Österreichische
Computer
Gesellschaft

NJSZT

Neumann János
Számítógéptudományi
Társaság

BEYOND NUMBER CRUNCHING

AUSTRIAN—HUNGARIAN CONFERENCE

AUSTRIA

September 14—16, 1988

Austrian
Computer Society

NJSZT

John von Neumann
Society for Computing
Sciences

BEYOND NUMBER CRUNCHING

AUSTRIAN—HUNGARIAN CONFERENCE

PROCEEDINGS

Editors: V. Haase, E. Knuth

Published by the John von Neumann Society
for Computing Sciences

Sponsored by:

Austrian Computer Society (OCG)
Federal Ministry of Science and Research, Austria
Forschungsgesellschaft Joanneum Ges.m.b.H.
Institute für Informationsverarbeitung Graz
Hungarian Academy of Sciences, Budapest
John von Neumonn Society for Computing Sciences (NJSZT)
Central Statistical Office, Hungary
Arbeitsgemeinschaft für Datenverarbeitung (ADV)
Steiermark

I S B N 963 8431 57 1

PREFACE

The Third Austrian-Hungarian Informatics-Conference and the first to be held in Austria took place at Schloß Retzhof near Graz. Like its predecessor at Sopron 1987 key researchers in the field of computer science from both countries were present. Nun numeric data processing issues were presented in 10 Austrian and 13 Hungarian papers in the field of cooperative and man-machine systems, graphics, databases and artificial intelligence.

We want to thank all supporting bodies and all individuals who have contributed to make this conference a success. A special "Dankeschön" goes to Mrs. Maria Toth for the organization of all Hungarian matters, including the printing of the proceedings, and to Ms. Anita Werner for keeping happy all participants as a conference secretary.

Graz/Budapest, September 1988

V. H. Haase
E. Knuth

MTA	=	HUNGARIAN ACADEMY OF SCIENCES
SZÁMALK	=	COMPUTING APPLICATIONS AND SERVICE COMPANY
ELTE	=	EÖTVÖS LORÁND UNIVERSITY
KFKI	=	CENTRAL RESEARCH INSTITUTE FOR PHYSICS, HUNGARIAN ACADEMY OF SCIENCES
BME	=	TECHNICAL UNIVERSITY OF BUDAPEST
SZKI	=	COMPUTER RESEARCH AND INNOVATION CENTRE
MTI	=	HUNGARIAN NEWS AGENCY

CONTENTS

Graphics and Databases

Artificial Intelligence

BEYOND NUMBER CRUNCHING - SPECIAL TOPICS

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 9-13.

INFORMATION-BASED AND COMPUTATIONAL COMPLEXITY OF PHYSICALLY REALIZABLE MACHINES

A. CSURGAY

Hungarian Academy of Sciences
Budapest, Roosevelt tér 9, H-1051

Abstract. In our review computation is considered as a physical process, defined by a Hamiltonian (hardware) and initial conditions (software). Information-based and computational complexities are introduced. It is shown that bounded propagation speed in planar digital machines does limit the computational power of parallelism. In the universe of binary strings the asymptotic limits of sequential deterministic Turing-machines can not be overcome. Probable mismatch between measurability and computability of physical quantities is exposed.

Key-words. Computation, Complexity, Turing machines, Parallel computing, Computability

I. PHYSICS AND COMPUTATION

Science has a history of synthetizing many phenomena into a few and elegant theories. Heat and sound were explained by the laws of motion; electricity,magnetism and light by the laws of electromagnetism. Quantum mechanics supplied the laws behind the whole chemistry, and by explaining the interaction of photons and elementary particles a synthetizing theory of quantum electrodynamics (QED) has emerged. At this stage all phenomena of the physical world can be explained by three laws: QED, gravitation and nuclear physics. Recent attempts to synthetize these laws into a "grand unified theory" are also promising.

