina, švedščina, grščina, laponščina, arabščina, islandščina, stara cerkvena slovanščina in stara akadijščina. V tem formalizmu je bila opisana tudi že srbo-hrvaščina in del slovenščine (Erj89).

6. Literatura

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

(Erj89) Erjavec T., Tancig P.: **Dvo-nivojska pravila za alternacije slovenskih samostalniških sklanjatev** v zborniku V. kongresa zveze društev za uporabno jezikoslovje Jugoslavije; Ljubljana 1989

(Kay82) Kay M.: **When meta-rules are not meta-rules** v Sparck-Jones & Wilks (eds.): Automatic natural language processing; University of Essex, Cognitive Studies Centre (CSM-10), 1982

(Kos84) Koskenniemi K.: **A General Computational Model for Word-Form Recognition and Production** v Computational Linguistics, Conference Proceedings; COLING 1984

(Kos85) Koskenniemi K.: **A General Computational Model for Word-Form Recognition and Production** v Computational Morphosyntax, Report on Research 1981-84; University of Helsinky, Dept. of General Lingusistics; Publications No. 13, 1985

(Top84) Toporišič J.: **Slovenska slovnica** Založba Obzorja Maribor, 1984

O ZNANJU V PORAZDELJENIH SISTEMIH

INFORMATICA 3/90

Keywords: logics of knowledge, distributed system, protocol, knowledge-based protocol, information-based approach

Tatjana Kapus
Tehniška fakulteta Maribor,
Smetanova 17, Maribor

Povzetek: V članku najprej predstavimo definicijo logike znanja v modelu možnih svetov, znano iz filozofije in umetne inteligence. Nato pa prikažemo nekaj primerov, kako je mogoče pojem znanja formalno vpeljati v porazdeljene sisteme in ga koristno uporabiti za opis in sklepanje o dogajanju v njih. Govorimo o informacijsko osnovanem pojmu znanja.

On Knowledge in Distributed Systems: The possible-worlds approach to definition of logic of knowledge, as known from philosophy and artificial intelligence, is presented. It is then shown by examples how knowledge can be formally introduced into distributed systems, and how it can be usefully employed for describing the systems and for reasoning about them. The information-based notion of knowledge is used.

1. UVOD

Značilno za procese v porazdeljenih sistemih je, da so precej neodvisni drug od drugega. Vsak proces ima dostop do svojega stanja, nima pa neposrednega dostopa do stanj drugih procesov. Med seboj komunicirajo z izmenjavo sporočil. Lahko rečemo, da si procesi s tem nabirajo znanje, se učijo o stanjih drugih procesov med izvajanjem sistema. Spomnimo se porazdeljenih algoritmov za odkrivanje zagate ali algoritmov usmerjanja v računalniških mrežah. Pri načrtovanju protokolov za prenos podatkov prek nezanesljivega kanala na primer intuitivno sklepamo, da mora sprejemni proces poslati oddajnemu potrditev sprejema, da bo ta vedel, ali lahko pošlje novo sporočilo ali pa naj ponovno pošlje prejšnje. V zadnjih letih se zato uveljavlja mnenje, da je treba o porazdeljenih sistemih formalno, načrtno in ne le priložnostno intuitivno sklepati na osnovi tega, kako se v njih spreminja znanje procesov.

Formalni pristopi k analizi porazdeljenih sistemov se v osnovi med seboj razlikujejo po tem, kakšen formalen model sistema imajo v ozadju. Težko bi rekli, da obstaja "najboljši" model. Vsak je primeren za določeno vrsto analize /4/. V modelih večinoma zasledimo dva ključna pojma: stanja in akcije. Imejmo preprost porazdeljen sistem z enim samim procesom, ki izvaja sekvenčen program. Lahko ga predstavimo z množico možnih obnašanj, zaporedij naslednje oblike:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$

$s_0, s_1, s_2, \ldots$ so stanja, $a_0, a_1, a_2, \ldots$ pa imenujemo akcije, dogodki ali operacije /5/. Proces je tu avtomat, ki je vedno v enem od (morda neskončno mnogo) notranjih stanj. Akcija je pretvornik stanja. Tak način opisa je mogoče razširiti tudi na sisteme z več procesi. V modelu lahko damo pomembnejšo vlogo stanjem /6/ ali pa sisteme opisujemo samo z akcijami /7/.

Najprej bomo podali standardni način formalne definicije semantike znanja v filozofiji in umetni inteligenci /2/. Tu je logika znanja uvedena v model možnih svetov ali stanj. Nato bomo pokazali, da lahko tudi porazdeljen sistem predstavimo s sorodnim modelom in vpeljemo logiko znanja na enak način. Omenili smo že, da pri načrtovanju protokola za prenos prek nezanesljivega kanala sklepamo, da oddajnik potrebuje potrditev o sprejemu, da se lahko odloči, kaj bo storil v naslednjem koraku. Njegove akcije v nekem stanju so neposredno odvisne od njegovega znanja v tem stanju. Pokazali bomo, kako se lahko v nasprotju z običajnim, t.i. standardnim protokolom, s protokolom na osnovi znanja *neposredno formalno izražamo o odvisnosti* akcij od znanja procesa, torej na način, ki se ujema z razmišljanji načrtovalca protokola. Model in na novo uvedene pojme v zvezi z znanjem bomo ponazorili z "uganko o nezvestih možeh".

Nazadnje bomo pokazali še primer uvedbe znanja v porazdeljene sisteme, modelirane samo z zaporedji dogodkov. Na primeru protokola za medsebojno izključevanje procesov pa bomo spoznali, kako je varnost in živost algoritmov povezana z znanjem.

Povejmo še, da v članku govorimo o zunanjem ali informacijsko osnovanem pojmu znanja v porazdeljenem sistemu. To pomeni, da ne jemljemo, da procesi dejansko razmišljajo,

ampak jim znanje pripisujemo sami oziroma načrtovalec sistema.

2. MODEL MOŽNIH SVETOV IN LOGIKA ZNANJA

Ideja pri uvedbi znanja v model možnih svetov je, da je poleg dejanskega stanja stvari še mnogo drugih mogočih stanj stvari ali možnih svetov. Mogoče je, da osebek nekaterih izmed teh možnih svetov ne more razlikovati od dejanskega sveta. Pravimo, da osebek ve dejstvo φ, če je pravilno v vseh svetovih, za katere meni, da so možni.

Za formalni opis vpeljemo jezik, ki mu pravimo logika znanja. Poleg običajnih oznak $\wedge$, $\neg$, $\vee$ iz izjavne logike potrebujemo še oznake za izražanje o znanju. Zato dodamo jeziku modalne operatorje K_1, ..., K_n. Formula oblike $K_i\varphi$ pove, da "osebek i ve φ".

Imejmo množico enostavnih izjav Φ in formulo *true*. Množica naj bo zaprta za negacijo, konjunkcijo in modalne operatorje K_1, ..., K_n. Če sta φ in ψ formuli, so tako formule tudi $\neg\varphi$, $\varphi \wedge \psi$ in K_i, $i = 1, \ldots, n$. $\Rightarrow$ in $\vee$ definiramo s pomočjo $\neg$ in $\wedge$. *false* pa je krajši zapis za $\neg$*true*. Opazimo, da govorimo o več osebkih, saj imajo modalni operatorji indekse. To nam bo koristilo pri uvedbi znanja v porazdeljene sisteme, kjer bodo osebki predstavljali procese.

Ena od poti za formalno definicijo semantike jezikov z modalnimi operatorji je uporaba Kripkejevih struktur. Bodi M Kripkejeva struktura $(S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_n)$. Tu je S množica stanj ali možnih svetov, π je prireditev logičnih vrednosti enostavnim izjavam p za vsako stanje $s \in S$ (tako da je $\pi(s,p) \in \{$true, false$\}$ za vsako stanje s) in $\mathcal{B}_i$, $i = 1, \ldots, n$, je binarna relacija na S, ki je refleksivna, tranzitivna in simetrična. Torej je ekvivalenčna relacija. Od lastnosti $\mathcal{B}_i$ je odvisno, kakšni aksiomi bodo veljali.

Definirajmo relacijo $\vdash$. Izraz $M,s \vdash \varphi$ pravi: "φ je pravilna ali izpolnjena v stanju s strukture M". Velja:

$M,s \vdash$ *true*,

$M,s \vdash p$, če in samo če $\pi(p,s)$ = true,
(p je enostavna izjava),

$M,s \vdash \neg\varphi$, če in samo če $M,s \not\vdash \varphi$,

$M,s \vdash \varphi \wedge \psi$, če in samo če $M,s \vdash \varphi$ in $M,s \vdash \psi$,

$M,s \vdash K_i\varphi$, če in samo če
za vsak t, $(s,t) \in \mathcal{B}_i$, velja $M,t \vdash \varphi$.

Zadnja izjava nam pove, da osebek i ve φ natanko tedaj, če je φ pravilna v vseh svetovih, za katere i meni, da so možni.

Pravimo, da je formula φ *veljavna v strukturi* M, če $M,s \vdash \varphi$ za vsa stanja s v M. φ je *izpolnljiva v* M, če $M,s \vdash \varphi$ za neko stanje s v M. φ je *veljavna*, če je veljavna v vseh Kripkejevih strukturah, in φ je *izpolnljiva*, če je izpolnljiva v neki Kripkejevi strukturi.

Naš pojem znanja je mogoče opredeliti z naslednjim skladnim in popolnim sistemom aksiomov, ki mu pravijo S5:

Vse izjavne tavtologije. (A1)

$K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\psi$ (A2)

$K_i\varphi \Rightarrow \varphi$ (A3)

$K_i\varphi \Rightarrow K_iK_i\varphi$ (A4)

$\neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$ (A5)

$$\frac{\varphi,\ \varphi \Rightarrow \psi}{\psi}$$ (modus ponens) (R1)

$$\frac{\varphi}{K_i\varphi}$$ (R2)

(A1) in (R1) sta podedovana iz izjavne logike. (A2) pravi, da je znanje osebka zaprto za logično implikacijo. (A3) pove, da osebek lahko ve samo pravilne izjave. (A4) in (A5) sta aksioma samoopazovanja ali introspekcije. Pravita, da osebek natanko ve, kaj ve in česa ne ve.

Aksiomi (A3), (A4) in (A5) so veljavni prav zato, ker smo vzeli, da so relacije $\mathcal{B}_i$ refleksivne, tranzitivne in simetrične. Če bi na primer zahtevali samo, naj bo relacija serijska, ne pa tudi refleksivna (vsaka refleksivna relacija je tudi serijska) bi dobili aksiome za znanju soroden pojem - zaupanje. Ne moremo reči, da osebek zaupa (ali verjame) samo pravilnim izjavam. Zato bi namesto (A3) veljal šibkejši aksiom $\neg L_i$(*false*), ki pravi samo, da osebek ne more verjeti v "popolno laž" (L_i je modalni operator za zaupanje).

Na veljavnost (A2) in (R2) pa ne moremo vplivati samo s preprosto spremembo lastnosti $\mathcal{B}_i$. V vsakem primeru osebek ve vse veljavne formule in njegovo znanje je zaprto za implikacijo. To pomeni, da je osebek logično vseveden, "popoln mislec" z neskončno računsko močjo, česar pa ne bi mogli reči za osebke v realnosti. Obstajajo tudi bolj realistične logike /2/, tu pa bomo pokazali, kako si lahko načrtovalec porazdeljenega sistema pomaga s prvo. Seveda pa velja, da je vsaka logika in torej vsak pojem znanja primeren za določeno vrsto raziskav.

Našo logiko znanja lahko razširimo s pomembnim pojmom skupnega znanja. Predstavljamo si, da ima skupina G skupno znanje o dejstvu φ, če vsak v skupini G ve φ, vsak v G ve, da vsak v skupini G ve φ,... Dodajmo logiki še dva operatorja, E_G in C_G za vsako podskupino osebkov G. Preberemo ju "vsak v skupini G ve φ" in "φ je skupno znanje v skupini G".

$(M,s) \vdash E_G\varphi$, če in samo če za vsak $i \in G$,
$(M,s) \vdash K_i\varphi$,

$(M,s) \vdash C_G\varphi$, če in samo če za vsak $k \geq 1$,
$(M,s) \vdash E_G^k\varphi$,

kjer je $E_G^1\varphi$ okrajšava za $E_G\varphi$
in $E_G^{k+1}\varphi$ okrajšava za $E_GE_G^k\varphi$.

Tako razširjenemu jeziku dodamo še aksiome:

$$E_G\varphi \equiv \bigwedge_{i \in G} K_i\varphi$$ (A6)

$(C_G\varphi \wedge C_G(\varphi \Rightarrow \psi)) \Rightarrow C_G\psi$ (A7)

$C_G\varphi \equiv E_G(\varphi \wedge C_G\varphi)$ (A8)

$$\frac{\varphi \Rightarrow E_G\varphi}{\varphi \Rightarrow C_G\varphi}$$ (R3)

(A6) podaja semantiko operatorja E_G. (A7) ustreza analogni lastnosti znanja. (A8) imenujemo aksiom fiksne točke. Izraža dejstvo, da je $C_G\varphi$ rešitev enačbe s fiksno točko $X \equiv E_G(\varphi \wedge X)$, pravzaprav njena največja rešitev.

Pravilo sklepanja (R3) imenujemo indukcijsko pravilo, ker z uporabo dejstva, da velja $\varphi \Rightarrow E_G\varphi$, z indukcijo po k lahko pokažemo, da velja $\varphi \Rightarrow E_G{}^k\varphi$ za vse $k \geq 1$.

Jeziku je mogoče dodati tudi časovne operatorje, vendar bomo o tem govorili pri interpretaciji znanja v porazdeljenih sistemih.

V porazdeljenih sistemih je mogoče formalno definirati znanje tako, da se definicija ujema s prej predstavljeno definicijo znanja v Kripkejevih strukturah. To storimo na naslednji način. Vzemimo, da je vsak proces v sistemu v nekem lokalnem stanju, sistem pa v nekem globalnem stanju. Z $s(i)$ označimo lokalno stanje procesa i v globalnem stanju s. Pravimo, da proces ve φ v globalnem stanju s, če je φ pravilna v vseh globalnih stanjih s', za katera velja $s(i) = s'(i)$. Globalna stanja sistema ustrezajo možnim svetovom Kripkejeve strukture. Če $\mathcal{B}_i$ definiramo tako, da $(s, s') \in \mathcal{B}_i$ natanko tedaj, kadar $s(i) = s'(i)$, lahko ugotovimo, da je $\mathcal{B}_i$ ekvivalenčna relacija nad globalnimi stanji. Tako vpeljani pojem znanja v porazdeljenih sistemih zato zadošča sistemu aksiomov S5, aksiomom klasične logike znanja.

3. MOŽNI SVETOVI IN ZNANJE V PORAZDELJENEM SISTEMU

V tem poglavju si bomo ogledali formalen model, v katerega je znanje vpeljano na pravkar omenjeni način. Njegova avtorja Halpern in Fagin /3/ sta želela dovolj splošen model, ki bi omogočal predstavitev sistemov z asinhronim prehajanjem sporočil, paralelnih sistemov s skupnim pomnilnikom, pa tudi sistemov ljudi ali robotov, ki med seboj komunicirajo. Povejmo, zakaj je model zelo zanimiv za načrtovalca sistema. Protokol lahko definiramo kot funkcijo iz lokalnega stanja procesa v akcije. Ta definicija je zelo splošna in gotovo zajema vse protokole, ki jih je mogoče opisati z današnjimi programskimi jeziki. Vendar s takšnimi standardnimi protokoli načrtovalec ne more naravno opisati situacij, kjer so akcije procesa izrecno odvisne od njegovega znanja. Zato bomo definirali protokol na osnovi znanja. Ti protokoli nam omogočajo neposredno opisovanje odnosa med znanjem in akcijami. Zato predstavljajo primeren visokonivojski opis, kaj naj proces naredi v določeni situaciji.

3.1 Model

Sistem opredelimo z množico možnih tekov; tek je funkcija iz časa v globalna stanja - opis časovnega obnašanja sistema. Tek preprostega porazdeljenega sistema z enim samim sekvenčnim procesom lahko predstavimo z neskončnim zaporedjem stanj in akcij, ki vodijo od enega do drugega stanja.

Če je v sistemu n procesov, k stanju sistema gotovo prispeva stanje vsakega procesa, ker pa nas morda zanima še kaj drugega (npr. sporočila, čakajoča v vrstah...), sta se avtorja odločila, naj sistem sestavljata dva dela: procesi in okolje. Globalno stanje sistema z n procesi je potem $(n+1)$-terica $(s_e, s_1, \ldots, s_n)$, kjer je s_e stanje okolja, s_i pa (lokalno) stanje procesa i. Kako razdelimo sistem na procese in okolje, je odvisno od sistema, ki ga analiziramo.

Kot pri enem procesu akcije spreminjajo tudi globalno stanje sistema, vendar tu na akcije posameznih procesov ne moremo gledati izolirano. Akcije različnih procesov namreč lahko med seboj sodelujejo. Zato definirajmo skupno akcijo $(a_e, a_1, \ldots, a_n)$, kjer je a_e akcija, ki jo izvede okolje, akcijo a_i pa izvede proces i. τ naj bo funkcija, ki priredi pretvornik globalnega stanja vsaki skupni akciji. Potem je $\tau(a_e, a_1, \ldots, a_n)(g)$ globalno stanje, rezultat akcij $a_e, a_1, \ldots, a_n$, izvedenih hkrati v okolju in procesih, ko je bil sistem v glavnem stanju g. Vidimo, da akcije lahko izvaja tudi okolje. Če želimo povedati, da proces ali okolje ne izvede akcije, rečemo, da izvede ničelno akcijo. Ne bomo privzeli, da so akcije atomične.

Tek sistema je zaporedje globalnih stanj, povezanih z usmerjenimi povezavami, označenimi s skupno akcijo. Izkaže se, da za mnoge aplikacije lahko akcije zanemarimo (in tudi jih) in "zbrišemo" iz predstavitve. Formalno jemljemo, da je tek funkcija iz "realnega časa" v globalna stanja. Zaradi preprostosti jemljemo, da je "realni čas" množica naravnih števil, lahko pa bi vzeli tudi realna števila ali kak drug linearen red. Ne jemljemo, da imajo procesi nujno dostop do realnega časa; če ga imajo, ga preprosto jemljemo kot del njihovih lokalnih stanj.

Bodi L_e množica mogočih (lokalnih) stanj okolja, L_i, $i = 1, 2, \ldots, n$ pa množice lokalnih stanj procesov. $\mathcal{G} \subseteq L_e \times L_1 \times \ldots \times L_n$ je množica globalnih stanj. Tek nad $\mathcal{G}$ je funkcija iz naravnih števil v $\mathcal{G}$ (tek nad $\mathcal{G}$ torej lahko jemljemo kot zaporedje globalnih stanj). Z $r(m)$ označimo globalno stanje sistema ob času m v teku r. Če $r(m) = (s_e, s_1, \ldots, s_n)$, potem $r_i(m) = s_i$ za $i = 1, \ldots, n$. Paru (r,m) recimo točka. $r_i(m)$ je lokalno stanje procesa i v točki (r,m). Sistem nad $\mathcal{G}$ je množica tekov nad $\mathcal{G}$. Pravimo, da je (r,m) točka v sistemu $\mathcal{R}$, če $r \in \mathcal{R}$. V praksi množico tekov, ki tvorijo sistem, izbere načrtovalec ali tisti, ki sistem analizira, saj predpostavljamo, da ima model mogočih izvajanj protokola. Zaradi preprostosti privzemimo, da je v sistemu stalno število n procesov. Za razliko od mnogih drugih pristopov je tu čas linearen in ne delen red.

Teki so neskončen opis dogodkov po času. Včasih je koristno govoriti o končnih tekih. Ti so funkcije iz nekega začetnega dela množice naravnih števil v globalna stanja. Imejmo tek $r \in \mathcal{R}$. Pravimo, da je $r|_m$, omejitev teka r do časa m, končni tek z območjem $\{0, \ldots, m\}$, ki se ujema z r na njunem skupnem območju. ρ je predpona r, če obstaja tak $m \geq 0$, da je $\rho = r|_m$. Če ima končen tek ρ območje $\{0, \ldots, m\}$, je njegova dolžina $|\rho| = m$. ($|\rho|$ je tudi število prehodov v ρ.) Če imamo sistem $\mathcal{R}$, Pref($\mathcal{R}$) vsebuje vse teke iz $\mathcal{R}$ skupaj z vsemi končnimi predponami tekov v $\mathcal{R}$. Če $\rho, \rho' \in$ Pref($\mathcal{R}$), pravimo, da je ρ predpona ρ', če obstaja tak $r \in \mathcal{R}$ in $m \leq m'$, da velja $\rho = r|_m$, $\rho' = r|_{m'}$.

Za sisteme so značilni določeni tipi akcij. Tipične akcije so branje, pisanje v skupno spremenljivko, pošiljanje sporočila, sprejem sporočila, lokalno procesiranje. Modeliranje sistema - izbira prostora stanj procesov in okolja ter množice tekov sistema - je stvar posameznika. V /3/ najdemo modele nekaj značilnih sistemov.