The observed complexity of nature can be explained by addig adaquate algorithms to the laws, which starting from an initial state and applying the laws would calculate, thus predict the measured data. We use machines to execute the algorithms.

Complexity should be grasped by our machines. Bounding N bits of information increases the negative of the thermodynamic entropy ("negentropy") by at least N.kT. In every moment of time the information already captured is represented by spatial negentropy. The algorithm changes this negentropy in time.

In our machines there is an unchangable time-invariant structure, defining a Hamiltonian, carved into the lattice, called hardware, and an easily changable dynamics of photons and electrons is superimposed on it, the initial condition of which is defined by the programs, called software. All machines, digital, analog or hybrid, are defined by a Hamiltonian, which can be programmed by implementing an initial state on it.

Computation is a physical process, obeying the laws of physics. However, the laws should be formulated in terms of computable algorithms, which depend on the laws. We face a mutual determination.

II. THE UNIVERSE OF BINARY STRINGS

Mapping of a binary string into another one, i.e. a program written in a formal language, can be represented by a binary string as well. Thus recursive functions, even formal languages and Turing-machines can be considered as elements of the set of binary strings.

Thus binary strings form a "universe", in which all observations, and theorems are binary strings. In case of a computer one string defines another when it is a program for the computer to calculate the second string.

Information-based complexity of a binary string is defined to be the shortest program that makes the computer to output the string. Any string of length n can be calculated by putting it directly into a program as a table, thus the complexity is less or equal to the length of a string. Kolmogorov (1965) and Chaitin (1966) proposed to call random those strings of length n whose complexity is appoximately n.

Solomonoff and Chaitin used the notion of information-based complexity to formulate the situation that a researcher faces when he has made observations and wishes to understand them and make predictions. He searches for a theory. We consider his observations to be represented by a binary string, and the theory to be a program that ouputs the string. Scientists consider the simplest theory to be the best one, and that if a theory is too "ad hoc", it is useless. The simpler the theory, the shorter the program, thus the information grasped by the string can be squeezed into a smaller space.

The relevance of this notion to information processing, storage and retrieval is obvious: if a binary sequence can be described by a short program, then this means that the information content of it can be "squeezed" into small space. Information-based complexity has a spatial character.

If we try to recover the compressed sequence we have to be able to carry out the necessary computations. Information-based complexity sketched above does not say anything about the time necessary to compute the sequence from its generating program. In information processing a different kind of complexity, the so-called computational one is widely applied to describe the number of steps necessary to run a program, to perform an algorithm.

Let us pose the problem of the algorithmic calculation of the shortest program that will generate a given series, i.e. the problem of finding the information-based complexity. It has been shown that the computational complexity of this problem is undecidable, i.e. there is no such a solution of this problem which would terminate in finite steps.

Nevertheless, the universe of binary strings is an extremely rich universe. The Turing-Church Thesis tells us that Turing-machines, formal languages and recursive functions are isomorph sets.

Turing-machines are sequential. This has an impact on the computational complexity of problems solved on them.

III. FULL PARALLELISM IS EQUIVALENT TO TURING MACHINES

There were some hopes that by full parallelism, i.e. by machines in which arbitrary number of bits can be processed at any instant, the asymptotic blow-up of a number of important algorithms could be overcome. This would mean that problems of NP-hard computational complexity could be solved by polinomial algorithms.

We have looked at a model, introduced by Chazelle and Monier, which exploit the possibility of unbounded parallelism while trying to remain realistic. (Most of the old models contradict basic laws of physics by making the assumption that the transmission of information is instantaneous.)

It is a model for planar, digital computing devices, and the propagation speed of information is bounded by a constant.

In this model

- the information is digital (binary) and encoded by the value of a physical parameter at specified times and locations;
- a circuit computes a boolean function

  $(Y_1, Y_2, \ldots, Y_m) = F(x_1, x_2, \ldots, x_n)$
- the size of a problem is the total number of input and output bits;
- a circuit is a planar layout of a directed graph, where the nodes are finite-state-automata (FSA) and the edges are wires. The inputs and outputs of the FSAs are boolean values stored at the endpoints of the wires, and we can assume long wires to be decomposed into unit-length segments connected by nodes computing the identity function. This allows us to associate each wire with exactly one variable, and thus assume that it has unit bandwith;
- communicating information with the outside of the circuit takes place at special nodes called I/O ports and located on the boundary of the circiut;
- both the area A and the time of computation T have quantized units, usually denoted by λ and τ. A node performs an operation in at least unit time τ, and it has an area at least λ^2;
- wires have width at least λ, and they transmit information at bounded speed.

The model is a physical parallel model, since an arbitrary number of bits is processed at any instant. It can be considered to describe any planar, digital, physical machine.

It has been shown that any model described above, solving a problem of size N in time T and area A can be simulated on a two-dimensional Turing mechine in sequential time $T_s = O(NT^{A})$, using the same area, and vica versa, any deterministic Turing machine which computes a function in time T with a tape of length L can be simulated on our model-circuit of area O(L) in time O(T).

Thus the high parallelism does not change the class of computational complexity. In physical machines there exist a relation between the time of computation and the area that can be active during this time. From an asymptotic point of view any physically realizable digital machine is polynomially equivalent to sequential machines, e.g. to deterministic Turing machines.

As a consequence not only the uncomputable (e.g. halting) problems bat also the NP-hard problems remain intractable even with the use of an unbounded amount of digital hardware.

IV. COMPUTABILITY AND MEASURABILITY

In the binary universe there are well-posed mathematical problems, e.g. the halting problem, or the set of noncomputable numbers, which cannot be solved by digital machines, i.e. there is no algorithm which would solve the problem by a terminating program. Noncomputability depends neither on computers nor on languages. Computability is an attribute of the number itself, and not how that number is presented, and noncomputable numbers are dense in the reals, because the cardinality of the set of programs is equal to the set of integers, which is smaller than the cardinality of the set of real numbers.

In physics each real number is considered to be "measurable, at least in principle. Is it a prejudice that every measurable quantity should be computable, or we have to accept that the universe of binary strings is not rich enough to reflect the observed complexity of nature, i.e. there are measurable numbers noncomputable.

In case of noncomputability two cases should be distinguished: (i) we do not know the algorithmic solution of a problem as yet, or (ii) we can prove that no solution does exist in the universe of binary strings, like in case of the halting problem.

Examples have been presented in which physical quantities can be measured with given accuray, on the other hand they cannot be computed with given accuracy.

It is suggested that universal machines covering broader sets then digital machines should be looked for. Others suggest that the binary universe is rich enough, and the complexity can be grasped by inventing new models of evolution reflecting the self-organization of nature. Active research is going on in both directions.

REFERENCES

1 Special Issue of Foundations of Physics, Vol. 16, No 6, 1986

R.P.Feynman, Quantum Mechanical Computers, pp. 507-531;

R.Geroch, J.B.Hartle, Computability and Physical Theories, pp. 533-550;

R.Landaer, Computation and Physics, pp. 551-564;

C.H.Bennett, On the Nature and Origin of Complexity, pp. 585-592.

2 G.J.Chaitin, Information-Theoretic Computational Complexity, IEEE Trans. on Information Theory, IT-20, No.1, January, 1974

3 B.Chazelle, L.Monier, A Model of Computation for VLSI with Related Complexity Results, 13th ACM Symp.on Theory of Computing, ACM, May, 1981.

4 C.H.Bennett, Notes on the history of reversible computation, IBM J.,Res.Develop. Vol. 32, No.1, January, 1988

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics. Conf. p. 15-25.