3.2 Znanje

Omenili smo že, da je v naš model lahko vgraditi znanje. Želimo zajeti idejo, da proces ve neko dejstvo v določeni točki v sistemu, če je to dejstvo pravilno v vseh drugih točkah v sistemu, kjer je proces v istem lokalnem stanju kot v tej točki.

Oglejmo si vpeljavo znanja natančneje. Imejmo množico enostavnih izjav Φ, ki naj opisujejo osnovna dejstva o sistemu, na primer "vrednost spremenljivke x je 0", "začetna vhodna vrednost procesa 1 je bila 17", "proces 3 pošlje sporočilo m okoli časa 5 tega teka" ali "sistem je v zagati". V praksi so osnovna dejstva odvisna samo od trenutnega globalnega stanja, čeprav tega ne bomo privzeli (tako da so za osnovna dejstva dovoljena tudi dejstva kot "protokol se sčasoma konča", čeprav je njegova pravilnost morda odvisna od nekega prihodnjega globalnega stanja). V mnogih primerih bo osnovno dejstvo p lokalno za določen proces i, tako da bo pravilnost trditve p odvisna samo od lokalnega stanja procesa i.

Razširimo naš jezik z osnovnih dejstev Φ na formule, ki izražajo konjunkcijo, negacijo in izjave o znanju. Torej, če sta φ in ψ formuli, so to tudi $\varphi \wedge \psi$, $\neg\varphi$ in $K_i\varphi$ ("proces i ve φ"). Da bi lahko ugotovljali, kdaj so te pravilne v porazdeljenem sistemu, najprej priredimo logične vrednosti osnovnim dejstvom v Φ.

Definicija: Interpretiran sistem $\mathcal{I}$ je par $(\mathcal{R},\pi)$, kjer je $\mathcal{R}$ sistem in π prireja logične vrednosti osnovnim dejstvom v vsaki točki v $\mathcal{R}$, tako da za vsak $p \in \Phi$ in točko (r,m) v $\mathcal{R}$ velja $\pi(r,m)(p) \in \{\text{true},\text{false}\}$. Pravimo, da je točka (r,m) v interpretiranem sistemu $\mathcal{I} = (\mathcal{R},\pi)$, če $r \in \mathcal{R}$. □

Za dan interpretiran sistem $\mathcal{I} = (\mathcal{R},\pi)$ in točko (r,m) v $\mathcal{I}$ definirajmo izpolnljivostno relacijo $\models$ med trojico $(\mathcal{I},r,m)$ in formulo φ. Za osnovno dejstvo $p \in \Phi$ velja:

$(\mathcal{I},r,m) \models p$, če in samo če $\pi(r,m)(p) = \text{true}$.

Relacijo $\models$ razširimo na $\wedge$ in $\neg$ na običajni način:

$(\mathcal{I},r,m) \models \neg\varphi$, če in samo če $(\mathcal{I},r,m) \not\models \varphi$,
$(\mathcal{I},r,m) \models \varphi \wedge \psi$, če in samo če
$(\mathcal{I},r,m) \models \varphi$ in $(\mathcal{I},r,m) \models \psi$.

Da bi lahko s formulami zajeli znanje, definirajmo, da sta točki (r,m) in (r',m') nerazločljivi za i, $(r,m) \sim_i (r',m')$, če $r_i(m) = r'_i(m')$. (r,m) in (r',m') sta torej nerazločljivi za i, če je i v istem lokalnem stanju v obeh točkah. Definirajmo:

$(\mathcal{I},r,m) \models K_i\varphi$, če in samo če
za vse r' in m', $(r,m) \sim_i (r',m')$,
velja $(\mathcal{I},r,m) \models \varphi$.

Kot smo že povedali, ta interpretacija znanja zadošča aksiomom modalne logike S5. Njihov pomen spoznamo, če kot osebke v razlagi jemljemo procese. Veljavni so tudi aksiomi o skupnem znanju.

Ponovno si oglejmo le (A2) in (R2), ki skupaj pravita, da so procesi "popolni misleci" in ne upoštevata, da "normalni" ljudje ali stroji nimajo takšne računske moči, da bi izvedli vsa mogoča logična sklepanja. S to logiko je mogoče predstaviti pomanjkanje znanja zaradi pomanjkanja potrebnih informacij. Ne moremo pa predstaviti pomanjkanja znanja, katerega vzrok je ta, da izvedljivo sklepanje ni bilo izvedeno /8/. Nas to ne bo motilo, ker bomo ves čas govorili o zunanjem pojmu znanja. To pomeni, da znanje procesom pripiše načrtovalec za pomoč pri analizi, in sicer toliko znanja, da procesi vedo ravno to, kar morajo vedeti. Ne jemljemo, da morajo procesi znanje "izračunati". Temu pravimo informacijsko osnovan pojem znanja.

Jezik lahko razširimo s standardnimi časovnimi operatorji, na primer z $\Box$ ("vedno"), $\Diamond$ ("sčasoma"), $\mathcal{U}$ ("dokler ne"). Z njimi se izražamo o času, na primer "proces 3 bo sčasoma poznal vrednost spremenljivke x". V splošnem časovne operatorje uporabljamo za sklepanje o dogodkih v posameznem teku (ni zagate, sčasoma se transakcija zaključi,...), operatorje znanja pa za sklepanje o dogodkih, ki se morda dogajajo v drugih tekih, za katere določen proces ve, da bi se lahko dejansko odvijali.

Oglejmo si dva značilna pojma v zvezi s sklepanjem o znanju in času.

(Popolnoma) sinhron sistem $\mathcal{R}$ je tak, kjer imamo globalno uro in je urin čas del stanja vsakega procesa. Tako vsi procesi poznajo čas ("vedo, koliko je ura"). Formalno je $\mathcal{R}$ sinhron sistem, če za vsak proces i in točki (r,m), (r',m') v $\mathcal{R}$ velja: če $(r,m) \sim_i (r',m')$, potem $m = m'$. Pravimo, da je interpretiran sistem $\mathcal{I} = (\mathcal{R},\pi)$ sinhron, če je $\mathcal{R}$ sinhron. Opazimo, da je sistem sinhron natanko tedaj, če lahko proces vedno razloči točke v sedanjosti od točk v prihodnosti.

Pravimo, da procesi ne pozabljajo, če je, intuitivno, njihova množica možnosti vedno enaka ali pa se s časom manjša (temu rečejo tudi neomejen spomin ali kumulativno znanje). Povejmo to natančneje. Definirajmo zaporedje stanj procesa i v točki (r,m) kot zaporedje lokalnih stanj, skozi katera je šel proces v teku r do časa m, brez zaporednih ponavljanj. Če je na primer proces i šel v teku r od časa 0 do 4 skozi zaporedje lokalnih stanj $\langle s, s, s', s, s\rangle$, je njegovo zaporedje stanj v točki $(r,4)$ enako $\langle s, s', s\rangle$. Pravimo, da proces i ne pozablja v sistemu $\mathcal{R}$, če v vseh točkah (r,m) in (r',m') v $\mathcal{R}$ velja: če $(r,m) \sim_i (r',m')$, potem ima proces i isto zaporedje stanj v (r,m) in (r',m'). Proces i torej ne pozablja, če si "zapomni" svoje zaporedje stanj.

3.3 Protokol

Procesi ponavadi izvajajo akcije skladno z nekim protokolom (algoritmom, strategijo...). Protokol za proces i je neke vrste opis, katere akcije bo proces izvedel kot funkcijo svojih lokalnih stanj. Natančneje, vzemimo, da je A_i množica akcij procesa i in definirajmo protokol nad prostorom stanj L_i kot funkcijo iz L_i v neprazne množice akcij iz A_i. S tem, da protokol preslikava lokalno stanje v množico akcij, zajamemo možen nedeterminizem protokola. Seveda se v vsakem koraku protokola izvede samo ena od mogočih akcij. Deterministične protokol preslikuje stanja v množice z eno akcijo. A_i tipično vsebuje nekaj osnovnih akcij, na primer branje podatka, vpisovanje vrednosti, pošiljanje sporočila, potezo v igri...

Tako kot je koristno vzeti, da okolje izvaja akcije, je koristno jemati, da tudi okolje izvaja protokol. S protokolom okolja na primer predstavimo možnost izgube sporočil ali sprejem sporočil v drugačnem vrstnem redu, kot so bila oddana; s sporočili iz okolja lahko predstavimo tudi vhod v sistem iz zunanjosti... Izberemo torej množico akcij okolja A_e in definiramo protokol okolja kot funkcijo iz L_e v neprazne množice akcij iz A_e.

Avtorja tega pojma protokola ne omejujeta funkcij, ki definirajo protokol, samo na izračunljive funkcije, čeprav bi to prav lahko storila. Opazimo, da zahtevamo, naj je protokol funkcija iz lokalnih stanj v množice akcij, ne pa iz globalnih stanj. Kaj bo proces

storil, je tako odvisno samo od njegovega lokalnega stanja.

Skupen protokol P je $(n+1)$-terica $(P_e, P_1, \ldots, P_n)$, kjer je P_e protokol okolja, P_i, $i = 1, \ldots, n$, pa so protokoli procesov. Za analizo je protokolu dobro pridružiti sistem, ki je množica tekov protokola. Množico tekov pridružimo skupnemu protokolu $P = (P_e, P_1, \ldots, P_n)$, kjer je $P_e : L_e \dashrightarrow 2^{A_e} - \emptyset$ in $P_i : L_i \dashrightarrow 2^{A_i} - \emptyset$, $i = 1, \ldots, n$, tako, da izberemo množico globalnih stanj $\mathcal{G} \subseteq L_e \times L_1 \times \ldots \times L_n$, množico začetnih stanj $\mathcal{G}_0 \subseteq \mathcal{G}$ in prehodno funkcijo τ, ki vsaki skupni akciji $(a_e, a_1, \ldots, a_n) \in A_e \times A_1 \times \ldots \times A_n$ pridruži pretvornik globalnih stanj $\tau(a_e, a_1, \ldots, a_n)$, t.j. funkcijo iz $\mathcal{G}$ v $\mathcal{G}$. Pravimo, da je tek r skladen s skupnim protokolom P, če velja:

1. $r(0) \in \mathcal{G}_0$ (tako je $r(0)$ dovoljeno začetno stanje).
2. Za vsak $m \geq 0$, če $r(m) = (s_e, s_1, \ldots, s_n)$, potem obstaja taka skupna akcija $(a_e, a_1, \ldots, a_n) \in P_e(s_e) \times P_1(s_1) \times \ldots \times P_n(s_n)$, da $r(m+1) = \tau(a_e, a_1, \ldots, a_n)(r(m))$ $(r(m+1)$ je rezultat transformiranja $r(m)$ s skupno akcijo, ki bi lahko bila izvedena v $r(m)$ v skladu s P).

Z $\mathcal{R}(P)$ označujemo množico vseh tekov, skladnih s skupnim protokolom P. Ko govorimo o "tekih protokola P", ponavadi mislimo na sistem $\mathcal{R}(P)$. Če pa so dane globalne omejitve za sistem (na primer zahteve po poštenosti), upoštevamo podmnožico množice $\mathcal{R}(P)$, ki zadosti omejitvi.

3.4 Protokol na osnovi znanja

Povedali smo že, da je naš pojem protokola dovolj splošen, da zajame vse algoritme, ki se jih da napisati v kateremkoli danes znanem programskem jeziku. Vendar ta pojem protokola ne omogoča visokonivojskega, sistemsko neodvisnega opisa odnosov med znanjem in akcijami. Ponazorimo to s primerom, v katerem nekdo sreča človeka, ki je znan bodisi kot Resnicoljub (vedno govori resnico) bodisi kot Lažnivec (vedno laže). Z vprašanji je treba ugotoviti, kdo je. Pravila igre so taka, da Resnicoljub odgovori na vprašanje "Je φ res?" z "da", če je trditev φ pravilna, in z "ne", če je φ nepravilna. Lažnivec pa odgovori z "da", če φ ni pravilna, in z "ne", če je φ pravilna.

Če Resnicoljuba vprašamo o φ, za katero ne ve, ali je pravilna ali nepravilna, ne more izvajati tega protokola (razen če odgovora ne ugane). Pravila spremenimo takole: če Resnicoljub ve, da je φ pravilna, naj odgovori z "da"; če ve, da je φ nepravilna, naj odgovori z "ne"; sicer pa naj reče "ne vem". Podobno naj odgovarja Lažnivec. (Vprašanje je le, kaj naj odvrne Lažnivec, če ne ve, ali je φ pravilna ali nepravilna - privzemimo, da mu vprašanj, na katera ne ve odgovora, ne postavljamo.)

Na ta način smo za Resnicoljuba in Lažnivca dobili protokola z izrecnim testiranjem znanja. Na primer, če jemljemo Resnicoljuba kot proces 1, lahko gledamo njegov protokol P_1 kot funkcijo, ki ima v stanju, ko postavimo vprašanje o φ, obliko:

if $K_1\varphi$ then reci "da"
else if $K_1\neg\varphi$ then reci "ne"
else reci "ne vem".

To ni standarden protokol, ker logične vrednosti $K_1\varphi$ ne moremo ugotoviti le s pregledom lokalnega stanja procesa 1. Odvisna je od logične vrednosti φ v drugih točkah (v vseh, kjer proces 1 ne more globalnega stanja razločiti od trenutnega globalnega stanja).

Standarden protokol za proces i je torej funkcija iz lokalnih stanj tega procesa v akcije, protokol na osnovi znanja za proces i pa lahko gledamo kot program s stavki oblike "if $K_i\varphi_1$ then a else if $K_i\varphi_2$ then a'...", kjer so a, a',... akcije iz A_i. Za odločanje o izidu testov potrebujemo interpretiran sistem.

Protokol na osnovi znanja je tehnično primerno jemati kot funkcijo iz para (lokalno stanje, interpretiran sistem) v akcije. Formalno, vzemimo množico globalnih stanj $\mathcal{G} \subseteq L_e \times L_1 \times \ldots \times L_n$ in množico akcij procesa i, A_i. $INT(\mathcal{G})$ bodi množica vseh takih interpretiranih sistemov $\mathcal{I} = (\mathcal{R}, \pi)$, da so za vsak $r \in \mathcal{R}$ vsa globalna stanja v r iz $\mathcal{G}$. Potem je protokol na osnovi znanja za proces i funkcija P_i iz $L_i \times INT(\mathcal{G})$ v neprazno množico akcij iz A_i. Vzemimo, da je Resnicoljub v stanju l, ko mu postavimo vprašanje: "Ali φ drži?" Potem imamo:

$P_1(l, \mathcal{I}) =$
reci "da", če $(\mathcal{I}, r, m) \models \varphi$ za vse točke (r,m), kjer $r_1(m) = l$;
reci "ne", če $(\mathcal{I}, r, m) \models \neg\varphi$ za vse točke (r,m), kjer $r_1(m) = l$;
reci "ne vem", sicer.

Opazimo, da je edina razlika med formalno definicijo protokola na osnovi znanja in standardnega protokola v tem, da ima protokol na osnovi znanja za argument še interpretiran sistem. Če fiksiramo $\mathcal{I}$, protokol na osnovi znanja preide v standarden protokol. Torej lahko gledamo protokole na osnovi znanja kot funkcije iz interpretiranih sistemov v standardne protokole. Standarden protokol lahko jemljemo kot posebno obliko protokola na osnovi znanja, kjer funkcija ni odvisna od interpretiranega sistema.

Podobno kot za standardne protokole definiramo skupen protokol na osnovi znanja kot $(n+1)$-terico protokolov na osnovi znanja $(P_e, P_1, \ldots, P_n)$, ki so vsi definirani glede na isto množico $\mathcal{G}$ (t.j. obstaja taka množica $\mathcal{G} \subseteq L_e \times L_1 \times \ldots \times L_n$, da je P_e funkcija iz $L_e \times INT(\mathcal{G})$ v neprazno množico akcij iz A_e in P_i funkcija iz $L_i \times INT(\mathcal{G})$ v neprazno množico akcij iz A_i, $i = 1, \ldots, n$). Radi bi definirali tudi tek, skladen s protokolom na osnovi znanja, vendar moramo najprej specificirati interpretiran sistem, ker je argument protokola te vrste. Imejmo skupen protokol na osnovi znanja P kot prej, množico začetnih stanj $\mathcal{G}_0 \subseteq \mathcal{G}$, prehodno funkcijo τ in interpretiran sistem $\mathcal{I}$. Tek r je skladen s P glede na $\mathcal{I}$ v istem primeru kot tek s standardnim protokolom, le da je zdaj skupna akcija $(a_e, a_1, \ldots, a_n)$ v prejšnji definiciji element množice $P_e(s_e, \mathcal{I}) \times P_1(s_1, \mathcal{I}) \times \ldots \times P_n(s_n, \mathcal{I})$ in ne $P_e(s_e) \times P_1(s_1) \times \ldots \times P_n(s_n)$. Interpretiran sistem $\mathcal{I} = (\mathcal{R}, \pi)$ je skladen s protokolom na osnovi znanja P, če je vsak $r \in \mathcal{R}$ skladen s P glede na $\mathcal{I}$.

Opazimo, da je definicija interpretiranega sistema, skladnega s protokolom znanja, krožna. Zato tudi ni presenetljivo, da se v nasprotju s standardnimi protokoli lahko zgodi, da ni nobenega interpretiranega sistema, skladnega z danim protokolom na osnovi znanja, bolj pogosto pa je več interpretiranih sistemov, skladnih z njim.

Ta razlika med protokoli na osnovi znanja in standardnimi protokoli je pomembna pri dokazovanju pravilnosti protokolov, t.j. ali protokol zadošča specifikaciji. Če hočemo dokazati, da standarden protokol P zadošča specifikaciji, običajno dokažemo, da specifikacija drži za vse teke v $\mathcal{R}(P)$ (ali pa

za vse teke, ki zadoščajo določenim omejitvam, na primer zahtevi po poštenosti). Pravilnost protokola na osnovi znanja P pa dokažemo tako, da dokažemo, da specifikacija drži za vse interpretirane sisteme, skladne s P.

Želimo si, da bi protokol na osnovi znanja določal eno množico tekov, a poglejmo, zakaj to včasih ni mogoče. Imejmo protokol na osnovi znanja P, množico globalnih stanj $\mathcal{G}$, podmnožico $\mathcal{G}_0 \subseteq \mathcal{G}$ in prehodno funkcijo τ. Očitna pot za konstrukcijo množice tekov je ta, da konstruiramo vse predpone skladnih tekov dolžine m z indukcijo po m. Vzemimo, da smo uspeli skonstruirati predpone dolžine m. Da bi ugotovili, katera akcija naj se izvede v točki (r,m), moramo poznati rezultat testiranja izraza oblike $K_i\varphi$. Ta rezultat pa je odvisen od logične vrednosti φ v drugih točkah, kjer je proces i v istem stanju kot v trenutni točki. Torej sta lahko dva razloga za to, da ne moremo ugotoviti logične vrednosti $K_i\varphi$:
1. morda so v prihodnosti točke z istim lokalnim stanjem procesa i kot v tej točki, (r,m), pa jih še nismo skonstruirali,
2. tudi če i lahko loči točke v sedanjosti od točk v prihodnosti, je morda logična vrednost izraza φ samega odvisna od prihodnosti (na primer, φ je lahko izraz oblike $\Diamond p$, ki je pravilen samo, če bo p sčasoma pravilna).

Spomnimo pa se, da imajo v sinhronih sistemih procesi dostop do globalne ure, tako da lahko vedno razločijo točke v sedanjosti od točk v prihodnosti. Za sinhrone sisteme namreč lahko privzamemo, da je lokalno stanje procesa i oblike $(m,\ldots)$ - prva komponenta je čas - in podobno za lokalno stanje okolja. Tako je globalno stanje sinhronega sistema v točki (r,m) oblike $((m,\ldots), (m,\ldots), \ldots, (m,\ldots))$. Privzamemo lahko tudi, da so vsa globalna stanja v množici globalnih stanj sistema te oblike in da ima njegova prehodna funkcija τ lastnost, da je rezultat akcije vedno povečanje časa za eno enoto. Tako za katerokoli skupno akcijo $(a_1, \ldots, a_n)$ v sistemu velja $\tau(a_1, \ldots, a_n)((m,\ldots), (m,\ldots), \ldots, (m,\ldots)) = ((m+1,\ldots), (m+1,\ldots), \ldots, (m+1,\ldots))$. Torej se prvi težavi izognemo, če se omejimo na sinhrone sisteme. Če se poleg tega omejimo na protokole na osnovi znanja, katerih akcije so odvisne samo od preteklosti, pa odpravimo še drugi razlog. Kljub temu še vedno ne dobimo ene množice tekov. Mogoče pa je dokazati, da dobimo kanonično množico tekov, ki jo lahko jemljemo kot želeno množico tekov, določeno s protokolom na osnovi znanja /3/.

V naslednjem razdelku podajamo primer analize sistema s pomočjo formalnega modela in pojmov v zvezi z znanjem, ki smo jih spoznali. Sistem zadošča obema prej navedenima omejitvama.

3.5 Uganka o nezvestih možeh

Najprej bistvo uganke:
Med ženskami Mamajorce je bilo splošno znano, (da so popolni misleci), da so njihove kraljice resnicoljubne in da so ženske pokorne kraljicam. Bilo je tudi splošno znano, da vse ženske slišijo vsak strel, ki poči v Mamajorci. Nekega jutra se je kraljica Henrieta I. prebudila trdno odločena, da bo opravila z moško nezvestobo v Mamajorci. Ženske vseh gospodinjstev je zbrala na mestnem trgu in jim prebrala tale razglas:

> "V naši skupnosti je eden ali več nezvestih mož. Čeprav nobena od vas pred tem shodom ni vedela, ali ji je mož nezvest, pa vsaka od vas ve, kateri izmed drugih mož so nezvesti. Prepovedujem vam, da bi se o nezvestobi lastnih mož pogovarjale med seboj. Če pa kljub temu odkrijete, da vam mož ni zvest, ga morate ustreliti ob polnoči na dan odkritja."

Minilo je 39 tihih noči, na štirideseti dan pa so se zaslišali streli.

Uganiti je treba, koliko nezvestih mož je bilo in zakaj so se slišali streli prav na štirideseto noč.

Lahko pokažemo, da je moralo biti 40 nezvestih mož. Z indukcijo lahko dokažemo, da bi bili streli na k-to noč in ne prej, če bi bilo natanko k nezvestih mož. Če bi bil samo en mož nezvest, bi njegova žena, vedoč, da so vsi drugi možje zvesti, takoj ko bi slišala od kraljice, da obstajajo nezvesti možje, vedela, da je nezvest prav njen mož. Če k=1, je torej edini nezvesti mož ustreljen na prvo noč. Vzemimo, da je k+1 nezvestih mož. V tem primeru njihove žene vedo za k nezvestih mož (za vse razen za svojega). Ko na k-to noč ne slišijo strelov, ugotovijo, da ne more biti natanko k nezvestih mož, saj po indukcijski hipotezi velja, da bi bili vsi ustreljeni na k-to noč. Tako lahko vsaka žena sklepa, da ji je mož nezvest in ga ustreli na k+1-vo noč. Prej tega ne bi mogla ugotoviti, ker ne bi mogla razločiti dejanske situacije od tiste, kjer bi bil njen mož zvest, k drugih pa nezvestih.

Na prvi pogled se zdi, da je kraljičin uvodni stavek, češ da so v skupnosti nezvesti možje, odveč, saj so ženske že prej vedele, da obstajajo. Vendar se da pokazati, da nobena ne bi mogla sklepati o nezvestobi svojega moža brez tega stavka. Da bi to bolje razumeli, analizirajmo, kako se znanje žensk s časom spreminja.

Jasno je, da vse ženske sledijo temu preprostemu protokolu na osnovi znanja: "Za vsak dan $k = 1,2,3,\ldots$, če veš, da ti je mož nezvest, ga ustreli ob polnoči, sicer ne stori ničesar." Ta protokol teče v sinhronem sistemu, katerega akcije so odvisne samo od preteklosti, tako da lahko konstruiramo kanoničen interpretiran sistem, skladen s tem protokolom. Najprej storimo to neformalno.

Predpostavimo, da je v mestu n zakonskih parov, 1, ..., n. Položaj lahko opišemo z n-terico ničel in enic oblike $(x_1, \ldots, x_n)$; x_i=1, če je mož nezvest, sicer pa x_i=0. Če n=5, potem (1, 0, 1, 0, 0) pove, da sta nezvesta natanko dva moža, tisti od žene številka 1 in številka 3. Imejmo opis dejanske situacije $(x_1, \ldots, x_n)$. Ker žena 1 ve za vse nezveste može razen za svojega, v začetku upošteva dve možni situaciji, (0, $x_2, \ldots, x_n$) in (1, $x_2, \ldots, x_n$). Opazimo, da ženska i ne more razločiti med dvema n-tericama natanko tedaj, če se razlikujeta samo v i-ti komponenti. Tako imamo na dan 0 (preden je kraljica spregovorila) 2^n možnih začetnih situacij. Na dan 1, potem ko kraljica spregovori, da so v mestu nezvesti možje, odpade (0, 0,..., 0). Če n=5 in če bi bila začetna situacija (1, 0, 1, 0, 0), bi, čeprav vsaka ženska že pred govorom ve, da obstajajo nezvesti možje, ženska 1 menila, da je možna situacija (0, 0, 1, 0, 0). V tem primeru bi ženska 3 vzela za možno (0, 0, 0, 0, 0) in ženska 1 bi pred govorom menila, da je možno, da ženska 3 meni, da je možno (0, 0, 0, 0, 0). Tako bi pred govorom ženska 1 menila, da je možno, da ženska 3 meni, da nezvestih mož ni. Po govoru pa postane jasno, da nezvesti možje so. Kraljičina uvodna izjava spremeni skupinsko znanje. Čeprav vsaka ženska že pred govorom ve, da so nezvesti možje, pa takrat to ni splošno znano. (Namesto da bi rekli, da "je

neko dejstvo skupno znanje", raje uporabljamo v slovenskem jeziku znani izraz, da "je dejstvo splošno znano".)

Na dan 2, ko prejšnjo noč ni strelov, lahko ženske izločijo še več možnosti, in sicer vse n-terice z natanko eno enico. Če bi bila dejanska situacija podana z (1, 0,..., 0), bi v začetku ženska 1 predpostavila dve možnosti: (1, 0,..., 0) in (0, 0,..., 0). Ker je splošno znano, da (0, 0,..., 0) ni možna, bi vedela, da velja (1, 0,..., 0) in da je njen mož nezvest). Ker pa ni bilo strelov, tega ni mogla vedeti. Torej ne velja (1, 0,..., 0). S podobnim sklepanjem tudi vse druge ženske izločijo n-terice z eno enico. Ker je splošno znano, da so vse žene popolni misleci, je na dan 2 (pred polnočjo) splošno znano, da sta najmanj dva nezvesta moža. Vsak dan lahko izločimo nekaj več. Po polnoči dne k lahko izločimo n-terice z natanko k enicami. Tako je na dan k+1 (pred polnočjo) splošno znano, da je najmanj k+1 nezvestih mož. (Opazimo, da se znanje spreminja celo brez komunikacije!) Če je v resnici k+1 enic, bodo na dan k+1 ženske z nezvestimi možmi poznale dejansko situacijo in svojega moža ponoči ustrelile.

Oglejmo si uganko še formalno, najprej opis lokalnih stanj. Stanje okolja naj bo (m, x), kjer m označuje dan, x pa popoln opis, kateri možje so nezvesti. Stanja žensk naj bodo (m, y, h); m je oznaka za dan, y je opis, koliko ženska ve o nezvestih možeh, in h je zaporedje dolžine m, ki pove, kaj je ženska slišala prejšnje dni. y je n-terica x^i, ki je kot x, le da ima na i-tem mestu *, ker i-ta ženska za svojega moža ne ve, ali ji je zvest. Začetna globalna stanja so oblike $((0, x), (0, x^1, <>), \ldots, (0, x^n, <>))$. Privzemimo, da Henrieta pošlje sporočilo na dan 0. Za lažje razumevanje vzemimo, da pošlje sporočilo na ta dan ali pa ga sploh ne pošlje. V naslednjih dneh k h dodamo enico, če so bili streli, ali ničlo, če jih ni bilo. Omejitve za globalna stanja so: vsi časi morajo biti enaki (enake prve komponente stanja okolja in vseh žensk); vsi x^i morajo biti enaki kot dejanska situacija, le na i-tem mestu je *; vse ženske morajo slišati isto stvar (enake komponente h - če bi bilo na primer splošno znano, da katera slabo sliši, bi bile lahko različne).

Edina ničelna akcija žensk je streljanje. Edina ničelna akcija okolja pa je oddaja kraljičinega sporočila na dan 0. Ker privzamemo, da kraljica govori resnico, je to sporočilo poslano samo, če je stanje okolja $(0, x)$ tako, da je v x najmanj ena enica (vsaj en nezvesti mož).

Ženska i sledi protokolu na osnovi znanja "za dan k = 1, 2, 3, ..., if K_i(mož i nezvest) then streljaj". Okolje izvaja protokol, ki (nedeterministično) bodisi pošlje kraljičino sporočilo bodisi na dan 0 ne naredi ničesar (in naslednje dni tudi ne). S tem je ponazorjeno dejstvo, da ženske ne vedo vnaprej, ali jim bo kraljica poslala sporočilo.

Sedaj lahko konstruiramo teke ustrezno protokolom na osnovi znanja z indukcijo, podobno kot prej neformalno. Z indukcijo vemo, da imamo na vsak dan $k \geq 1$ natanko 2^n-1 predpon dolžine k - to so predpone tekov, kjer je kraljica poslala sporočilo in ena predpona ustreza vsakemu od začetnih stanj, v katerem je najmanj en mož nezvest - in 2^n predpon dolžine k, ki pripadajo tekom, kjer kraljica ni poslala sporočila. V podmnožici predpon, kjer je kraljica poslala sporočilo in je najmanj k nezvestih mož, ženska i ve, da je njen mož nezvest (in ga ustreli) ob času k na tistih tekih, kjer je natanko k nezvestih mož in je eden izmed njih njen. To opazovanje nam omogoča razširitev predpon dolžine k na dolžino k+1. V tem interpretiranem sistemu je protokol na osnovi znanja ekvivalenten preprostemu standardnemu protokolu: "Če si slišala kraljičino začetno izjavo, če ima tvoje začetno stanje k enic (t.j. če si na začetku vedela za k nezvestih mož) in če na k-to noč ni strelov, ustreli moža na (k+1)-vo noč; sicer ne stori ničesar."

4. LOGIKA ZNANJA V MODELU Z DOGODKI

Spoznali smo, kako lahko formalno definiramo znanje s pomočjo stanj sistema in njegovih komponent. Zdaj pa bomo na kratko pokazali, kako je mogoče v porazdeljen sistem vpeljati znanje samo na osnovi njegovih dogodkov.

4.1 Model, protokol, znanje

Avtorja modela, ki si ga bomo ogledali, Parikh in Ramanujam /8/, pravita, naj imamo n procesov, $1 \leq i \leq n$, ki lahko med seboj komunicirajo prek binarnih kanalov, s splošno oddajo ali z nastavljanjem spremenljivk, do katerih ima dostop več kot en proces. Z I označimo množico vseh n procesov. Privzemimo, da imamo uro na metanivoju, t.j. da so vsi dogodki delno urejeni v globalnem (diskretnem) času (dva dogodka se lahko zgodita hkrati). Procesi pa nimajo dostopa do te ali katerekoli ure, razen če to ni posebej rečeno.

Lokalna zgodovina naj bo zaporedje vseh dogodkov nekega procesa i, notranjih dogodkov ter odposlanih in sprejetih sporočil, v takšnem vrstnem redu, kot so se pojavili. V primeru skupne ure vzemimo, da dobijo dogodki časovne znamke; to pomeni, da bodo vsi elementi lokalne zgodovine pari oblike (t,e), kjer je e dogodek in t trenutek, v katerem se je dogodek zgodil. S spremenljivkami h, h',... bomo označevali lokalne zgodovine. Imejmo tudi sestavljene dogodke in privzemimo, da se v vsakem procesu ob kateremkoli globalnem času zgodi največ en dogodek.

Globalna zgodovina naj bo zaporedje parov (t,X), kjer je t globalni čas in X množica lokalnih dogodkov. Rekli smo že, da je v množici X lahko največ en dogodek določenega procesa i. S spremenljivkami H, H',... bomo označevali globalne zgodovine.

Definirajmo preslikavo Φ_i iz množice globalnih dogodkov v množico lokalnih dogodkov U {null} (lokalnemu dogodku procesa i recimo i-dogodek) na naslednji način:

$\Phi_i(t,X)$ = null, če ni i-dogodka v X,
sicer
$\Phi_i(t,X)$ = edini i-dogodek v X,
če ni skupne ure,
= (t,e),
če imamo skupno uro in
je e edini i-dogodek v X.

Preslikavo Φ_i lahko razširimo v preslikavo (tudi imenovano Φ_i), ki ohranja konkatenacijo in katere območje je množica globalnih zgodovin. Če je U podmnožica I, potem je $\Phi_U(H)$ zaporedje $<\Phi_i(H): i \in U>$.

V prejšnjem modelu smo protokolu za lažjo analizo le pridružili množico tekov, skladnih s protokolom. Avtorja tega modela pa takšno množico istovetita s pojmom protokola. Vzemimo torej, da je protokol množica možnih globalnih zgodovin, zaprta za relacijo "je predpona". Običajno pravimo protokol končnemu mehanizmu

za zagotovitev, da ni nobena zgodovina zunaj protokola (množice). Z drugimi besedami, običajni pojem protokola ustreza gramatiki, naš pa jeziku, generiranemu z gramatiko. Če nas zanima poštenost, vzamemo za protokol samo "poštene" zgodovine. Ne obstaja pa končen mehanizem, ki bi generiral samo "poštene" zgodovine.

Definicije logike znanja za opis tega modela ne bomo natanko podali. Zanima nas predvsem definicija znanja. V jeziku imejmo formule, ki ustrezajo lastnostim posameznih globalnih zgodovin. Takšne lastnosti so na primer "vrednost lokalne spremenljivke x je zdaj 0", "vrednost spremenljivke y ni bila nikoli 0" ali lastnosti, odvisne od protokola, na primer "ta zgodovina nima razširitev v tem protokolu". Imejmo klasične Boolove povezovalne operatorje in operator K_U, kjer je U množica procesov. Označimo tako sestavljeno formulo z A in definirajmo znanje za globalno zgodovino H iz protokola P:

$H \vdash K_U(A)$, če in samo če
za vsak H', $\Phi_U(H) = \Phi_U(H')$,
velja $H' \vdash A$.

Opazimo, da je znanje tako kot v modelu možnih svetov definirano na podlagi tega, da procesi ne razločijo med seboj nekaterih globalnih dogajanj (enačaj). Zanimivo je, da je znanje tokrat definirano za množico procesov (ki ima seveda lahko tudi samo en element). Konstrukt K_U ne predstavlja skupnega znanja, ampak mu je na neki način dualen. Skupno znanje definiramo na enak način kot prej. Da se na primer pokazati, da $K_{\{1,2\}}$ ne moremo izraziti tako kot skupno znanje, namreč s pomočjo K_1 (pravzaprav $K_{\{1\}}$) in K_2. Intuitivno $K_{\{1,2\}}(A)$ pomeni, da imata procesa 1 in 2 skupaj dovolj informacij, da lahko sklepata, da je A pravilna. Tudi znanje, izraženo s K_U, je zaprto za logično implikacijo. Veljata tudi aksioma samoopazovanja.

4.2 Vloga znanja pri medsebojnem izključevanju

Za protokol za medsebojno izključevanje procesov pri dostopu do kritične sekcije je v tem modelu mogoče dokazati zanimiv izrek /8/. Upoštevati bi ga moral načrtovalec, če želi dobiti varen in živ protokol. Protokol za medsebojno izključevanje je varen, če ne moreta biti v kritični sekciji dva procesa hkrati. Živ pa je, če velja: kadarkoli proces zahteva izvajanje kritične sekcije, sčasoma tudi vstopi vanjo /9/. Za protokol naredimo privzetek, ki je skoraj vedno uresničen v praksi. Privzemimo, da je akcija procesa i odvisna samo od njegove zgodovine h in ne neposredno od trenutnega stanja drugih procesov. (Seveda pa je posredno lahko odvisna od njih zaradi poprejšnje komunikacije med i in drugimi procesi.) Privzemimo tudi, da so relativne hitrosti procesov nedoločene, a končne.

Izrek: Bodi P protokol za dostop več procesov do kritične sekcije. Varnostne in živostne lastnosti protokola so z znanjem povezane na naslednji način:
1) Dva procesa ne moreta vstopiti v kritično sekcijo hkrati natanko tedaj, če noben proces ne poskuša vstopiti v kritično sekcijo, razen če "ve", da je prazna in da je trenutno ne skuša doseči noben drug proces.
2) Če je protokol varen in če se mora vsak proces pravilno končati, potem, kadarkoli je kritična sekcija prazna in bi kakšen proces moral vstopiti vanjo, sčasoma postane temu procesu znano, da je kritična sekcija prazna.□

Izrek nam kaže, da mora načrtovalec protokola zagotoviti v njem zadostno izmenjavo informacij, tako da procesi lahko dobijo potrebno znanje.

Primer:
Imejmo procesa 1 in 2, ki si delita kritično sekcijo. x naj bo Boolova spremenljivka, ki jo proces 1 lahko nastavlja, proces 2 pa bere. Obrnjeno naj velja za spremenljivko y. Ko želi proces 1 vstopiti v kritično sekcijo, postopa tako:
1: Nastavi x na nakljućno vrednost. If x=1 then goto 1.
2: Testiraj y. If y=0 then goto 1.
3: Izvajaj kritično sekcijo.
4: Postavi x na 1.

Procedura za proces 2 je tej dualna.

Poglejmo varnost tega protokola s stališča znanja v našem modelu.

Vzemimo, da je proces 1 na tem, da vstopi v kritično sekcijo. Potem se njegova lokalna zgodovina h_1 končuje z dogodkoma x:=0; y=1? (x dobi vrednost 0 in proces izve, da je y enak 1). Če bi bil proces 2 v kritični sekciji ali pred vstopom vanjo, bi se njegova lokalna zgodovina končevala z y:=0; x=1?; CS2 (CS2 pomeni, da proces 2 izvaja kritično sekcijo) ali z y:=0; x=1?. Vsekakor nobena globalna zgodovina H nima lastnosti, da bi bilo $\Phi_1(H) = h_1$ in $\Phi_2(H) = h_2$. Tako proces 1 ve, da je kritična sekcija prazna in da proces 2 ne poskuša vanjo pravkar vstopiti. Protokol je varen.

5. SKLEP

Obstaja mnogo formalnih jezikov brez pojma znanja za izražanje o stanjih, dogodkih in njihovih zaporedjih v različnih modelih porazdeljenih sistemov. V članku smo pokazali, kako s formalno uvedbo znanja jezik in način sklepanja približamo naravnemu človekovemu razmišljanju. Definirali smo protokole na osnovi znanja in spoznali vlogo znanja pri medsebojnem izključevanju. Zanimiva je ideja /3/, da bi imeli programski jezik, v katerem bi protokole pisali neposredno kot protokole na osnovi znanja. Takšen visok programski jezik bi imel "prevajalnik", ki bi teste znanja prevajal v teste, v katerih znanje ne bi nastopalo. Programerju bi bilo "računanje" znanja prikrito. S pomočjo poznavanja zakonitosti spreminjanja znanja s komunikacijo v sistemu se da na primer tudi dokazati, najmanj koliko sporočil je treba izmenjati za rešitev nekega problema /1/. Raziskovanje znanja v porazdeljenih sistemih je razmeroma novo področje, ki bo gotovo pripomoglo k boljšemu razumevanju dogajanja v njih in tako igralo pomembno vlogo pri njihovem načrtovanju.

6. LITERATURA

/1/ Chandy, K.M., Misra, J. (1986) How processes learn. Distributed Computing 1:40-52.

/2/ Fagin, R., Halpern, J.Y. (1988) Belief, awareness, and limited reasoning. Artificial Intelligence 34:39-76.

/3/ Halpern, J.Y., and Fagin, R. (1989) Modelling knowledge and action in distributed systems. Distributed Computing 3:159-177.

/4/ Kapus, T., Horvat, B. (1989) Formal

verification of distributed systems. Informatica 13, No.4, 44-47.

/5/ Lamport, L. (1983) What good is temporal logic? Proc. IFIP, 657-668.

/6/ Manna, Z., Pnueli, A. (1981) Verification of concurrent programs, Part I: The temporal framework. Technical Report STAN-CS-81-836, Stanford University.

/7/ Milner, R. (1980) A calculus of communicating systems (LCNS, vol.92), Springer-Verlag.

/8/ Parikh, R., Ramanujam, R. (1985) Distributed processes and the logic of knowledge. Parikh, R. (ed.) Proc. of the Workshop on Logic of Programs. (LNCS, vol.193), 256-267.

/9/ Walker, D.J. (1989) Automated Analysis of Mutual Exclusion Algorithms using CCS. Formal Aspects of Computing 1:273-292.