Algorithms in Polynomial Ideal Theory and Geometry *)

Franz Winkler

Institut für Mathematik and
Research Institute for Symbolic Computation
Johannes Kepler Universität, A-4040 Linz, Austria

Abstract

Many geometric problems can be formulated as problems about polynomials. By investigating the polynomial ideals associated with geometric problems one can often arrive at powerful decision algorithms. Algorithms for solving problems in the theory of polynomials and polynomial ideals, such as the Gröbner basis method and the cylindrical algebraic decomposition method, are described and applied to a variety of specific geometric problems.

The goal of this paper is to show how powerful algorithms for dealing with polynomial equations and inequations can be used to solve a variety of geometric problems. The paper is structured according to specific problems, and the theory is developed as far as it is necessary to solve those problems.

Problem 1

Suppose we are given a finite number of polynomial equations

$$f_1(x_1,\ldots,x_n) = 0, \ldots, f_m(x_1,\ldots,x_n) = 0 \tag{1}$$

over some ground field K and we are interested in the solutions of this system. The ring of polynomials in the indeterminates $x_1,\ldots,x_n$ over K is denoted by $K[x_1,\ldots,x_n]$. We observe that whenever $\bar{x} = (\bar{x}_1,\ldots,\bar{x}_n)$ is a solution of this system and the new polynomial f is a linear combination of $f_1,\ldots,f_m$, i.e. $f = \sum_{i=1}^{m} h_i f_i$ for some $h_i \in K[x_1,\ldots,x_n]$, then $\bar{x}$ is also a solution of $f = 0$. So the solutions of the system (1) are really the solutions of all polynomial equations $f = 0$, where f is a linear combination of $f_1,\ldots,f_m$. The set of all such linear combinations forms an ideal in $K[x_1,\ldots,x_n]$ and we denote it by ideal$(f_1,\ldots,f_m)$, the *ideal generated by* $f_1,\ldots,f_m$. The polynomials $f_1,\ldots,f_m$ are called a *basis* of this ideal. On the other hand, by Hilbert's Basis Theorem, every ideal in $K[x_1,\ldots,x_n]$ has a finite basis.

The first problem that we consider is a fundamental problem in the theory of polynomial ideals, namely the *ideal membership problem*.

Problem 1:
given: $f, f_1,\ldots,f_m \in K[x_1,\ldots,x_n]$,
decide: $f \in$ ideal$(f_1,\ldots,f_m)$.

For solving Problem 1 we have to introduce some notation first. If u is a power product, i.e. u is of the form $x_1^{i_1}\cdots x_n^{i_n}$ for some nonnegative integers $i_1,\ldots,i_n$, then the *degree of* u

*) Work reported herein has been supported by the *Österreichische Forschungsgemeinschaft* and by the *Fonds zur Förderung der wissenschaftlichen Forschung*, Projekt Nr. P6763.

in x_j is defined as $\deg_{x_j}(u) = i_j$, and the *degree of* u is defined as $\deg(u) = \sum_{j=1}^{n} \deg_{x_j}(u)$. Let $\prec$ be a linear ordering of the power products in the indeterminates $x_1, \ldots, x_n$ which makes $1 = x_1^0 \cdots x_n^0$ the least power product and is compatible with multiplication, i.e. if $w_1 \prec w_2$, then for every power product u we have $u \cdot w_1 \prec u \cdot w_2$. Such an ordering $\prec$ is called a *term ordering*. Examples of term orderings are the lexicographic ordering $\prec_l$

$$u \prec_l u' \iff \text{exists } k, 1 \leq k \leq n, \text{ such that}$$
$$\deg_{x_r}(u) = \deg_{x_r}(u') \text{ for all } r < k \text{ and } \deg_{x_k}(u) < \deg_{x_k}(u'),$$

or the graduated lexicographic ordering $\prec_g$

$$u \prec_g u' \iff \deg(u) < \deg(u') \text{ or } (\deg(u) = \deg(u') \text{ and } u \prec_l u').$$

From now on let $\prec$ be some chosen term ordering.