Igor Kononenko
Univerza v Ljubljani,
Fakulteta za elektrotehniko in
računalništvo, Ljubljana
Tržaška 25, Ljubljana

Keywords: machine learning, artificial neural networks, artificial intelligence, naive Bayes, Hopfield's model

POVZETEK

V prispevku je pokazano, da z nevronsko mrežo lahko naravno implementiramo 'naivni' Bayesov klasifikator. Za učenje mreže zadošča preprosto Hebbovo pravilo, ki pravi, da se vrednost uteži na sinapsi poveča, če sta hkrati aktivna oba povezana nevrona. Dokazana je konvergenca izvajanja večsmerne Bayesove nevronske mreže. Podana je interpretacija izvajanja enega nevrona kot seštevanje informacijskih prispevkov ostalih nevronov. Definirana je tudi večsmerna Bayesova mreža, ki temelji na razmerju verjetnosti. Dokazana je konvergenca za tako definirano mrežo. Izvajanje enega nevrona se interpretira kot vsota uteži evidence. Podana je relacija s sistemi za induktivno gradnjo odločitvenih dreves. Pokazano je, da lahko 'naivni' Bayes prevedemo na obteženo vsoto. S tem je podana analogija s Hopfieldovim modelom nevronske mreže. Eksperimentalno je pokazano, da Bayesova nevronska mreža močno presega Hopfieldov model po klasifikacijski natančnosti, medtem ko kompleksnost učenja in izvajanja ostaneta enaki. V poskusih s štirimi medicinskimi diagnostičnimi problemi je bila Bayesova nevronska mreža nekoliko boljša od naivnega Bayesovega klasifikatorja, ker iteracije delno odpravijo šum in manjkajoče podatke, in je presegla diagnostično natančnost zdravnikov specialistov.

BAYESIAN ARTIFICIAL NEURAL NETWORKS

It is shown that the naive Bayesian classifier can be naturally implemented with an artificial neural network. The basic Hebbian learning rule is used which states that the weight of a synapse is increased if both connected neurons are active. The convergence of the execution of the multidirectional feedback Bayesian neural network is proved. The interpretation of the execution of one neuron is defined as the summation of information gains from other neurons. A multidirectional feedback Bayesian neural network based on probability ratio is also defined and the convergence of the execution is proved. The execution of one neuron of such network is interpreted as the summation of the weights of evidence. The relation with the systems for inductive learning of decision trees is given. It is shown that naive Bayes can be transformed into weighted sum which shows the analogy with Hopfield's feedback neural network model. It was shown experimentally that the Bayesian neural network significantly outperforms Hopfield's model with respect to the classification accuracy while the complexities of learning and execution remain the same. In experiments with four medical diagnostic problems the network slightly outperformed naive Bayesian classifier as the iterations make the network less sensitive to noise and missing data. The naive Bayesian classifier and the network outperformed the diagnostic accuracy of physicians specialists.

1 Uvod

V zadnjem času se vse več raziskovalcev ukvarja z umetnimi nevronskimi mrežami (Kononenko 1989d), ki kot sistemi za avtomatsko učenje odstopajo od strukturnega induktivnega avtomatskega učenja (Kononenko 1985a) po tem, da rezultat učenja ni v simbolični obliki in je zato težko interpretirati dobljeno znanje. Vendar so umetne nevronske mreže mnogo bolj fleksibilne.

Umetni nevroni so preprosti elementi, ki znajo izračunati obteženo vsoto prispevkov ostalih nevronov. Izhod nevrona je ponavadi binaren, z vrednostjo 0 ali 1. Prednosti umetnih nevronskih mrež so: analogija z biološkimi nevronskimi mrežami, paralelno izvajanje,

robustnost glede manjkajočih in napačnih podatkov in glede na poškodbe mreže, lokalen dostop do informacije, majhne potrebe po programski opremi ter sposobnost prilagajanja okolju z učenjem.

Problemi pri snovanju nevronskih mrež so: izbira topologije, izbira primernega učnega pravila, konvergenca učenja in konvergenca izvajanja. Slabost je tudi težavnost obrazložitve izvajanja in rezultatov.

Preden začnemo pisati o Bayesovih nevronskih mrežah, je treba nekoliko pojasniti sam izraz. Kombinacijska funkcija, ki jo izračunavajo nevroni v Bayesovi nevronski mreži temelji na 'naivnem' *Bayesovem* klasifikatorju (npr. Good 1950, Michie 1989, Michie & Attar 1989, Kononenko 1985b, Bratko & Kononenko 1987). Topologija, učenje in izvajanje pa je analogno Hopfieldovi *nevronski mreži* (Hopfield 1982,1984). Od tod izraz *"Bayesova nevronska mreža"*.

Čeprav ni naš namen simulirati bioloških nevronskih mrež, bomo našteli nekaj lastnosti Bayesovskih nevronske mreže (BNM), ki kažejo na analogijo z živimi možgani in opravičujejo izraz "nevronske mreže" (Kononenko 1989a):

- Nevroni v BNM so preprosti elementi, ki izračunavajo funkcijo, ki se da prevesti v obteženo vsoto. Pomnilnik BNM predstavljajo povezave (sinapse) med nevroni.
- Informacija, ki je potrebna nevronu za izračun kombinacijske funkcije je *lokalno* dosegljiva.
- Za učenje zadošča osnovno Hebbovo pravilo (Hebb 1949), za katerega je precej nevrofiziološke evidence, da predstavlja osnovno učenje v bioloških nevronskih mrežah.
- Nevroni v BNM delujejo paralelno in asinhrono.
- BNM delujejo večsmerno, t.j. ni razlike med vhodnimi in izhodnimi podatki. Podatek je lahko vhoden, izhoden ali oboje hkrati.
- BNM so robustne glede manjkajočih in napačnih podatkov.
- Izvajanje BNM lahko interpretiramo kot asociativni pomnilnik.
- Učenje BNM je inkrementalno.

Izraza ne smemo zamenjati z Bayesovimi mrežami (Bayesian networks). Bayesova mreža je usmerjeni aciklični graf, ki zadošča določenim pogojem vezanim na verjetnostno distribucijo dane množice spremenljivk (Pearl 1988). V literaturi vezani na področje umetnih nevronskih mrež smo tudi zasledili izraz Bayesove mreže (Deleu in sod. 1988), ki ga uporabljajo kot sinonim za "mreže zaupanja" (belief networks, Beuscart in sod. 1989).

2 Zaporedni Bayes in naivni Bayes

Naj bo $A_i, i = 1 \ldots n$ množica atributov, vsak z določenim številom možnih vrednosti $V_{i,j}, j = 1 \ldots NV_i$. Imejmo 2 disjunktna razreda, C in $\overline{C}$. Naj bo

$$W_i(J_i) = \frac{P(V_{i,J_i}|C, V_{1,J_1}, \ldots, V_{i-1,J_{i-1}})}{P(V_{i,J_i}|\overline{C}, V_{1,J_1}, \ldots, V_{i-1,J_{i-1}})} \quad (1)$$

kjer je $P(X|Y_1, \ldots, Y_k)$ pogojna verjetnost dogodka X pri pogojih $Y_1, \ldots, Y_k$. Osnovna formula za izračun razmerja verjetnosti razredov C in $\overline{C}$ za dani objekt je (Good 1950):

$$\frac{P(C|V_{1,J_1}, \ldots, V_{n,J_n})}{P(\overline{C}|V_{1,J_1}, \ldots, V_{n,J_n})} = \frac{P(C)}{P(\overline{C})} \prod_{i=1}^{n} W_i(J_i) \quad (2)$$

kjer so $J_i, i = 1 \ldots n$ indeksi vrednosti i-tega atributa danega objekta. Logaritem izraza (1) se interpretira kot utež evidence (the weight of evidence) za razred C vrednosti V_{i,J_i} i-tega atributa pri danih vrednostih $V_{1,J_1}, \ldots, V_{i-1,J_{i-1}}$ (Michie 1989). Naj bo

$$odds(X) = \frac{P(X)}{P(\overline{X})}$$

Ker iz Bayesovega teorema (Good 1950) sledi

$$P(X|C,Y) = \frac{P(C|X,Y)P(X,Y)}{P(C|Y)P(Y)} \quad (3)$$

lahko (1) pretvorimo v:

$$W_i(J_i) = \frac{odds(C|V_{1,J_1}, \ldots, V_{i,J_i})}{odds(C|V_{1,J_1}, \ldots, V_{i-1,J_{i-1}})} \quad (4)$$

in (2) v

$$odds(C|V_{1,J_1}, \ldots, V_{n,J_n}) =$$

$$odds(C) \prod_{i=1}^{n} \frac{odds(C|V_{1,J_1}, \ldots, V_{i,J_i})}{odds(C|V_{1,J_1}, \ldots, V_{i-1,J_{i-1}})} \quad (5)$$

Desna stran enačbe (5) se lahko pokrajša, da dobimo identiteto.

2.1 Zaporedni Bayes

Z logaritmiranjem enačbe (5) dobimo zaporedni Bayes (Good 1950, Michie 1989):

$$logodds(C|V_{1,J_1}, \ldots, V_{n,J_n}) =$$

$$logodds(C) + \sum_{i=1}^{n} logW_i(J_i) \quad (6)$$

Vrednost (6) se interpretira kot stopnja zaupanja ali možnost (plausibility), da dani objekt pripada razredu C (Michie 1989, Michie & Attar 1989). V (Kononenko 1989c) je pokazano, da enačbo (6) lahko uporabimo za učinkovito gradnjo odločitvenih dreves, ki je analogen algoritmu ID3 (Quinlan 1979), le da ID3 namesto

enačbe (5), v kateri nastopa razmerje verjetnosti, uporablja naslednjo enačbo:

$$P(C|V_{1,J_1},\ldots,V_{n,J_n}) = P(C)\prod_{i=1}^{n} Q_i(C,J_i) \qquad (7)$$

kjer je

$$Q_i(C,J_i) = \frac{P(C|V_{1,J_1},\ldots,V_{i,J_i})}{P(C|V_{1,J_1},\ldots,V_{i-1,J_{i-1}})} \qquad (8)$$

Pravilnost enačbe (7) je očitna, saj lahko desno stran pokrajšamo, tako da dobimo identiteto. Minus logaritem enačbe (7) predstavlja zaporedni Bayes, ki temelji na informacijski vsebini:

$$-log_2 P(C|V_{1,J_1},\ldots,V_{n,J_n}) =$$

$$-log_2 P(C) - \sum_{i=1}^{n} log_2 Q_i(C,J_i) \qquad (9)$$

$-log_2 P(X)$ se interpretira kot potrebna količina informacije, da zvemo da se je X zgodil (Kononenko & Bratko 1987,1989). Torej lahko enačbo (9) interpretiramo kot potrebna količina informacije, da zvemo, da dani objekt pripada razredu C pri danih atributnih vrednostih $V_{1,J_1},\ldots,V_{n,J_n}$. $-log_2 P(C)$ je količina informacije potrebna za klasifikacijo objekta v razred C preden poznamo vrednosti atributov. Minus logaritem (8) predstavlja informacijski prispevek vrednosti J_i i-tega atributa h klasifikaciji v razred C pri danih vrednostih $V_{1,J_1}, \ldots, V_{i-1,J_{i-1}}$. To je enako količini informacije, potrebni, da klasificiramo objekt v razred C pri danih vrednostih $V_{1,J_1}, \ldots, V_{i-1,J_{i-1}}$ po tem ko zvemo za vrednost i-tega atributa, minus ista količina informacije preden zvemo za vrednost i-tega atributa.

2.2 Naivni Bayes

Iz enačbe (7) dobimo naivni Bayesov klasifikator, če predpostavimo neodvisnost atributov. Enačbi (7) in (9) ostaneta nespremenjeni, le da so faktorji Q_i definirani z (8) zamenjani z Q_i' (tako spremenjeni enačbi (7) in (9) bomo označevali z (7') in (9')):

$$Q_i'(C,J_i) = \frac{P(C|V_{i,J_i})}{P(C)} = \frac{P(C,V_{i,J_i})}{P(C)P(V_{i,J_i})} \qquad (10)$$

Naivni Bayesov klasifikator se je izkazal za zelo učinkovitega v primerjavi z gradnjo odločitvenih dreves v mnogih domenah (Kononenko 1985b, Bratko & Kononenko 1987, Cestnik in sod. 1987, Cestnik 1990).

Če predpostavimo neodvisnost atributov v enačbah (5) in (6), potem moramo faktorje W_i definirane z (4) zamenjati z W_i' (tako spremenjeni enačbi (5) in (6) bomo označevali z (5') in (6')):

$$W_i'(J_i) = \frac{odds(C|V_{i,J_i})}{odds(C)} = \frac{P(C,V_{i,J_i})P(\overline{C})}{P(\overline{C},V_{i,J_i})P(C)} \qquad (11)$$

Enačbi (5') in (6') definirata naivni Bayesov klasifikator, ki temelji na razmerju verjetnosti (Michie 1989), ki je analogen naivnemu Bayesovemu klasifikatorju (7').

3 Bayesove nevronske mreže

3.1 Topologija in učno pravilo

Topologija BNM je najbolj splošna, t.j. vsak nevron je povezan z vsakim z usmerjeno povezavo, ki ji pravimo sinapsa. Dve različni topologiji bomo vpeljali: topologija, kjer je vsak nevron povezan tudi sam s seboj s *povratno povezavo* in topologija, kjer teh povratnih povezav ni. Vsaki sinapsi je pridružena utež, ki predstavlja del celotnega pomnilnika mreže. Poleg uteži na sinapsah vsebuje vsak nevron še dve dodatni uteži. Če imamo N nevronov, potem ima mreža s povratnimi povezavami N*N+2N uteži, mreža brez povratnih povezav pa N*(N-1)+2N. Topologija brez povratnih povezav ustreza topologiji Hopfieldove (1982) nevronske mreže.

Stanje nevrona ustreza njegovemu izhodu. Stanja nevronov so omejena na dve možni vrednosti 0 (neaktiven) in 1 (aktiven). Vzorci (primeri problemov), ki jih mreža obdeluje so zakodirani v vektorje, ki imajo toliko binarnih komponent, kolikor je nevronov v mreži. Stanje enega nevrona ustreza vrednosti ene komponente vzorčnega vektorja.

Mreža deluje v dveh fazah *učni* in *izvajalni*. V učni fazi mreža spreminja vrednosti uteži na sinapsah in v nevronih glede na učne primere, ki jih sprejema na vhodu. Na začetku imajo vse uteži vrednost 0. BNM se uči po osnovnem Hebbovem (1949) učnem pravilu, ki pravi, da se utež na sinapsi poveča, če sta hkrati aktivna oba nevrona.

Vzorec dobi mreža na vhodu tako, da se ustrezno postavijo stanja vseh nevronov. Ob tem se vrednost uteži na sinapsi poveča za 1, če sta stanji obeh nevronov, ki jih sinapsa povezuje, enaki 1 (oba nevrona sta aktivna). Hkrati se ob vsakem učnem primeru prva od obeh uteži v vsakem nevronu poveča za 1, druga pa se poveča za 1, če je stanje nevrona enako 1 (nevron je aktiven). Ko je učenje končano, se uteži ne spreminjajo več. Na koncu učenja je vrednost uteži na sinapsi enaka številu učnih primerov, pri katerih sta bili ustrezni komponenti enaki 1. Prva utež v vsakem nevronu je enaka številu vseh učnih primerov, druga pa številu učnih primerov, ki so imeli ustrezno komponento enako 1.

Iz učnega pravila sledi, da so uteži na sinapsah med nevroni simetrične in da lahko dve usmerjeni povezavi med dvema nevronoma nadomestimo z eno dvosmerno povezavo z eno samo utežjo. Na ta način se število uteži na sinapsah zmanjša za polovico.

V izvajalni fazi mreža na vhodu dobi primer, ki je delno pomankljiv in/ali napačen. Med izvajanjem nevroni glede na vrednosti, ki jih dobijo od ostalih nevronov, glede na uteži na sinapsah in v nevronu samem in glede na svoje trenutno stanje izračunavajo svoje novo stanje, ki ga na izhodu posredujejo preko povezav ostalim nevronom. Vsi nevroni delujejo paralelno in *as-*

inhrono. Izvajanje je končano, ko noben nevron ne more več spremeniti svojega stanja (mreža je prišla v fiksno točko). Stanja nevronov v fiksni točki predstavljajo rezultirajoči vzorec - oma izhod mreže.

Potrebno je samo še definirati *kombinacijsko funkcijo*, t.j. funkcijo s katero nevron izračunava svoje stanje oziroma izhod. V naslednjih razdelkih bomo definirali dve različni kombinacijski funkciji, eno za BNM, ki temelji na verjetnosti in eno za BNM, ki temelji na razmerju verjetnosti. Za obe kombinacijski funkciji bomo pokazali, da pri asinhronem izvajanju mreža po končnem številu iteracij (sprememb stanja enega nevrona) skonvergira v fiksno točko.

3.2 BNM, ki temelji na verjetnosti

Večsmerna Bayesova nevronska mreža, ki temelji na verjetnosti, (BNM-P) uporablja funkcijo (7') za izračun aktivacijskega nivoja nevrona. Celotna kombinacijska funkcija za izračun novega stanja S_j' in s tem izhoda nevrona je definirana z:

$$S_j' = D_j(P(S_j = 1|S_1, \ldots, S_N)) \tag{12}$$

kjer je

$$P(S_j|S_1, \ldots, S_N) = \\ = P(S_j = 1) \prod_{S_i=1,(i\neq j)} \frac{P(S_j = 1|S_i = 1)}{P(S_j = 1)} \tag{13}$$

verjetnost, da je j-ti nevron aktiven, pri danih indeksih aktivnih nevronov in

$$D_j(X) = \begin{cases} 1, & \check{c}e \quad X > P(S_j = 1) \\ S_j, & \check{c}e \quad X = P(S_j = 1) \\ 0, & \check{c}e \quad X < P(S_j = 1) \end{cases} \tag{14}$$

pragovna odločitvena funkcija. Pri tem so $S_i, i = 1..N$ trenutna stanja vseh nevronov v mreži in $P(S_j = 1)$ apriorna verjetnost aktivnega stanja j-tega nevrona. Če je izračunana verjetnost večja od apriorne verjetnosti, potem je novo stanje enako 1, če je enaka apriorni verjetnosti, potem se stanje nevrona ne spremeni, in če je izračunana verjetnost manjša od apriorne verjetnosti, potem je novo stanje nevrona enako 0.

Dejansko potrebujemo dve varianti enačbe (13). Prva, za mrežo brez povratnih povezav nevrona samega s seboj, vsebuje v produktu dodatno omejitev $i \neq j$. Druga, za mrežo s povratnimi povezavami, pa ta pogoj ne vsebuje.

Na izračun stanja nevrona vplivajo samo trenutno aktivni nevroni ($S_i = 1$). Verjetnosti v faktorjih v enačbi (13) se aproksimirajo z relativnimi frekvencami. Absolutne frekvence ustrezajo utežem na sinapsi in v nevronih samih. Tako se $P(S_j = 1)$ aproksimira z kvocientom dveh uteži v j-tem nevronu (število učnih primerov, ko je bil j-ti nevron aktiven deljeno z številom vseh učnih primerov). Verjetnost $P(S_j = 1|S_i = 1)$ pa se aproksimira z kvocientom med utežjo na sinpsi med i-tim in j-tim nevronom (število učnih primerov, pri katerih sta bila oba nevrona aktivna) in drugo utežjo v i-tem nevronu (število primerov, pri katerih je bil i-ti nevron aktiven).

V nadaljevanju bomo dokazali, da tako definirana mreža iz poljubnega začetnega stanja vedno v končnem številu sprememb stanj nevronov skonvergira v fiksno točko, če je delovanje mreže *asinhrono*, t.j. če samo en nevron naenkrat spremeni svoje stanje. Dokaz temelji na definiranju navzgor omejene funkcije stanja celotne mreže, ki s spreminjanjem stanj nevronov monotono raste. Ker je možnih stanj nevronske mreže končno mnogo, nam to zagotavlja konvergenco v končnem številu korakov.

Naslednja funkcija definira podobnost med aktivacijskimi nivoji nevronov in njihovimi trenutnimi stanji:

$$Sim(S_1, \ldots, S_N) = \\ = \prod_{S_i=1} \frac{P(S_i = 1|S_1, \ldots, S_N)}{P(S_i)} \tag{15}$$

Podobnost je večja, če je aktivacijski nivo (izračunana verjetnost, da je nevron aktiven) večji od apriorne verjetnosti, da je nevron aktiven in obratno. Najprej bomo konvergenco dokazali za BNM brez povratnih povezav, torej v enačbi (13) velja še dodatni pogoj $i \neq j$.

j-ti nevron spremeni svoje stanje v dveh primerih:

a) $S_j = 1, P(S_j = 1|S_1, \ldots, S_N) < P(S_j = 1), S_j' = 0$

Tedaj velja, da je kvocient med novo in prejšnjo podobnostjo enak:

$$\frac{Sim(S_1, \ldots, S_j', \ldots, S_N)}{Sim(S_1, \ldots, S_j, \ldots, V_N)} = \\ = \frac{P(S_j = 1)}{P(S_j = 1|S_1 \ldots S_j \ldots S_N)} \times \\ \times \prod_{S_i'=1} \frac{P(S_i = 1|S_1 \ldots S_j' \ldots S_N)}{P(S_j = 1|S_1 \ldots S_j \ldots S_N)} \\ = \frac{P(S_j = 1)}{P(S_j = 1|S_1 \ldots S_j \ldots S_N)} \times \\ \times \prod_{S_i=1, i\neq j} \frac{P(S_i = 1)P(S_j = 1)}{P(S_j = 1 \& S_j = 1)} \\ = \left(\frac{P(S_j = 1)}{P(S_j = 1|S_1 \ldots S_j \ldots S_N)}\right)^2 > 1 (zaradi\ pogoja\ a) \tag{16}$$

b) $S_j = 0, P(S_j = 1|S_1, \ldots, S_N) > P(S_j = 1), S_j' = 1$

Tedaj velja, da je kvocient med novo in prejšnjo podobnostjo enak:

$$\frac{Sim(S_1, \ldots, S_j', \ldots, S_N)}{Sim(S_1, \ldots, S_j, \ldots, V_N)} =$$

$$= \frac{P(S_j = 1|S_1 \dots S_j \dots S_N)}{P(S_j = 1)} \times$$

$$\times \prod_{S_i'=1, i \neq j} \frac{P(S_i = 1|S_1 \dots S_j' \dots S_N)}{P(S_j = 1|S_1 \dots S_j \dots S_N)}$$

$$= \frac{P(S_j = 1|S_1 \dots S_j \dots S_N)}{P(S_j = 1)} \times$$

$$\times \prod_{S_i=1, i \neq j} \frac{P(S_j = 1 \& S_j = 1)}{P(S_i = 1)P(S_j = 1)}$$

in ker je $P(S_j = 1|S_1 \dots S_j \dots S_N) = P(S_j = 1|S_1 \dots S_j' \dots S_N)$ dobimo

$$= \left(\frac{P(S_j = 1|S_1 \dots S_j \dots S_N)}{P(S_j = 1)}\right)^2 > 1 (zaradi\ pogoja\ b) \tag{17}$$

Dokaz konvergence za topologijo s povratnimi povezavami je analogen zgornjemu, le da v enačbi ni pogoja $i \neq j$. Zato bomo podali tu le rezultate. Spet sta možna dva primera a) in b). Za primer a) dobimo:

$$\frac{Sim(S_1, \dots, S_j', \dots, S_N)}{Sim(S_1, \dots, S_j, \dots, V_N)} =$$

$$= \frac{P(S_j = 1)}{P(S_j = 1|S_1 \dots S_j \dots S_N)^2} > 1 (zaradi\ pogoja\ a) \tag{18}$$

in za primer b) dobimo

$$\frac{Sim(S_1, \dots, S_j', \dots, S_N)}{Sim(S_1, \dots, S_j, \dots, V_N)} =$$

$$\frac{P(S_j = 1|S_1 \dots S_j \dots S_N)^2}{P(S_j = 1)^3} > 1 (zaradi\ pogoja\ b) \tag{19}$$

Ker je kvocient med novo in staro podobnostjo večji pri topologiji s povratnimi povezavami, lahko pričakujemo hitrejšo konvergenco pri izvajanju BNM, ki vsebuje povratne povezave.

3.3 BNM, ki temelji na razmerju verjetnosti

Večsmerna Bayesova nevronska mreža, ki temelji na razmerju verjetnosti (BNM-odds), uporablja funkcijo (5') za izračun aktivacijskega nivoja nevrona. Celotna kombinacijska funkcija za izračun novega stanja S_j' in s tem izhoda nevrona je definirana z:

$$S_j' = Dq_j(odds\ _j = 1|S_1, \dots, S_N)) \tag{20}$$

kjer je

$$odds(S_j|S_1, \dots, S_N) =$$

$$= odds(S_j = 1) \prod_{i \neq j} \frac{odds(S_j = 1|S_i = 1)}{odds(S_j = 1)} \tag{21}$$

razmerje verjetnosti, da je j-ti nevron aktiven, pri danih indeksih aktivnih nevronov in

$$Dq_j(X) = \begin{cases} 1, & če \quad X > odds(S_j = 1) \\ S_j, & če \quad X = odds(S_j = 1) \\ 0, & če \quad X < odds(S_j = 1) \end{cases} \tag{22}$$

pragovna odločitvena funkcija. Pri tem so $S_i, i = 1..N$ trenutna stanja vseh nevronov v mreži in $odds(S_j)$ apriorna verjetnost trenutnega stanja j-tega nevrona deljena z apriorno verjetnostjo nasprotnega stanja. Če je izračunano razmerje verjetnosti večje od apriornega razmerja, potem je novo stanje enako 1, če je enako apriornemu razmerju, potem se stanje nevrona ne spremeni, in če je izračunano razmerje verjetnosti manjše od apriornega, potem je novo stanje nevrona enako 0.

Analogno BNM-P bi lahko tudi pri BNM-odds definirali dve topologiji, z in brez povratnih povezav. Zaradi preprostosti bomo obravnavali samo topologijo brez povratnih povezav.

Za razliko od BNM-P tu na izračun stanja nevrona vplivajo vsi nevroni in ne samo tisti, ki imajo stanje enako 1. Zato bo interpretacija stanj 0 in 1 pri BNM-odds bolj simetrična. Verjetnosti v faktorjih v enačbi (13) se aproksimirajo z relativnimi frekvencami. Ker lahko $P(S_j \& \overline{S_i})$, $P(\overline{S_j} \& S_i)$, $P(\overline{S_j} \& \overline{S_i})$ in $P(\overline{S_j})$ izračunamo iz $P(S_j)$, $P(S_i)$ in $P(S_j \& S_i)$ velja za to mrežo isto učno pravilo kot za BNM-P.

V nadaljevanju bomo dokazali, da tudi BNM-odds iz poljubnega začetnega stanja vedno v končnem številu sprememb stanj nevronov skonvergira v fiksno točko, če je delovanje mreže *asinhrono*, t.j. če samo en nevron naenkrat spremeni svoje stanje. Za dokaz bomo uporabili isto funkcijo podobnosti, kot smo jo definirali z (15) za BNM-P.

Naj bo S_j trenutno stanje j-tega nevrona in naj velja $odds(S_j|S_1, \dots, S_N) < odds(S_j)$. Naj bo novo stanje j-tega nevrona enako $S_j' = \overline{S_j}$. Potem je kvocient med novo in prejšnjo podobnostjo enak:

$$\frac{Sim(S_1, \dots, S_j', \dots, S_N)}{Sim(S_1, \dots, S_j, \dots, S_N)} =$$

$$= \frac{odds(S_j)}{odds(S_j|S_1 \dots S_j \dots S_N)} \prod_{i \neq j} \frac{P(S_i|S_1 \dots S_j' \dots S_N)}{P(S_j|S_1 \dots S_j \dots S_N)}$$

$$= \frac{odds(S_j)}{odds(S_j|S_1 \dots S_j \dots S_N)} \prod_{i \neq j} \frac{odds(S_j)}{odds(S_j|S_i)}$$

$$= \left(\frac{odds(S_j)}{odds(S_j|S_1 \dots S_j \dots S_N)}\right)^2 > 1 \tag{23}$$

Rezultat je večji od 1 zaradi začetnega pogoja.

4 Interpretacija

Interpretacijo naivnega Bayesa nam omogočata enačbi (6') za naivnega Bayesa, ki temelji na razmerju verjet-

nosti, in (9') za naivnega Bayesa, ki temelji na verjetnosti. Interpretaciji sta analogni interpretacijam, podanim za enačbi (6) in (9), le da v vsotah nastopajo uteži evidence za enačbo (6') in prispevki informacije za enačbo (9') posameznih atributov, ki niso vezani na vrednosti ostalih atributov. Tako se interpretacija minus logaritma (8') spremeni v informacijski prispevek vrednosti J_i i-tega atributa h klasifikaciji v razred C, ne glede na vrednosti ostalih atributov. To je enako količini informacije, potrebni, da klasificiramo objekt v razred C po tem ko zvemo za vrednost i-tega atributa, minus ista količina informacije preden zvemo za vrednost i-tega atributa.

Analogno se spremeni interpretacija logaritma (4') v utež evidence s strani i-tega atributa v korist razreda C ne glede na vrednosti ostalih atributov. To je enako evidenci razreda C po tem ko zvemo za vrednost i-tega atributa, minus evidenca razreda C preden zvemo za vrednost i-tega atributa.

Izvajanje BNM lahko interpretiramo kot zaporedje aplikacij enačb (9') za BNM-P oziroma (6') za BNM-odds. Vsako aplikacijo enačbe lahko posebej interpretiramo, torej BNM lahko v celoti obrazloži svoje izvajanje. Da je zgoraj opisana interpretacija razumljiva in sprejemljiva za ljudi eksperte je pokazano v razdelku 5.5.

4.1 Relacija s Hopfieldovim modelom

Če minus logaritem (10) zamenjamo s $T_{ji}, -log_2 P(S_j)$ z I_j in levo stran enačbe (9') z A_j ("A" pomeni aktivacijski nivo), dobimo klasično obteženo vsoto, ki se uporablja za kombinacijsko funkcijo v Hopfieldovem (1982) modelu:

$$A_j = \sum_{S_i=1, i\neq j} T_{ji} + I_j = \sum_{i\neq j} S_i T_{ji} + I_j \qquad (24)$$

V Hopfieldovem modelu so T_{ji} elementi spominske matrike, dobljene kot vsota zunanjih produktov učnih vzorcev, ki so popravljeni tako, da so vrednosti komponent 0 spremenjene v 1. I_j predstavlja konstanten vhod v j-ti nevron. Model BNM brez povratnih povezav ($i \neq j$) ustreza spominski matriki z ničelno diagonalo ($T_{ii} = 0$ za vse $i = 1 \ldots N$). S_i ima lahko vrednosti 0 in 1 kot v originalnem Hopfieldovem modelu, čeprav je učno pravilo Hopfieldovega modela uporablja vrednosti -1 in 1.

Hopfieldov model uporablja posplošeno Hebbovo pravilo (Rumelhart in sod 1986), ki pravi, da se utež na sinapsi poveča za produkt stanj nevronov, ki jih povezuje sinapsa. Torej utež ustreza številu učnih primerov, ki imajo isto vrednost ustreznih dveh komponent minus število učnih primerov z različno vrednostjo teh dveh komponent. BNM uporablja uteži, ki ustrezajo številu učnih primerov, ki imajo vrednosti obeh komponent enaki 1. Ker se kombinacijska funkcija BNM lahko prevede na obteženo vsoto, je najpomembnejša razlika med modeloma v učnem pravilu, ki zagotavlja BNM bolj optimalno izkoriščanje razpoložljive informacije iz učnih primerov.

Dokaz konvergence izvajanja Hopfieldovega modela temelji na funkciji stanja nevronske mreže, ki ji Hopfield pravi *energija* stanja. Energija je navzdol omejena. Z iteracijami energija pada in doseže minimum v fiksni točki po končnem številu iteracij (Hopfield 1982). Energija je definirana z:

$$E(S_1, ..., S_N) = -\frac{1}{2} \sum_j \sum_{i\neq j} S_j S_i T_{ji} \qquad (25)$$

Energijska funkcija dejansko meri različnost med trenutnimi aktivacijskimi nivoji nevronov in njihovimi stanji. Logaritem funkcije podobnosti za BNM (15) pokaže analogijo s Hopfieldovo energijo:

$$E(S_1, ..., S_N) = -\frac{1}{2} log_2 Sim(S_1, ..., S_N) \qquad (26)$$

ker velja

$$\begin{aligned} log_2 Sim(S_1, ..., S_N) &= \sum_{S_j=1} log_2 \frac{P(S_j|S_1, ..., S_N)}{P(S_j)} \\ &= \sum_j S_j log_2 \frac{P(S_i|S_1, ..., S_N)}{P(S_j)} \\ &= \sum_j S_j \sum_{i\neq j} S_i T_{ji} \\ &= \sum_j \sum_{i\neq j} S_i S_j T_{ji} \end{aligned} \qquad (27)$$

Najpomembnejša razlika med obema funkcija je kot pri kombinacijskih funkcijah v različnih utežeh, oziroma v drugačni informaciji dobljeni iz množice učnih primerov. Iz relacije (27) sledi, da bi bilo energijo možno interpretirati tudi kot *entropija* ali *informacijska vsebina* stanja mreže. Minus logaritem (15) lahko interpretiramo kot vsoto informacijskih prispevkov aktivnih nevronov k zaključku, da so ostali trenutno aktivni nevroni zares aktivni. Ker se vsak informacijski prispevek zaradi simetrije pojavi dvakrat v vsoti, je potrebna še konstanta 1/2. Fiksna točka se interpretira kot stanje z minimalno informacijsko vsebino.

4.2 Lupina ekspertnega sistema

Lupina ekspertnega sistema, ki temelji na BNM, je dejansko BNM z integrirano možnostjo obrazložitve izvajanja. Vsak nevron predstavlja dogodek. Če je nevron aktiven, se je dogodek zgodil, sicer se ni oziroma ne vemo, če se je. Uteži na sinapsah se lahko določijo z učenjem, ali pa jih določi človek ekspert na danem problemskem področju.

Ko so uteži določene, je ekspertni sistem pripravljen odgovarjati na vprašanja. Vprašanje ustreza

prezentaciji situacije tako, da se podajo vrednosti nevronom za dogodke, za katere vemo, da so se zgodili. Naloga ekspertnega sistema je določiti najbolj verjetne vrednosti stanj vseh nevronov, tudi tistih, za katere vrednosti niso bile podane v naprej. Seveda, je lahko opis situacije tudi šumen. Torej mora ekspertni sistem aproksimirati manjkajoče podatke in odpraviti napake v vhodnih podatkih.

Ekspertni sistem iterativno spreminja vrednosti nevronov. Vsaka sprememba vrednosti predstavlja en sklep v verigi sklepov ekspertnega sistema. Vsak sklep lahko interpretiramo kot vsoto informacijskih prispevkov ostalih nevronov za BNM-P oziroma vsoto uteži evidence za BNM-odds.

Ko pride izvajanje v fiksno točko dobimo končni odgovor tako, da preverimo stanja vseh nevronov. Po želji lahko preverimo tudi aktivacijske nivoje nevronov, ki so lahko tudi koristna informacija. Npr., če neki nevron ni aktiven, a ima precej visok aktivacijski nivo, potem lahko sklepamo, da se je dogodek, ki ga nevron predstavlja, zgodil z določeno verjetnostjo.

BNM lahko obrazloži svojo odločitev, čeprav nima predstavljenega znanja v simbolični obliki. Očitno ekspertni sistem, ki temelji na BNM, ne vsebuje *baze znanja* v klasičnem pomenu besede. Znanje je shranjeno v utežeh na sinapsah v mreži. Vsaka utež se lahko direktno interpretira kot apriorna verjetnost dogodka oziroma dveh dogodkov hkrati. Pravila, po katerih ekspertni sistem sklepa, pa niso eksplicitno shranjena v bazi znanja ampak se dinamično kreirajo po potrebi.

Pristop je do določene mere analogen Quinlanovemu sistemu INFERNO (Quinlan 1983). Struktura mreže je podobna. Prednost sistema INFERNO je ta, da so lahko odločitve neeksaktne in nezanesljive. Generalizacija BNM na zvezna s nja bi odpravila nezmožnost BNM, da predstavi neeksaktno znanje. Po drugi strani je pri implementaciji sistema INFERNO problem tična konvergenca izvajanja, zaradi katere je potrebno vpeljevati različne strategije kontrole. Pri BNM pa je konvergenca izvajanja dokazana. Prav tako je zmožnost obrazložitve izvajanja sistema INFERNO omejena, medtem ko je interpretacija izvajanja BNM naravno sledi iz definicije informacije (Shannon & Weaver 1949).

Klasični sistemi za induktivno učenje kot npr. C4 (Quinlan 1986), CN2 (Clark & Niblett 1987) in ASISTENT (Kononenko 1985b, Bratko & Kononenko 1987, Cestnik in sod. 1987) se lahko uporabljajo za generiranje baze znanja za ekspertne sisteme. Generalizacija nad učnimi primeri tipično pripelje do *majhne množice splošnih pravil* za klasifikacijo v dani problemski domeni. Generalizacija v BNM pa generira *veliko množico specializiranih pravil*, ki ustrezajo fiksnim točkam. S povečevanjem števila fiksnih točk postaja BNM vse bolj podobna naivnemu Bayesu, kjer je vsaka točka v problemskem prostoru pokrita s svojim pravilom.

Prednost BNM je tudi preprost in učinkovit algoritem učenja, ki je tudi *inkrementalen*. Učenje v sistemih za induktivno gradnjo odločitvenih pravil je časovno zahtevno in za inkrementalno učenje je ponavadi potrebno dodajati posebne mehanizme v algoritem učenja. Poleg tega je BNM večsmerna, torej lahko ekspertni sistem uporabimo za sklepanje v različne smeri. Iteracije pri sklepanju poskušajo aproksimirati manjkajoče podatke in popraviti napačne podatke.

Zanesljiva aproksimacija verjetnosti je ključnega pomena pri zanesljivosti odločanja. Prednost BNM pred sistemi za induktivno učenje je v tem, da je aproksimacija verjetnosti polj zanesljiva, ker je narejena iz večje množice primerov. Zaradi tega v mnogih problemskih domenah, kjer je nekaj sto učnih primerov na voljo, naivni Bayes močno presega klasične sisteme za induktivno učenje po klasifikacijski natančnosti (Cestnik 1990).

5 Eksperimenti v medicinski diagnostiki

5.1 Opis podatkov in eksperimentov

Za testiranje BNM smo uporabili podatke za štiri medicinske diagnostične probleme: lokalizacija primarnega tumorja, prognostika ponovitve raka na dojki, diagnostika obolenj ščitnice in diagnostika revmatoloških obolenj, ki smo jih dobili iz Univerzitetnega kliničnega centra v Ljubljani. Podatki predstavljajo opise pacientov, ki so se zdravili na Onkološkem inštitutu v Ljubljani, V Kliniki za nuklearno medicino v Ljubljani oziroma v Bolnici Petra Držaja, Revmatološki Kliniki v Ljubljani.

Vsak pacient je opisan z množico atributov. Vsak atribut lahko zavzema določeno število vrednosti. Za vsakega pacienta je tudi znana končna diagnoza. Naloga mreže je naučiti se diagnosticiranja novih pacientov na osnovi učnih primerov, to je pacientov z znanimi diagnozami. Osnovne karakteristike štirih množic podatkov so podane v tabeli 1.

Diagnostična natančnost zdravnikov specialistov je povprečje natančnosti štirih specialistov iz dane domene na naključno izbrani testni množici pacientov. Natančnost diagnosticiranja zdravnikov je bila testirana v Univerzitetnem kliničnem centru v Ljubljani. Natančnost zdravnikov skupaj s številom razredov in entropijo na grobo ocenjuje težo problema. Očitno je problem lokacije primarnega tumorja najtežji, saj je možnih 22 razredov, entropija pa je 3.64 bita. Najlažji problem je prognostika raka na dojki, kjer sta 2 razreda in entropija 0.72.

Število atributov približno pove, kako natančno so opisani primeri. Povprečno število vrednosti na atribut po eni strani govori o natančnosti opisa, po drugi

domena	primarni tumor	rak na dojki	ščitnica	revmat.
# primerov	339	288	884	355
# razredov	22	2	4	6
# atributov	17	10	15	32
povpr. # vred./atribut	2.2	2.7	9.1	9.1
povpr.#manjk.pod/primer	0.7	0.1	2.7	0.0
večinski razred	25%	80%	56%	66%
entropija razredov	3.64 bitov	0.72 bitov	1.59 bitov	1.70 bitov
# nevronov za BNM-P	59	29	141	298
# nevronov za BNM-odds	25	17	43	111
natančnost zdravnikov	42%	64%	64%	56%

Tabela 1: Opis podatkov za štiri medicinske diagnostične probleme

strani pa skupaj s številom učnih primerov in številom razredov o zanesljivosti relativnih frekvenc dobljenih iz učnih primerov. Povprečno število manjkajočih podatkov na en učni primer govori o (ne)popolnosti podatkov, ki so bili na voljo. Najbolj nepopolni podatki so v domeni diagnosticiranja obolenj ščitnice, kjer v povprečju manjkajo vrednosti treh atributov pri enem primeru. Najbolj popolni podatki so v domeni revmatologija, kjer od 355 primerov samo pri štirih manjka vrednost enega atributa.

Podatek o procentu primerov iz večinskega razreda pomeni klasifikacijsko natančnost klasifikatorja, ki klasificira vsak primer v večinski razred. Torej bi tak preprost klasifikator močno presegel klasifikacijsko natančnost zdravnikov v revmatologiji (za 10%) in v prognostiki raka na dojki (za 16%). To kaže na zelo šibko informativnost atributov, hkrati pa tudi na to, da klasifikacijska natančnost ni zadosti dober način merjenja uspešnosti klasifikacije.