Every nonzero polynomial f can be uniquely decomposed into its *leading term* $\mathrm{lt}(f)$ (consisting of a *leading coefficient* $\mathrm{lc}(f)$ and a *leading power product* $\mathrm{lpp}(f)$) and its *reductum* $\mathrm{red}(f)$, where

$$f = \mathrm{lc}(f)\mathrm{lpp}(f) + \mathrm{red}(f) \tag{2}$$

and $\mathrm{lc}(f) \neq 0$ and $\mathrm{lpp}(f)$ is greater (w.r.t. $\prec$) than any power product occurring in $\mathrm{red}(f)$.

Every set of polynomials $F \subseteq K[x_1, \ldots, x_n]$ induces a *reduction relation* $\longrightarrow_F$ on $K[x_1, \ldots, x_n]$ in the following way: $g_1 \longrightarrow_F g_2$ iff $g_2 = g_1 - \frac{a}{\mathrm{lc}(f)} \cdot u \cdot f$ for some $f \in F$ and u a power product such that $u \cdot \mathrm{lpp}(f)$ occurs in g_1 with coefficient a. In words: g_1 *is reducible to* g_2 *modulo* F. If no such u and f exist, then g_1 is *irreducible modulo* F. This reduction relation is Noetherian, i.e. every chain $f_1 \longrightarrow_F f_2 \longrightarrow_F \cdots$ terminates. By $\longrightarrow_F^*$ we denote the reflexive, transitive closure of $\longrightarrow_F$, i.e. $g \longrightarrow_F^* h$ if and only if g is reducible to h modulo F in finitely many steps. If $f \longrightarrow_F^* g$ and g is irreducible modulo F, then g is a *normal form of* f *modulo* F.

It is quite obvious that whenever $f_1 \longrightarrow_F f_2$ then $f_1 - f_2$ is a linear combination of the polynomials in F. Therefore, if $f \longrightarrow_F^* 0$ then $f \in \mathrm{ideal}(F)$. However, the reverse implication does not hold in general. Fortunately one can always transform a given basis F of a polynomial ideal I into a so called *Gröbner basis* F' of I, which is characterised by the property that

$$f \in I \iff f \longrightarrow_{F'}^* 0. \tag{3}$$

An existence proof and also an algorithm for transforming any finite basis into a Gröbner basis are given in [Buchberger 65], [Buchberger 85]. If the elements of a Gröbner basis are reduced with respect to each other, we get a *minimal reduced* Gröbner basis.

<u>Theorem 1</u>: *Let $\prec$ be a term ordering. For every ideal $I \subseteq K[x_1, \ldots, x_n]$ there exists a Gröbner basis F of I. The minimal reduced Gröbner basis of I is uniquely determined.*

Now let us return to Problem 1. In order to determine whether $f \in \mathrm{ideal}(F)$, where $F = \{f_1, \ldots, f_m\}$, we compute a Gröbner basis G for $I = \mathrm{ideal}(F)$ and reduce f to a normal form f' modulo G, i.e. $f \longrightarrow_G^* f'$ and f' is irreducible modulo G. Then $f \in \mathrm{ideal}(F) \iff f' = 0$.

As an example we consider the ideal I generated by $F = \{f_1, f_2, f_3\}$ in $\mathbb{Q}[x, y, z]$, where $f_1 = xz - xy^2 - 4x^2 - \frac{1}{4}$, $f_2 = y^2 z + 2x + \frac{1}{2}$, $f_3 = x^2 z + y^2 + \frac{1}{2}x$. We want to determine whether the polynomial $f = 702y^2 - 64x^5 + 2788x^3 - 348x^2 + 395x - 10$ is contained in the ideal generated by F. As the term ordering we choose the lexicographic ordering with $x \prec y \prec z$. The minimal reduced Gröbner basis for I is $G = \{g_1, g_2, g_3\}$, where

$$g_1 = z + \frac{64}{65}x^4 - \frac{432}{65}x^3 + \frac{168}{65}x^2 - \frac{354}{65}x + \frac{8}{5},$$
$$g_2 = y^2 - \frac{8}{13}x^4 + \frac{54}{13}x^3 - \frac{8}{13}x^2 + \frac{17}{26}x,$$
$$g_3 = x^5 - \frac{27}{4}x^4 + 2x^3 - \frac{21}{16}x^2 + x + \frac{5}{32}.$$