Neusmerjena BNM ne dela razlike med atributi in razredi. Razred je opisan s posebnim dodatnim atributom. Primeri za BNM-P so kodirani tako, da vsaki vrednosti vsakega atributa ustreza en nevron. Tako je število nevronov za BNM-P enako številu vseh vrednosti vseh atributov plus številu razredov. Če dani primer ima dano vrednost atributa, potem je ustrezni nevron aktiven, sicer neaktiven. Če vrednost atributa ni znana, so vsi nevroni, ki ustrezajo vrednostim tega atributa, neaktivni. Definicija izvajanja BNM-P, ki je asinhrono, implicira, da je za vsak atribut lahko naenkrat aktiven samo en nevron.

BNM-odds predpostavlja samo binarne atribute, zato je potrebno atribute binarizirati. Tako je npr. atribut s tremi vrednostmi lahko predstavljen s dvema binarnima atributoma, 22 razredov v primarnem tumorju pa s 5 binarnimi atributi. Dve vrednosti binarnega atributa ustrezata dvem stanjem enega nevrona. V tem primeru neznane vrednosti ni možno direktno predstaviti in je potrebno uvesti posebno stanje nevrona za neznano vrednost To stanje se spremeni v bolj verjetno vrednost od dveh zanih vrednosti že po prvi iteraciji.

Druga možnost je ta, da uporabljamo isto kodiranje kot pri BNM-P. V tem primeru je en atribut predstavljen s toliko binarnimi atributi, kolikor ima ta atribut vrednosti. V tabeli je podano število nevronov za BNM-odds, ki jih dobimo, če kodiramo na prvi način.

V enem eksperimentu smo celotno množico primerov naključno razbili na 70% primerov za učenje in 30% za testiranje. Iz učnih primerov je mreža izračunala vrednosti uteži na sinapsah in v nevronih. Mrežo smo testirali tako, da smo za vsak testni primer zbrisali razred in "pokazali" mreži primer. Tako so na začetku vsi nevroni, ki so predstavljali posamezne razrede, bili neaktivni. Nato je mreža asinhrono spreminjala svoja stanja, dokler ni prišla v fiksno točko. Odgovor mreže smo dobili tako, da smo poiskali aktivni nevron, med nevroni, ki so predstavljali razred. Če noben od teh nevronov ni bil aktiven, smo klasificirali v razred, ki ga je predstavljal nevron z *maksimalnim aktivacijskim nivojem.*

5.2 Rezultati

Za simulacijo asinhronega izvajanja BNM je potrebno določiti pravilo izbiranja nevrona, ki v dani iteraciji spremeni svoje stanje. V poskusih smo uporabljali 3 pravila:

Naklj.: Naključna izbira nevorna, med vsemi nevroni, ki lahko spremenijo svoje stanje po enačbi (14).

Prob.: Nevron je izbran z verjetnostjo, ki je proporcionalna aktivacijskemu nivoju nevrona.

Max.: Izbran je nevron z maksimalnim aktivacijskim nivojem. Za implementacijo tega kriterija izbire nevrona lokalna informacija ne zadošča, zato ni primeren za implementacijo v nevronskih mrežah.

Poleg klasifikacijske natančnosti smo merili tudi povprečno informacijsko vsebino odgovora (Kononen ko & Bratko 1987,1989), ki izniči vpliv apriornih verjetnosti razredov in se lahko uporablja za ocenjevanje nepopolnih odgovorov. Zaradi kompleksnosti je definicija informacijske vsebine odgovora podana v dodatku. Merili smo tudi povprečno število iteracij (sprememb stanj nevronov), ki so bile potrebne, da je mreža prišla v fiksno točko.

Simulirali smo obe topologiji BNM-P: s povratnimi vezmi in brez povratnih povezav nevronov samih s seboj. Hkrati smo testirali BNM-P brez iteracij, kar ustreza rezultatom naivnega Bayesovega klasifikatorja. Tako dobimo za vsako domeno 3x2 + 1 rezultatov, ki so podani v tabelah 2.1-2.4. Vsak eksperiment smo ponovili 10 krat z naključnim razbitjem vseh primerov na učno in testno množico. Rezultati v tabelah so povprečje desetih poskusov.

Iz informacijske vsebine odgovorov lahko sklepamo, da so v domeni lokalizacije primarnega tumorja atributi najbolj informativni, v domeni prognostike raka na dojki pa najmanj. Sicer se pa relativni trend informacijske vsebine v grobem ne razlikuje od trenda klasifikacijske natančnosti.

Najslabše rezultate dosega BNM-P, ko je izbira nevrona za spremembo stanja naključna, najboljše pa, ko se izbere nevron z maksimalnim aktivacijskim nivojem. Kot je bilo za pričakovati, je v prvem primeru potrebnmo največ iteracij, da mreža skonvergira v fiksno točko, in v drugem primeru najmanj.

Iz rezultatov je razvidno, da osnovna BNM-P dosega precej slabšo klasifikacijski natančnost kot sam naivni Bayes. Vidi se, da čim več iteracij je potrebnih, da mreža skonvergira v fiksno točko, tem slabša je klasifikacijska natančnost. Torej iteracije med izvajanjem BNM-P preveč pokvarijo vhodni vzorec, tako da je rezultat nezanesliv. BNM-P s povratnimi povezavami skuša zmanjšati število iteracij, tako da povratna povezava skuša zadržati nespremenjeno stanje nevrona. Število iteracij je zato manjše in natančnost boljša.

5.3 Omejevanje prehodov nevronov

Vendar iteracije bi lahko bile koristne za aproksimacijo manjkajočih podatkov in popravljanje napačnih podatkov, če bi bile spremembe nevronov zadosti zanesljive. Zanesljivost lahko povečamo tako, da vpeljemo dva praga v odločitveno funkcijo (14):

$$D_j(X) = \begin{cases} 1, & \text{če } X \geq Prag_1 \\ S_j, & \text{če } Prag_1 > X > Prag_2 \\ 0, & \text{če } X \leq Prag_2 \end{cases} \quad (28)$$

pri čemer velja $Prag_1 \geq P(S_j) \geq Prag_2$, kar zagotavlja konvergenco. S tem ko omejimo prehode nevronov, se poveča število fiksnih točk, in bo potrebno število iteracij manjše. V poskusih smo prevzeli vrednosti 1 in 0 za zgornji in spodnji prag v odločitveni funkciji. Zanimivo je, da sedaj praktično ni razlike med obema topologijama in različnimi kriteriji za izbiro nevrona. Vseh šest različnih variant mreže dosega isto natančnost, verjetno zato, ker je število iteracij drastično nižje kot prej. Rezultati so podani v tabeli 3. Rezultati BNM-P, ki uporablja odločitveno funkcijo (28), so nekoliko boljši od naivnega Bayesovega klasifikatorja.

Ker je *spominski obseg* (memory capacity) nevronske mreže povezan s številom fiksnih točk, (Guez in sod. 1988, McEliece in sod 1987), Lahko pričakujemo, da bo s povečanjem števila fiksnih točk povečan tudi sam spominski obseg mreže. Rezultati, dobljeni pri testiranju mreže nad učnimi primeri, ki so predstavljeni v tabeli 4, potrjujejo to domnevo.

5.4 Primerjava z drugimi klasifikatorji

V tabeli 5 je podana primerjava natančnosti BNM-P, naivnega Bayesa, Asistenta (Cestnik in sod. 1987) in zdravnikov specialistov. Na istih podatkih smo testirali tudi Hopfieldov model, vendar je Hopfieldova nevronska mreža skoraj vedno skonvergirala v fiksno točko, v kateri je bil rezultirajoči razred enak večinskemu razredu. Hopfieldova mreža brez iteracij kot analogija naivnemu Bayesovemu klasifikatorju je dosegala približno enake rezultate kot Hopfieldova mreža z iteracijami. Ker rezultati Hopfieldovega modela niso priemerljivi z ostalimi, so izpuščeni iz tabel.

5.5 Ocenjevanje razumljivosti interpretacije

V razdelku 4 smo podali interpretacijo klasifikacije z naivnim Bayesovim klasifikatorjem kot vsoto informacijskih prispevkov posameznih nevronov h končni odločitvi. Da bi ocenili razumljivost, pravilnost in uporabnost take razlage smo iz vsake medicinske domene naključno izbrali 10 testnih primerov, 5 takih, ki jih je naivni Bayes pravilno klasificiral, in 5 nepravilno klasificiranih.

Za klasifikacijo vsakega od pacientov smo izpolnili formular, kot ga prikazuje tabela 6. Na vrhu formularja je bil vsak pacient opisan z vrednostmi vseh atributov. V spodnjem delu je bila izpisana za vsako možno diagnozo apriorna verjetnost diagnoze, verjetnost diagnoze izračunana z naivnim Bayesom, seznam atributov, ki

povratne vezi	kriterij	natančnost	informacijska vsebina	# iteracij
NE	Naklj	28.8%	0.74 bitov	16.2
NE	Prob	36.2%	1.10 bitov	13.9
NE	Max	37.2%	1.15 bitov	11.9
DA	Naklj	31.5%	0.90 bitov	9.8
DA	Prob	39.5%	1.19 bitov	9.4
DA	Max	41.0%	1.27 bitov	8.7
naivni Bayes		47.1%	1.42 bitov	0.0

Tabela 2.1: Rezultati BNM-P v domeni primarni tumor.

povratne vezi	kriterij	natančnost	informacijska vsebina	# iteracij
NE	Naklj	63.1%	-0.23 bitov	10.6
NE	Prob	73.2%	0.05 bitov	9.8
NE	Max	70.7%	-0.05 bitov	8.9
DA	Naklj	69.9%	-0.07 bitov	4.2
DA	Prob	70.0%	-0.05 bitov	4.2
DA	Max	72.5%	0.04 bitov	4.1
naivni Bayes		77.6%	0.15 bitov	0.0

Tabela 2.2 Rezultati BNM-P v domeni rak na dojki

povratne vezi	kriterij	natančnost	informacijska vsebina	# iteracij
NE	Naklj	45.9%	0.33 bitov	13.3
NE	Prob	50.3%	0.40 bitov	11.8
NE	Max	54.6%	0.48 bitov	10.8
DA	Naklj	52.9%	0.46 bitov	9.1
DA	Prob	59.8%	0.58 bitov	8.3
DA	Max	61.9%	0.63 bitov	8.2
naivni Bayes		67.8%	0.73 bitov	0.0

Tabela 2.3 Rezultati BNM-P v domeni ščitnica.

povratne vezi	kriterij	natančnost	informacijska vsebina	# iteracij
NE	Naklj	40.2%	0.03 bitov	24.9
NE	Prob	44.4%	0.09 bitov	24.3
NE	Max	44.9%	0.15 bitov	22.4
DA	Naklj	43.0%	0.10 bitov	25.0
DA	Prob	43.6%	0.08 bitov	23.7
DA	Max	47.3%	0.18 bitov	23.7
naivni Bayes		50.5%	0.21 bitov	0.0

Tabela 2.4: Rezultati BNM-P v domeni revmatologija.

domena	natančnost	inf.vsebina	# iteracij
primarni tumor	47.9 %	1.46 bitov	0.6
rak na dojki	78.1 %	0.15 bitov	0.4
ščitnica	68.2 %	0.74 bitov	1.7
revmatologija	59.7 %	0.42 bitov	10.3

Tabela 3: Rezultati BNN-P z odločitveno funkcijo (28) in pragoma $Prag_1 = 1$ in $Prag_2 = 0$.

	odločitvena funkcija (14)			odločitvena funkcija (28)		
domena	natančn.	inf.vseb.	#iter.	natančn.	inf.vseb.	#iter.
primar.tum.	55.5%	1.91 bitov	8.3	60.5%	2.06 bitov	0.6
rak na doj.	72.7%	0.11 bitov	2.8	79.7%	0.19 bitov	0.4
ščitnica	71.9%	0.89 bitov	5.8	75.2%	0.95 bitov	0.6
revmatolog.	82.0%	1.32 bitov	8.6	84.8%	1.38 bitov	3.9

Tabela 4: Primerjava klasifikacijske natančnosti različnih odločitvenih funkcij na učnih množicah.

	prim. tumor		rak na dojki		ščitnica		revmatologija	
klasifikator	nat.	inf.vseb.	nat.	inf.vseb.	nat.	inf.vseb.	nat.	inf.vseb.
BNN-P	48%	1.46 bita	78%	0.15 bita	68%	0.74 bita	60%	0.42 bita
naivni Bayes	47%	1.42 bita	78%	0.15 bita	68%	0.73 bita	50%	0.21 bita
Asistent	44%	1.38 bita	77%	0.07 bita	73%	0.87 bita	61%	0.46 bita
zdravniki	42%	1.22 bita	64%	0.05 bita	64%	0.59 bita	56%	0.26 bita

Tabela 5: Primerjava uspešnosti različnih klasifikatorjev v štirih medicinskih domenah.

opis pacienta:

atribut	vrednost	atribut	vrednost	atribut	vrednost
Att 1	Val i	Att 2	Val j	Att 3	Val k
...					

diagnoza pacienta			RAZLAGA			
diag.	verjet.	(aprior)	atribut za		atribut proti	
D_1	XX %	(YY %)	Att i	Inf_i	Att j	Inf_j
			Att k	Inf_k	Att l	Inf_l
...			...		...	

Tabela 6: Formular za razlago diagnoze enega pacienta. YY je apriorna verjetnost dane diagnoze in XX verjetnost, ki jo je vrnil klasifikator.

podpirajo dano diagnozo in seznam atributov, ki govorijo proti dani diagnozi. Zraven vsakega atributa je bil izpisan tudi informacijski prispevek atributa za/proti dano diagnozo.

Tako izpisane formularje smo dali v oceno zdravnikom specialistom v dani medicinski domeni. Pri tem zdravniki niso dobili podatka o pravilni končni diagnozi pacienta. Zaprosili smo jih naj od 0 do 10 ocenijo 6 parametrov, kot jih prikazuje tabela 7. V tabeli so podane povprečne ocene za klasifikacijo vseh 10 pacientov in povprečne ocene za klasifikacijo 5 pacientov, ki jih je naivni Bayes narobe klasificiral.

Uporabnost v ščitnici in revmatologiji niso ocenjevali, ker je način diagnosticiranja v praksi precej drugačen, z uporabo drugih podatkov, kot so bili nam na voljo. Iz ocen v tabeli je razvidno, da se v treh domenah (razen revmatologije) pravilnost diagnoze in pravilnost razlage močno ujema z zdravnikovim mišljenjem in da je sama razlaga zdravnikom razumljiva. Sam klasifikator se je zdel specialistom zelo uporaben za študente medicine, malo manj za zdravnike nespecialiste in najmanj a vseeno vsaj delno uporaben za zdravnike specialiste. Ker ni bistvenih odstopanj med povprečji klasifikacij vseh 10 pacientov in 5 narobe klasificiranih pacientov, lahko sklepamo, da dela sistem podobne napake kot sami zdravniki specialisti.

V revmatologiji so ocene nekoliko nižje kot v drugih domenah. Podrobna analiza je pokazala, da je razlaga pravilna, če opasujemo atribute neodvisno z drugimi. Ker v tej domeni obstaja nekaj delnih korelacij med atributi, jih ne smemo obravnavati popolnoma neodvisno.

Splošen vtis zdravnikov je bil, da je način klasifikacije sistema analogen njihovemu, saj oni tudi seštevajo evidenco za/proti diagnozi. V primeru, ko se niso strinjali z odločitvijo sistema so bili mnenja, da je sistem naredil napako.

6 Zaključki

Bayesove nevronske mreže lahko temeljijo na verjetnosti ali na razmerju verjetnosti. Večsmerne BNM so analogne Hopfieldovemu modelu.

Razlika med Hopfieldovim modelom in večsmerno BNM je v učnem pravilu. Za učenje BNM zadošča osnovno Hebbovo pravilo, za katerega je precej nevrofiziološke evidence, da se po tem pravilu učijo biološki nevroni. Uteži na sinapsah ustrezajo apriornim verjetnostim, da sta povezana nevrona aktivna. Hopfieldov model uporablja posplošeno Hebbovo pravilo. Uteži na sinapsah ustrezajo verjetnostim, da sta povezana nevrona v istem stanju, kar nudi manj informacije kot uteži pri BNM. Kompleksnost učenja in izvajanja je pri obeh modelih enaka, po klasifikacijski natančnosti pa BNM daleč presega Hopfieldov model.

Za opis stanja modela Hopfield uporablja energijsko funkcijo, ki je analogna funkciji podobnosti za opis stanja večsmerne BNM. Logaritem funkcije podobnosti lahko interpretiramo kot entropijo ali informacijsko vsebino sistema. Iz tega sledi, da je fiksna točka stanje sistema z lokalno minimalno entropijo.

Prednost večsmernih BNM je večja robustnost glede manjkajočih podatkov in šuma v podatkih. Ker ni razlike med atributi in razredi, lahko klasifikacija poteka od atributov k razredom in obratno. Učni algoritem je za razliko od sistemov za induktivno učenje, kot so ID3 in Asistent, relativno hiter in inkrementalen. Učenje in izvajanje lahko poteka paralelno.

	primarni tumor	rak na dojki	ščitnica	revmat.
pravilnost diagnoze	7.7/6.0	8.0/8.0	7.9/7.6	5.2/3.6
pravilnost razlage	5.9/5.2	7.4/7.8	8.2/8.2	3.8/4.3
razumljivost	7.6/6.4	8.4/8.6	7.9/8.2	5.0/5.0
uporabno za specialiste	4.5/5.0	5.8/5.4	-	-
uporabno za nespecial.	6.0/5.6	7.3/7.0	-	-
uporabno za študente	7.6/6.2	8.2/8.2	-	-

Tabela 7: Rezultati zdravniških ocen formularja za razlago diagnoze v štirih medicinskih diagnostičnih problemih. Vsaka ocena je v obliki VSI/NAP, kjer je VSI povprečna ocena za vseh 10 klasifikacij pacientov, NAP pa povprečna ocena za 5 napačno klasificiranih pacientov. Ena ocena je število med 0 in 10.

BNM-P ima prednost pred BNM-odds, ker se lahko entropija direktno posploši na več kot dva disjunktna dogodka. BNM-P je zato bolj fleksibilna in računsko manj zahtevna. BNM-odds je primerna samo za binarne atribute in razrede in je manj primerna za predtavitev manjkajočih podatkov. V BNM-odds je lahko manjkajoči podatek predstavljen z apriornim odds samo med učenjem. Interpretacija izvajanja BNM-P je bolj naravna (prispevek informacije v bitih) kot interpretacija BNM- odds (utež evidence). Po drugi strani pa BNM-odds potrebuje bistveno manj nevronov za predtavitev istega problemskega prostora kot BNM-P.

ID3 in Asistent sta prav zaradi nenaivnosti in nezanesljive aproksimacije verjetnosti bolj primerna za eksaktne domene, medtem ko je naivni Bayes zaradi zanesljive aproksimacije verjetnosti primeren za mehke domene, v katerih je atribute definiral človek strokovnjak. Človek tipično definira relativno neodvisne atribute zaradi lažjega (linearnega) razmišljanja. Zato je predpostavka o neodvisnosti atributov v takih domenah sprejemljiva.

Prednost naivnih BNM pred odločitvenim drevesom je tudi v obravnavanju manjkajočih podatkov. Če nimamo podatka za atribut v drevesu postane klasifikacija v precejšnji meri nezanesljiva. Naivna BNM atribut, za katerega ni podana vrednost, preprosto ne upošteva pri klasifikaciji. Poleg tega BNM pri klasifikaciji upošteva vse razpoložljive atribute, medtem ko v odločitvenem drevesu tipično nastopa le majši del razpoložljivih atributov. Zdravniki so mnenja, da pravila v odločitvenih drevesih tipično vsebujejo premalo atributov, preskromno opisujejo pacienta in je zaradi tega diagnosticirenje nezanesljivo (Pirnat in sod. 1989, Kononenko 1989b).

Največja slabost nevronskih mrež v primerjavi s klasičnimi sistemi umetne inteligence je v nezmožnosti obrazložitve svoje odločitve. Za razliko od Hopfieldovega modela, lahko BNM obrazložijo svojo odločitev. Pri zmožnosti obrazložitve svoje odločitve se je pokazala prednost naivnega Bayesa pred odločitvenim drevesom. Odločitveno drevo za obrazložitev ponudi samo vejo drevesa, po kateri je bil dani primer klasificiran. Naivni Bayes obrazloži svojo odločitev kot vsoto informacijskih prispevkov posameznih atributov za/proti dani odločitvi. Taka obrazložitev se je pokazala kot naravna, ljudem strokovnjakom razumljiva in uporabna, ker je podobna človeškemu načinu odločanja.

Generalizacija v sistemih za induktivno učenje generira malo splošnih pravil. Znanje je predtavljeno v simbolični obliki. Učenje v BNM generira mnogo razdrobljenega znanja, ki je predstavljeno v obliki apriornih verjetnosti. Pri izvajanju se te verjetnosti kombinirajo v specializirana pravila za reševanje konkretnih problemov.

Z nadaljnjimi raziskavami bo potrebno rešvati naslednja odprta vprašanja:

- *Diskretizacija zveznih atributov*: V tem delu smo predpostavljali, da so zvezni atributi v naprej diskretizirani, tako da ni bilo pri obravnavanju posameznih vrednosti nobene razlike med diskretnimi in zveznimi atributi. Ker se ni upoštevala urejenost vrednosti, je s tem bila zavržena (mogoče) koristna informacija. Poleg tega je vnaprejšnja diskretizacija premalo fleksibilna. Lahko pride do po nepotrebnem prevelike razdrobljenosti učne množice, kot je to bilo v domeni ščitnice. Sistem Asistent je v tej domeni dosegel boljšo diagnostično natančnost, ker med samo gradnjo drevesa določi meje zveznim atributom tako, da učne množice ne razdrobi preveč zaradi zanesljivejše aproksimacije verjetnosti.

  Verjetno najbolj primeren pristop v BNM je vnaprejšnja diskretizacija zveznih atributov z mehkimi mejami intervalov. Pri mehkih mejah en primer ne pripada samo enemu intervalu, ampak več intervalom, vsakemu z določeno verjetnostjo. S tem se ohranja množica primerov skupaj, ohranja se informacija o ure-

jenosti intervalov, s parametrom "mehkosti" pa lahko kontroliramo razdrobljenost atributa na več vrednosti. Ta pristop lahko posplošimo na mehke vrednosti diskretnih atributov.

- *Določanje pragov za prehode nevronov*: Pri osnovni večsmerni BNM, ki uporablja odločitveno funkcijo (14) iteracije preveč pokvarijo vhodni vzorec, zato je končna klasifikacija slabša kot pri naivnem Bayesovem klasifikatorju. Z uvedbo odločitvene funkcije (28) je večsmerna BNM boljša od naivnega Bayesa, ker odločitvena funkcija dovoljuje samo zanesljive prehode nevronov. Tako iteracije uspešno popravljajo šum in nadomeščajo manjkajoče podatke. V nadaljnjem delu bo potrebno bolj formalno raziskati razmerje med zanesljivostjo prehodov nevronov in uspešnostjo klasifikacije, in razviti metodo za avtomatsko nastavitev pragov odločitvene funkcije. Metoda bo verjetno odvisna od količine manjkajočih podatkov in od ocenjenega šuma v podatkih. Za vsak nevron bi lahko določili tudi zanesljivost vhodnega podatka, ki bi vplivala na določitev pragov.
- *Eksperimentalno testiranje robustnosti BNM*: Potrebno je sistematično preveriti robustnost glede šuma, manjkajočih podatkov in glede števila uničenih nevronov. Treba je preveriti možnost bolj distribuirane reprezentacije za povečanje robustnosti glede okvar nevronov, kar bi lahko s pridom uporabili pri strojni realizaciji BNM.
- *Strukturirana BNM*: Zanimivo bi bilo posplošiti BNM na strukturirano BNM po nivojih, kjer bi bil en nevron mrež na višjem nivoju realiziran z BNM na nižjem nivoju. Tako strukturirana BNM se lahko uporabi za reševanje kompleksnejših problemov, ki se dajo razdeliti na manjše podprobleme.
- *Posplošitev delno naivnega Bayesa na predikatni račun prvega reda*: Na analogen način, kot je Quinlan posplošil induktivno učenje odločitvenih pravil v atributnem jeziku na induktivno učenje Hornovih stavkov v sistemu FOIL (Quinlan 1989), je smiselno poskusiti posplošiti delno naivni Bayes na jezik z močnejšo izrazno močjo.

ZAHVALA

Pri zbiranju podatkov za eksperimentiranje je bila neprecenljiva pomoč zdravnikov specialistov dr. Matjaža Zwittra, dr. Sergeja Hojkerja in dr. Vlada Pirnata iz Univerzitetnega Kliničnega Centra v Ljubljani. Zahvaljujem se jim za zbiranje in interpretacijo podatkov, za izvedbo anket za oceno diagnostične natančnosti zdravnikov specialistov, za interpretacijo rezultatov. Raziskave opisane v tem delu so bile opravljene v Laboratoriju za umetno inteligenco na Fakulteti za elektrotehniko in računalništvo v Ljubljani, katerega predstojnik je prof. dr. Ivan Bratko.

REFERENCE

Beuscart R. , Duhamel A., Deleu J. (1989) Belief Theory and Expert Systems, submitted paper.

Bratko, I. & Kononenko, I. (1987) Learning Diagnostic Rules from Incomplete and Noisy Data. In Phelps, B. (ed) *Interactions in Artificial Intelligence and Statistical Methods.* Technical Press.

Cestnik B. (1990) Estimating Probabilities: A Crucial Task in Machine Learning, *Proc. ECCAI 90*, Stockholm, August 1990.

Cestnik, B. , Kononenko, I., Bratko, I. (1987) ASSISTANT 86 : A Knowledge Elicitation Tool for Sophisticated Users. In I.Bratko, N.Lavrač (eds.), *Progress in Machine learning.* Wilmslow, England: Sigma Press.

Clark, P., Niblett, T. (1987) Learning if then rules in noisy domains. In B. Phelps (ed.), *Interactions in Artificial Intelligence and Statistical Methods.* Hampshire, England: Technical Press.

Deleu J. , Beuscsrt R., Becquart E., Duhamel A., Comyn G. (1988) Reseu Bayesien & Diagnostic Medical (Bayesian Networks & Medical Diagnosis), *Proc. Neuro-Nimes 88: International Workshop on Neural Networks & their Applications'*, Nimes, France, Nov.1988.

Good I.J. (1950) *Probability and the weighing of evidence.* London: Charles Griffin.

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.

Hebb, D.O. (1949) *The Organization of Behavior.* New York: Wiley.

Hopfield J.J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences* 79:2554-2558.

Hopfield J.J. (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. of the National Academy of Sciences* 81:4586-4590.

Kononenko, I. (1985a) Strukturno avtomatsko učenje. *Informatica*, vol. 9, no. 4, pp. 44-55.

Kononenko, I. (1985b) Razvoj sistema za induktivno učenje ASISTENT, Magistrsko delo, Univerza v Ljubljani, Fakulteta za elektrotehniko.

Kononenko, I. (1989a) Bayesian neural networks. *Biological Cybernetics*, Vol 61, pp.361-370.

Kononenko,I. (1989b) Interpretation of neural networks decisions. *Proc. of IASTED Internat. Conf. on Expert Systems*, Zurich, Switzerland, June 26-28, pp 224-227.

Kononenko, I. (1989c) ID3, sequential Bayes, naive Bayes and Bayesian neural networks. *Proc. 4th European Working Session on Learning*, Montpellier, France, December 1989, pp.91-98.

Kononenko I. (1989d) Nevronske mreže, *Informatica*, letnik 13, št. 2, str. 56-71.

Kononenko I., Bratko I. (1987) Ocenjevanje uspešnosti sistemov za induktivno učenje, *Zbornik ETAN*, Bled, junij 1987

Kononenko, I. & Bratko, I. (1989) Informativity Based Evaluation Criterion for Classifier's Performance. Machine Learning Journal (submitted). (also: *Proc. ISSEK Workshop*, Udine, Sept. 1989)

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. (1989) Personal Models of Rationality. Journal of Statistical Planning. (in press).

Michie, D., A. Al Attar (1989) Use of Sequential Bayes with Class Probability Trees. To appear in: J.E. Hayes-Michie, D.Michie & E. Tyugu (eds.) *Machine Intelligence 12*, Oxford: Oxford University Press.

Pearl J. (1988) *Probabilistic reasonong in intelligent systems: Networks of plausible inference*, San Mateo: California: Morgan Kaufmann Publishers.

Pirnat V. , Kononenko I., Janc T., Bratko I. (1989) Medical Estimation of Automatically Induced Decision Rules, *Proc. of 2nd Europ. Conf. on Artificial Intelligence in Medicine*, City University, London, August 29-31 1989, pp.24-36.

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

Quinlan, J.R. (1983) Inferno: A Cautious Approach to Uncertain Inference, *The computer journal*, Vol. 26, No. 3, pp.255-269.

Quinlan, J.R. (1986) Induction of Decision Trees. *Machine Learning*. Vol. 1, Num. 1, pp. 81-106.

Quinlan J.R. (1989) Learning relations: Comparison of a symbolic and connectionist approach, *ISSEK Workshop*, Udine, Sept. 1989.

Rumelhart, D.E. , Hinton, G.E. & McClelland, J.L. (1986) A General Framework for Parallel Distributed Processing. in: Rumelhart, D.E. & McClelland, J.L. (eds.) *Parallel Distributed Processing, Vol. 1: Foundations*. Cambridge: MIT Press.

Shannon & Weaver (1949) *The mathematical theory of communications*. Urbana: The University of Illinois Press.

DODATEK: DEFINICIJA INFORMACIJSKE VSEBINE ODGOVORA

Nepristranska ocena klasifikacije mora upoštevati vpliv apriornih verjetnosti razredov, ki lahko onemogočijo trivialnemu klasifikatorju, ki vedno klasificira v najbolj verjeten razred, da doseže navidezno visoko klasifikacijsko natančnost. Ocena informacijske vsebine odgovora izniči vpliv apriornih verjetnosti, ocenjuje tudi nepopolne odgovore in se lahko uporablja za primerjavo klasifikacijske uspešnosti v različnih domenah. Interpretacija je naravna, namreč količina informacije v bitih.

Če je razred primera, ki ga klasificiramo, R, apriorna verjetnost razreda R $P_a(R)$ in verjetnost razreda R, ki smo jo dobili od klasifikatorja, $P(R)$, potem definiramo *informacijsko vsebino odgovora* posebej za dva primera:

$P(R) \geq P_a(R)$: V tem primeru smo od sistema dobili naslednjo količino informacije, da primer pripada razredu R:

$$Inf = -log_2 P_a(R) + log_2 P(R)$$

ali z besedami, dobljena količina informacije je enaka celotni potrebni količini informacije, da zvemo, da primer pripada razredu R, minus preostala potrebna količina informacije za klasifikacijo primera. V primeru, da je $P_a(R) = P(R)$ je $Inf = 0$, t.j. sistem ni spremenil apriornih verjetnosti in je dobljena količina informacije enaka 0.

$P(R) < P_a(R)$: V tem primeru smo od sistema dobili naslednjo količino informacije Ninf, da primer ne pripada razredu R:

$$Ninf = -log_2(1 - P_a(R)) + log_2(1 - P(R))$$

ali z besedami, dobljena količina napačne informacije je enaka celotni količini informacije, da primer ne pripada razredu R, minus preostala potrebna količina informacije za napačno klasifikacijo primera. Ker je ta informacija napačna, definiramo dobljeno količino informacije kot negativno:

$$Inf = -[-log_2(1 - P_a(R)) + log_2(1 - P(R))]$$

NOVICE IN ZANIMIVOSTI

NEWS

The Breakthrough of Unix

...thus the question arises whether IBM after 30 years of declaring the standards, will be forced to accept that there is a greater force in the marketplace ? ...

Systems International, Oct. 1989, pg. 94

As the production of the open standards (UNIX/POSIX for operating systems, OSF/Motif for user interface, X/Open for application interface, SQL for databases) continues, more firms continue to migrate to Unix. In the early 80's, few people would have believed that IBM would be selling a UNIX operating system for its mainframes by the end of the decade. Today, especially after the CeBit show, it has become evident that part of the IBM's strategy is to take its share of the demand for the open systems market. Furthermore, all other major hardware vendors have announced versions of UNIX for their platforms.

According to International Data Corporation, the information system managers choose Unix over VMS, MVS, OS/2, MS-DOS for three reasons. First, Unix is the only major multi-user, multitasking operating system available on systems ranging in size from personal computers to supercomputers. Second, Unix has very good communications and networking capabilities. Third, Unix provides a standard that is largely hardware and software independent.

With a widespread availability of Intel Corp's i386 and i486 microprocessor personal computers, Unix finally has a suitable hardware platform available for the desktop systems. Unix PC market growth rate for year 1989 was 62%, compared with 19% for the overall PC market. Furthermore, Unix mainframe market growth rate was 30%, while the overall mainframe market grew up by only 9%.

Why Unix?

Due to its wide availability, Unix became the de facto hardware- independent platform. Because the majority of the Unix operating system code is written in C, a high level programming language, Unix may be ported to to other platforms as soon as a C compiler is available for the target computer. Theoretically, it is then possible to take a application that was developed on a Unix microprocessor-based computer and have it run on a Unix supercomputer and vice versa, with just a compile and link of source code on the new platform.

Since the Unix market is driven by industry competition, *there is no single monopolistic controlling organization,* there is continuous enhancement to Unix and greater choice of ever growing software. Furthermore, there is no loss of investment in personnel, training and data when system is upgraded or changed, and vital corporate data is immediately available to approved personnel whenever required.

Issues of 1990's

According to the International Data Corporation's ongoing research, the most vital area of information science in the 90's will be the connection and management of distributed data resources. The successful information system department of the 90's will need a computer system that is sufficiently flexible to handle today's problems as well to handle problems that will be considered in the 90's. Of today's widely accepted operating sys-

tems, Unix's flexibility is unique in that it does meet the demands of today and tomorrow.

Unix and distributed data resources

Unix was designed to be the great communicator. In some situations Unix is sufficient to connect attached terminals to a host, in other situations, for example, a number of workstations may be connected together in a local-area network, linking a minicomputer in Ljubljana, which may in turn be connected to a host to a New York-based mainframe via a satellite link. The workstations may be transferring digital information as well as audio-visual signals, news, telex, fax. Networked Unix based systems successfully handle these situations. Furthermore, from both the end user's and developer's perspective, a networked UNIX computer system appears to be just one machine. Therefore, as in a well managed nondistributed system, a given piece of data resides in only one location on the Unix network. All applications needing to access the data, will merely reference the data.

Unix in Europe

In Europe, Unix in 1989 accounted for around 15% of computer systems sold ($ 3.7 billions), and it is expected to reach around 20% by 1991, according to Dataquest. In the Federal Republic of Germany, which takes 21% of all European computer market, the sales figures of 1989 point to a trend towards Unix. For example, the ISIS Unix Report from summer 1989 reported close to 2000 new applications in Unix offered by nearly 700 companies.

So far in Europe, Unix has been sold more by VARs (Value-Added Reseller) than direct sales. Most major suppliers sell around 50% through indirect channels. The reason for this is the structure of Europe markets, which are limited by language, national business, practices and laws. It is therefore normal to have VARs suppliers in Yugoslavia, too. One highly qualified supplier for Unix and open software in Yugoslavia is **PAREX,** *Institute for computer engineering and consultancy* in Ljubljana (Kardeljeva 8/III, phone (+38 61) 22 34 63). PAREX in collaboration with *The International Consortium for Open Software*, London offers various pre- and post-sale services such as installation, maintenance, application porting and development, and what is the most important, training and continuous program of technical courses.

Unix in Slovenia

Unix is in its infant stage in Yugoslavia, especially in Slovenia. Until recently there has been very little interest for Unix at all. There are at least three reasons for such low Unix interest.

First, there are no independent software VAR suppliers, which would offer Unix marketing, consultancy and training for the end users, as well as for the information builders.

Second, the strategy, or better, no strategy in computer purchasing. On one side, there has been a monopolistic stand by the large national corporations and representatives that have supplied proprietary systems for multi-user applications, even there, where the mainframe power was not necessary. On the other side, there has been a chaotic purchasing megalomania for PC computers, without software and application strategy and support for end users. Such a purchasing strategy, based solely on hardware and no application characteristics, has brought to a situation where there are many heterogeneous large mainframe systems, and infinitely many PCs without connections, portability, transparency, standards, mutual applications. In Croatia, for example, the situation is little different. The influence of large national computer corporations has been lower than in Slovenia, as well as the purchasing power of the end users, and therefore, numerous PC Unix systems have been installed for multi-user applications, as alternative for large mainframes and networks of PCs.

Third and the most important reason for the low Unix rate in Slovenia is the Unix avoidance by the educational program at the major academic institutions. In contrast to European and USA students, our students have very little academic, and absolutely no practical knowledge of Unix. Unix has been not taught at the department of computer science in Ljubljana, and it is not used at the Jožef Stefan Institute as a software tool. A positive exception is only the department of informatics at the University of Maribor, where three genera-

tions of students have had some exposure to Unix concepts and programming.

Hopefully, it seems that the better days are coming for Unix. According to a recent PAREX market research, information builders and end-users are willing and prepared to use Unix. There are several reasons for a sudden Unix positive trend. First, an active role of a Yugoslav Unix User Group, which for example at it's February 1990 conference brought well known Unix companies and organizations such as European Unix User Group, International Consortium for Open Software, Open Software Foundation, AT&T, IBM, Hewlett Packard to more than 150 participants from Yugoslavia. Second, more and more small VAR companies have be founded with a strategy to offer Unix and Open software services. Third, the University of Ljubljana has bought a Unix system and students can finally work and become familiar with the open software standards.

With this recent positive Unix trends, and the IBM's announcement at the CeBit Show, we hope that Unix would receive the share of the interest, that it ought to have long time ago.

A.P. Železnikar

Information and Postmodernism

Anton P. Železnikar
Volaričeva 8, 61111 Ljubljana, Yugoslavia

Introduction

In Middle Europe, several parallel informational markers and their explanations are appropriated to the movement, epoch, and society in which people are beginning to live, for instance, the postindustrial society, the information society, and the cultural (cult-ural) society (in German Kult-urgesellschaft). Within these movements of contemporary (postmodernistic) thought, concepts, ideas, and philosophies as informational phenomena, information as philosophic and technologic notion can in no way lose its significance and its thinking breakthroughs, however, has to be *redefined* in a *postmodernistic* way. The view of informational relevance still remains central for the development or evolution of human mind in the next epoch, on one side, and for the development of human's intelligent tools which might decide not only on survival of species, but also on their cultural coexistence (for instance, the intellectual arising of the so-called ecologic philosophy or ecosophy as information, of the new *postmodernistic spirit* as specific counter-information in regard to the modernistic epoch, etc.), on the other side.

In the last years, the Middle European philosophic orientation tends to the concept of *cultural society* (it might seem sometimes also in the sense of Kult-Urgesellschaft, i.e. cultic primitive society) which does not accept the so-called modernistic, natural (naturalistic, scientized), and technologic (computerized, automatized, cybernetic) concept of information (SEC). The philosophically traditional triplet *matter-energy-information* which was relevant for the development of natural sciences and technology seems to be passed by the comprehension of the arising postmodernistic spirit. This way of movement of the Middle European thought and spirit (sometimes Spirit) could be only a specific (as far as possible a subcultural) philosophic orientation, however, could become also the main European stream of thinking. What is the White Paper of the European Communities, in 1985, other than an early postmodernistic approach in the area of possible common European life and harmonious politics?

Modernism and Postmodernism

From the informational point of view, modernism and postmodernism are informational forms which are characteristic for two successive present epochs of human culture. Modernism and postmodernism are cultural forms as information and informing of information of human individuals and populations in the so-called modern (present) and postmodern (postpresent) epoch. In this respect, study and understanding of wisdom, philosophy, science, technology, style of life, politics, etc. might differ evolutionarily in the modern and the postmodern era.

To understand the movement of postmodernism as a reaction to the phenomenon of modernism it is senseful to determine the modernism in a broader, not only the strict modernism-determinative manner. The adjective **modern** comes

from late Latin *modernus* and this from Latin *modo* meaning *just now,* and further from Latin *modus* meaning *measure. To be modern* means also *to give out by measure, mete.* In English, **modern** means **1 a:** *of, relating to, or characteristic of a period extending from a relevant remote past to the present time;* **b:** *of, relating to, or characteristic of the present or the immediate past:* CONTEMPORARY; **2:** *involving recent techniques, methods, or ideas:* UP-TO-DATE; **3:** *of, relating to, or having the characteristics of the present or most recent period of development of a language;* synonym NEW.

The noun **modernism** means **1:** *a practice, usage, or expression peculiar to modern times;* **2:** *a tendency in theology to accommodate traditional religious teaching to contemporary thought and especially to devalue supernatural elements;* **3:** *modern artistic or literary philosophy and practice; a self-conscious break with the past and a search for new forms of expression.*

However, the modernism we are speaking about concerns also a specific era, beginning by the development of modern philosophy, exact sciences, and technology, by their prosperity, and approaching its end by the recognition that modern philosophy, exact sciences, and contemporary technology do not guarantee the survival of mankind, cognition and mastering of the universe, coexistence of cultures, equilibrium of the ecologic systems, etc. Some Middle European philosophers believe that the philosophy of Being, culture, sense, the sacred, individuality, new (postmodernistic) spirit, etc. are on the way of a harmonious coexistence of individuals, people, nations, ideologies, cultures, ecologic systems, etc. and mark the end of the traditional modernism and the beginning of postmodernism.

By the general definition of modernism, postmodernism is nothing else than a newer form of modernism, which roots in the previous form of modernism. For the newest modernism, i.e. postmodernism, characteristic philosophic, scientific, and technologic progressing (breakdowns of the previous „blindness„) is remarkable. For instance, the philosophy of Being seems to be adequate for new times, for it offers a solid ground of the care which must be considered in the field of science and technology and in the development of ecologic system to preserve the possibilities of coexistence of life-important circumstances on the Earth. To this philosophy, a new cultural spirit is necessary penetrating into the consciousness of individuals, populations, nations on one side, and into scientific and technologic awareness on the other side. This new spirit should regenerate all institutions of the society in a new sense and in the direction of harmonious coexistence and development of individuals, populations, nations, etc. A postmodern philosophy does not exist explicitly yet, at least there are not leading philosophers or philosophic schools visible at this time. However, Heidegger's philosophy of Being and Time seems to be already on the way to postmodernism, for instance.

Understanding of Information Society

The notion of information society concerns the philosophic-scientific-technologic chain of notions, i.e., mainly the historically arisen triplet matter-energy-information of human thought.

The evolution of philosophy yields no single concept of matter, but rather a large and still growing set of interrelated principles. The notion of *matter* comes from the Greek philosophy and dominates in philosophic and scientific comprehension until the eighteen century as a separate entity. Matter has traditionally been contrasted either with *form* or with *mind.* Matter, as understood by Aristotle, it the *passive* principle of things, whereas the form is an *active* principle which was mainly accepted also by scholasticism. Within the Renaissance philosophy and the work of Giordano Bruno, the dialectical comprehension of matter as an active principle is coming up again. Modern post-Cartesian thought has contrasted matter with mind.

The Greek *energeia* means activity, efficacy, effect, work. Energy is the ability of a body to perform a work. Energy means vigorous exertion of power, the capacity of acting or being active, dynamic quality. Energy can be *actual,* if it is in an acting state; or it can be *potential,* if it is in the state from which it can be activated (static energy). The movement, caused by energy of a body, can be, for instance, a movement of the entire mass (mechanical energy), movement of molecules (physical energy), movement of atoms (chemical energy), etc. Energy is constant and it can be transformed from one into another form. Energy

which cannot be transformed into another form (for instance, from lower to higher temperature) is called *entropy*. However, entropy is a general measure of the unavailable change or the disorder; it is also a measure of the amount of information in a message and, for instance, the degradation of the matter and energy in the universe to an ultimate state of inert uniformity. In informational sense, entropy concerns the steady degradation or disorganization of information, for instance, in a system or society.

The principle, important for understanding of matter and energy (their mutual transforming) is the so-called *relativity*. Relativity is a theory which is based on the two postulates: (1) that the speed of light in a vacuum is constant and independent of the source or observer and (2) that the mathematical forms of the laws of physics are invariant in all inertial systems, which leads to the assertion of the *equivalence of mass and energy* and of change in mass, dimension, and time with increased velocity (special theory of relativity).

In the triplet matter-energy-information there is a notional hierarchy from the left to the right. The priority of the matter in the triplet does not mean its a priori preference. The matter is at the origin, however as a substance, therefore as something that is at the bottom. Moreover, energy is on the second place, however concerning its importance, on a higher place as the matter. On the last, the highest place, is information. In short: the matter can be understood as energy and vice versa: the both can be understood as information. In general, matter and energy impact (inform) and are impacted (informed). Impacting of matter and energy as sensing, observing, understanding. etc. is information and informing of information. The material and energetic phenomenology can be comprehended as (in the form of) informational phenomenology of the universe, that is in an informationally consistent (natural, regular) way.

Understanding the term »information society« means to be aware of the importance of information in human societies, of information's principles which govern our biologic and cultural existence and arising. In information era, the human being comprehends definitely and cries for the first time explicitly: »All is information!« And slowly, but steady, this being becomes aware that informational nature of things changes previous and contemporary truth, belief, knowledge, and faith and let them arise, change and vanish as informational entities. In this sense, human being begins to understand that life and cultural forms depend and perform as information, irrespective of their nature. And this is the topics of the sense characteristic for the informational age. Thus, something of this basic modernistic recognition remains for the culture of postmodernism. It can hardly be neglected that philosophy, science and technology, or thought and action, are not informational or, for instance, that the concept of God is not a case of the most pretentious and universal information.

The Postmodernistic Sense and Information

Thus, we have suddenly arrived to the point where the question of the *sense* of postmodernism as information has to be brought into a more critical discourse. Where does the postmodernism surpass the modernism informationally? What is the sense of the arising postmodernism, its dominating Spirit? Can this Spirit be already explicated here and now?

Let us show a part of the kernel discussion concerning the notion of information demonstrated in (SEC).

What is the sense of modernism? In modernism, if we can believe, the so-called realm of culture becomes a means in the sense similar to the syntagma „to cultivate the nature„, that is to subordinate the nature by human power. The modern subject tends to master the inward and the outward nature. This striving is explicated as a direct domination of culture over the nature. The industrial society was an introduction to this domination; the postindustrial, i.e. information society should supplement and accomplish this domination. By means of information technology, the human being should become the master not only over itself, but also over the nature and society. As concluded correctly by the system theory, the information society is an autopoietic society.

However, this »is« only in the intention (purpose, design). In fact, the place of the Aristotelian God which is the thinking of thinking (idea of idea, form of form) is unattainable for this intention. Although it is understood by itself as Form, as a form of all forms, in-formation as such is without

spirit. Informational structures are unconscious, including the structures of the unconscious.

In the domain of the artificial (for instance, in artificial intelligence), the trials of treating belief, reasoning, and awareness (consciousness) with the intention to bring them into the conscious domain are a sort of symbolic and experimental simulation. It is to agree that these sort of simulation does not follow the principle of informational arising yet (POI). The philosophy and the theory of information are still in their infantile stage, however, they are getting more and more support from the neural sciences (PNS), where, for instance, information processing in brain as a consequence of different phenomenologies arises into a unique and promising scientific sense. And, willing or unwilling, the development and the reasonableness of this consciousness will influence the sufficiently perceivable philosophers to rethink the fundaments of the modernistic philosophy of Being in an informational, i.e., also postmodernistic way.

Certainly, it is possible to proceed from the assumption that the so-called spirit is not information and that information is without spirit. But this assumption assumes another concept of information—which is modernistic—and can root formally, for instance, in the Shannonian (mathematical) concept and various sociological models of communication. To say that spirit is not an informational phenomenon means to explain how it is non-informational by modernistic informational means. And information, in-formation, Form, formation, in-Formation, In-formation, In-Formation, etc. are only markers of one and the same phenomenology which is informational and understood in a postmodernistic way. In this postmodernistic way, God as information is a fact within a living information and He is not only the Aristotelian God (the thinking of thinking), but even much stronger idea (thought as information) of the Christian God, saying generally, in this particular case: information is everywhere and it understands everything. As the concept, as the idea, this kind of information is possible as it exists in somebody's mind.

What could be the so-called spirit as living phenomenology if it does not appear as an informational phenomenon of living brains or if it is not understood as informing of things? Information informs the spirit of things and the spirit is a consequence of things to be informed. Also, spirit as the thing (being) informs and is informed. This means that it performs as regular information in an information realm. Spirit as information arises informationally. It would be extremely interesting to know why the so-called spirit *could not be* an informational entity, but somewhat of that what is called thought, thinking, experiencing, comprehending, etc. as information in the living brain. In this way, human thought, thinking, thinking of thinking, etc. must not be informational processes of the living human brain.

The postmodernistic concept of information argues simply that things inform and are informed. The verb »to inform« becomes a metanotion for informing in thing-specific ways and can be replaced (particularized or universalized) adequately to the thing in question. Thus, the spirit of a thing is not necessarily its own informing, but informing influenced by several informational processes or entities at the place where the spirit is sensed, where it arises informationally and where it pervades certain information of individuals, population, nation, or given informational realm.

In the artificial domain of human activities, artificial information systems can demonstrate only the spirit of their constitution (design, construction). It is known that this systems are based on the interaction between human and a computer system and thus, can attain only the anticipated degree of intelligent emancipation. By definition, in fact, this intelligent (conscious and/or unconscious) emancipation is not intelligent in the sense of natural or human intelligence yet. Artificial information systems are simply tools like nail and hammer. However, this does not mean that it is not possible to study and understand information as an evident, even commonsense and rationalistic phenomenology within the natural, the living, and the artificial.

Conclusion

On the way to postmodernistic understanding, science and technology are not anymore traditionally single-minded, stupid, and into themselves closed fields of work and imagination. A clear evidence of this breakthrough is, for instance, the treatise of understanding of computers and cognition, by Winograd and Flores (UUC). Science of this sort is evidently already lost for the old,

modernistic era, for its way of thinking lies outside of the relevant (still dominating) contemporary science. The modernistic science and technology become more and more routine, strictly logical, automated, »mathematized«, computing work, when they ignore philosophic, intuitive, and life concerning impulses. In physics, one of the postmodernistic orientation is already the speculative field of *quantum theory* (mechanics, physics), which offers infinite possibilities of comprehension and processing of information. In informatics, the *redefined notion of information* is postmodernistic, opening the entirety of possibilities of understanding and processing of information in the living and in the artificial. The postmodernistic theory of information opens and fuses informational horizons, let them arise, change and disappear in a spontaneous and circular manner, according to the circumstances, needs, and possibilities. This theory remains also completely (ideologically) *tolerant* to the most pretentious, universal, mystic, unbelievable forms of information (God, the divine, the sacred, etc.) exploring them solely informationally, i.e., by its arising formal and natural languages and processes.

It is evident that desperate and more and more unsuccessful trials in the area of formal systems and computer programming still exist with the aim to retain a deficiently satisfactory validity of the modernistic notions in the field of artificial intelligence. On the other side, the forerunner of the new orientation are breaking through the slowly cooling down opposition against the new, informational concepts. If information age already belongs to the era of modernism, the so-called informational understanding of information (OWI, POI, IPF) can be classified as postmodernistic.

References

(SEC) Hribar, T., *The Slovene Emancipation* [»self-beingness«] *and Culture* (Science—Art—Education) (in Slovene), Nova revija **9** (1990) *95*, 544-565.

(PNS) Kandel, E.R. and J.H. Schwartz, *Principles of Neural Science*, Elsevier, New York (1985).

(UCC) Winograd, T. and F. Flores, *Understanding Computers and Cognition*, Ablex PC, Norwood, New Jersey (1986).

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

(POI) Železnikar, A.P., *Principles of Information*, Cybernetica **31** (1988) *2*, 99-122.

(IPF) Železnikar, A.P., *Informational Principles and Formalization* (in Slovene), Informatica **13** (1889) *3*, 21-40.

Graf — a Computer Company for Modernization of Media

Graf (Letališka 33, 61000 Ljubljana, phone (+38 61) 44 82 41, fax (+38 61) 44 82 74) is a young information system company specialized to meet the needs appearing within contemporary requirements of automatization of media, for instance, in large and middle-sized publishing companies, public and private television services, etc. The kind of modernization in these activities is based on new and particular computer technology and methodology. Graf conjoins groups of computer project designers, software engineers, communication specialists, experts in journalism, and economists experienced in yearlong engineering and implementation in news agencies and elsewhere. Graf offers conceptual projects of modernization, creation of the investment project and to it belonging activities, consulting, design of agencies for different kinds of media, publishing methodologies, education and training.

Graf's approach is on open system architecture, based on Unix and other operating systems, networking mini and microcomputer systems. This concept seems to be sufficient for automatizing small, medium, as well as large news agencies in Yugoslavia. The work is based on joint analysis of requirements and searching solutions together with the users. In this way the optimal choice of computer gear, software, peripherals, and communication can be achieved.

Further, Graf offers introduction to new electronic media such as videotex and interactive videotex, new services for the so-called subscriber service and advertising, introduction of CAD/CAM into graphical and engineering services of mass-media, modernization of advertising, resolving data problems, diskoteques, and TV services .

LIRA Jugoslovenska konferencija Logika i Računarstvo

Mesto: Dubrovnik, hotel Dubrovnik Palace
Vreme: 6—9, septembar 1990
Organizator: Matematički institut, Kneza Mihaila 35, 11000 Beograd
Kotizacija: 250 din do 1.6.1990, 500 din posle

Jugoslovenska konferencija **Logika i računarstvo** održaće se od 6. do 9. septembra 1990. u Dubrovniku u organizaciji Matematičkog instituta iz Beograda. Konferencija nastavlja tradiciju godišnjih seminara za konstruktivnu matematiku i teoriju modela Beograd—Zagreb koji se održava od 1975.

Tematika. Glavne oblasti koje će biti zastupljene na konferenciji: teorija modela, Bulove algebre, teorija skupova, neklasične logike, teorija dokaza, teorija odlučivosti i efektivne izračunljivosti, automatsko dokazivanje teorema, matematički aspekti sintakse i semantike prirodnih jezika, prepoznavanje oblika, ekspertni sistemi, logičko i funkcionalno programiranje, teorija programskih jezika, paralelizam i srodna područja.

Program rada konferencije. Predviđeno je gostovanje uglednih stručnjaka iz zemlje i inostranstva koji će održati jednosatna predavanja i konsultacije. Pored toga očekuje se i veći broj kraćih saopštenja učesnika Konferencije. Za potrebe demonstracije programa biće na razpolaganju PC-računari. Organizovaće se svečani ručak, a u slučaju interesovanja i izlet.

Važna napomena. Neposredno pre naše konferencije od 3. do 7. septembra 1990. održava se CAS (Center for Advanced Studies) International Summer Seminar on Artificial Intelligence. S obzirom na srodnost tema Seminara i Konferencije, koji se održava na istom mestu, svi prijavljeni za oba naučna skupa imaće određene povlastice.

A.P. Železnikar

Umrl je Željko Cener

V Ljubljani je 21. maja 1990 nepričakovano umrl naš stanovski tovariš, g. Željko Cener, dipl. ing. el., vodja prvega industrijskega projekta proizvodnje računalnikov v Iskri in Sloveniji. V začetku šestdesetih let je ta proizvodnja obsegala že tranzistorske sisteme z magnetnim bo' nom, tedaj vodilnega nemškega proizvajalca, podjetja Zu e KG. S tem je Slovenija dobila svojo prvo licenčno polindustrijsko proizvodnjo računalniških sistemov. Zbogom Željko!

Slava njegovemu spominu!

Slovensko računalništvo v kapitalizmu

Nobenega dvoma ni več, da je stara podjetniška struktura na področju računalništva v Sloveniji dokončno pokopana in da je računalniška industrija de facto izbrisana. Nastalo je na stotine majhnih podjetij, agencij in institutov, ki zaposlujejo le enega ali kvečjemu nekaj uslužbencev in razpolagajo domala izključno z zasebnim kapitalom. Vendar je pretežna usmeritev vseh teh podjetij predvsem preprodajna, domače tržišče pa je že zasičeno. Ponudba obsega prodajo HW in SW za področje PCjev in miniračunalnikov. V tej zvezi se seveda resno postavlja **vprašanje smisla** obstoja paradržavnih institucij na področju izobraževanja, znanosti in tehnologije.

Če bo usmerjenost trga delovne sile na področju računalništva v Sloveniji pretežno prodajna in neindustrijska, se bo moral tudi smisel izobraževanja in »znano-sti« transformirati predvsem na prodajno problematiko. Seveda pa bi bilo neracionalno trošiti sredstva slovenskih davkoplačevalcev za izobraževanje in znanost, ki ju doma ne bomo nikoli uporabljali. Čeprav je to vprašanje po svoje boleče, je vendarle očitno, da bomo drago izobraževanje in znanost morali—če ju bomo sploh še želeli trpeti na naših tleh—vzdrževati s tujim kapitalom in domala izključno z naročili iz tujine. Medtem, ko je takšen razvoj do neke mere še zamisljiv za znanost, ki se praktično lahko financira izključno z naročili iz tujine pa bodo morali bodoči slovenski študentje študirati sodobno tehnologijo izven Slovenije—na Hrvaškem ali v inozemstvu. Da ta usodna odločitev ne bo niti lahka niti enostavna, nam postaja jasno že danes.

Politične parlamentarne komisije in demokratični gremiji v Sloveniji se bodo tako morali kmalu soočiti z bolj splošnim problemom Slovenije, kako vzpodbujati ali zapirati heterogeno slovensko industrijo. Če se bo Slovenija odločala predvsem za čevljarsko, tekstilno in stolarsko ozirma lesno industrijo (izjave vodilnega ekonomista) brez elektronske industrije telekomunikacij, avtomatike in računalništva, da o robotiki ne govorimo niti v sanjah, bo nemudoma potrebno zapreti preostalo elektronsko industrijo v Sloveniji in vrsto oddelkov in zavodov na fakultetah obeh slovenskih univerz in institutih. V tem primeru bomo morali prihodnjo slovensko znanstveno in tehnološko populacijo že dovolj zgodaj usmerjati na študij in delo izven Slovenije. Pravi (dosledni) slovenski eksodus znanstvene in tehnološke inteligence je tako šele pred nami, saj pametnih Slovencev ne bo mogoče do nadalnjega več obdržati v domačih hlevih.

Trenutno politično vprašanje par excellence je tudi vprašanje odprtih natečajev za t.i. znanstveno in raziskovalno dejavnost, ki naj bi se demokratično odprla tudi svobodnemu trgu, saj čaka v novih podjetjih in institutih vrsta registriranih znanstvenih raziskovalcev tudi na znanstveno delo, brez katerega ni napredka zasebnih podjetij.

A.P. Železnikar

Prometej—računalniško zanesljivostno inženirstvo

Dr. Rihard Piskar, član mednarodnih strokovnih združenj za zanesljivost in kvaliteto (SaRS, IASTED in EOQC) in strokovnjak s številnimi domačimi in mednarodnimi referencami, je ustanovitelj družbe Ptolomej, d.o.o., Bertoncljeva 21, 64000 Kranj, tel. (064)22-943. Aktivnost družbe obsega:

—**zanesljivostno inženirstvo** [*analiza podatkov:* probabilistika, statistično zaupanje, testiranje hipotez, test hi-kvadrat, Fisherjev test, itd.; *interakcija moči (odpornosti) z obremenitvijo; prognoza in modeliranje:* dekompozicija, »cut and tie sets« markovski sistemi, simulacije Monte Carlo, MIL-HDBK-217, bellcore itd.; *vzdrževalnost in načrtovanje rezervnih delov:* logistika vzdrževanja, sistemi čakajočih vrst itd.; *zanesljivosr softwara:* modeli Shooman, Mills, Musa, Jelinski-Moranda, Schick-Wolverton, večrazsežnostni modeli itd.; *načini odpovedi in njih kritičnosti—FMECA:* varnostne ocene itd.; izboljšave zanesljivosti: analiza Pareto, test signifikantnosti itd.; *procedure »burn-in«, »run-in« akceleracije in ekstremne porazdelitve; analize testabilnosti in diagnozabilnosti:* sindrom, vodljivost, spoznavnost; *analiza vplivov okolice:* temperatura, vlaga, vibracije; *optimizacija stroškov življenjskega cikla];*

—**optimizacije v storitvah** [*analiza podatkov:* porazdelitve časov med zahtevki, statistična signifikantnost območij itd.; *optimizacija reševanja zahtevkov; analiza čakalnih časov in števila nerešenih zahtevkov; ocene učinkovitosti storitev*];

—**računalništvo** [*prodaja SW super-cub:* ocene učinkovitosti strežnih sistemov, ocene razpoložljivosti sistemov v eksploataciji, ocene povprečnih časov do zahtevkov itd.; *preskus performanc računalnikov za osebno rabo:* PC, osebni računalniki, nakupovalna odločitev, manjše tveganje; instalacija sistemskega in uporabniškega SW po želji kupca; razvoj uporabniških programov].

A.P. Železnikar

Informatica

Informatica *is published four times a year in Winter, Spring, Summer and Autumn by the* [illegible] *Ljubljana, Yugoslavia.*

Volume 14 Number 3 July 1990 YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

C O,N T E N T S

Voice Driven Editor	*Z. Kačič* *B. Horvat* *I. Urlep* *B. Štok*	1
Understanding as Information	*A. P. Železnikar*	8
Design of Hardware with Programmable Gate Arrays	*P. Praprotnik* *B. Dugonik*	31
Transputers for Embedded Real – Time Systems (in Slovene)	*P. Kolbezen*	35
Simulation and Performance Evaluation of Parallel Software of Multiprocessor System (in Slovene)	*B. Čukić*	44
Sphinx – A Computer System Recognizing the Continual Speech of Independent Speakers Using a Large Dictionary (in Serbo – Croatian)	*M. M. Miletić*	54
System Software for Process Control in Industry (in Slovene)	*D. Mrdaković*	56
The Development and Description of the Two – Level Model of Morphological *Analysis and Synthesis (in Slovene)*	*T. Erjavec*	59
On Knowledge in Distributed Systems (in Slovene)	*Tatjama Kapus*	63
Bayesian Artificial Neural Networks (in Slovene)	*I. Kononenko*	72
News (in Slovene and English)		87
Information and Postmodernism	*A. P. Železnikar*	89