$f \longrightarrow^*_G 0$, so $f \in I$. In fact, $f = 702 g_2 - 64 g_3$. Observe that f is irreducible modulo F.

Problem 2

From a geometric point of view, we associate with every polynomial ideal I the set of points $V(I) \subseteq K^n$ at which all the polynomials of I vanish. $V(I)$ is called an (affine) *algebraic variety*, the *variety of* I. On the other hand, starting from an algebraic variety V we consider the set $A(V)$ of all polynomials J which vanish on V. Clearly $I \subseteq A(V(I))$, but in general the inclusion is proper. In fact $A(V(I))$ is the *radical* of I, radical(I), i.e. the set of all those polynomials f for which some power f^n is in I.

So geometrically, we are more interested in the *radical membership problem* $f \in$ radical(I) than in the ideal membership problem.

<u>Problem 2</u>:
given: $f, f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$,
decide: $f \in$ radical$(f_1, \ldots, f_m)$.

By adapting Rabinowitsch's method of proving Hilbert's Nullstellensatz, the following theorem can be proved [Buchberger 85].

<u>Theorem 2</u>: *Let* $f, f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$. $f \in$ radical$(f_1, \ldots, f_m)$ *if and only if* $1 \in$ ideal$(f_1, \ldots, f_m, z \cdot f - 1)$, *where z is a new variable.*

In order to decide whether $f \in$ radical(F), where $F = \{f_1, \ldots, f_m\}$, we compute a Gröbner basis G for $I =$ ideal$(f_1, \ldots, f_m, z \cdot f - 1)$ and check whether $1 \in I$. So the radical membership problem is reduced to the ideal membership problem, which we have already shown to be solvable. In fact, if G is the minimal reduced Gröbner basis for I, then testing whether $1 \in I$ amounts to testing whether $1 \in G$.

Problem 3

For given polynomials $f_1, \ldots, f_m$ we consider the homogeneous linear equation

$$f_1 z_1 + \cdots + f_m z_m = 0 \tag{4}$$

in the unknowns $z_1, \ldots, z_m$. We are interested in the solutions of this equation in $K[x_1, \ldots, x_n]^m$. Every solution $(g_1, \ldots, g_m) \in K[x_1, \ldots, x_n]^m$ is called a *syzygy* of $(f_1, \ldots, f_m)$. The set of all syzygies of a given sequence of polynomials forms a submodule of $K[x_1, \ldots, x_n]^m$ over $K[x_1, \ldots, x_n]$. Every such submodule has a finite basis.

<u>Problem 3</u>:
given: $f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$,
find: *a basis for the module of syzygies of* $(f_1, \ldots, f_m)$.

Again, the Gröbner bases method can be used to solve this problem. First a Gröbner basis $G = \{g_1, \ldots, g_k\}$ is computed for $I =$ ideal(F), where $F = \{f_1, \ldots, f_m\}$. For G it is very easy to construct a basis for the syzygies, see [Buchberger 85], [Winkler 86]. In a second step the basis of the syzygies of G is transformed to a basis of the syzygies of F by the following theorem.

Theorem 3: *Let* $F = (f_1, \ldots, f_m)^T$ and $G = (g_1, \ldots, g_k)^T$ *be two column vectors over* $K[x_1, \ldots, x_n]$. *Let* X, Y *be transformation matrices over* $K[x_1, \ldots, x_n]$ *such that* $G = X \cdot F$ *and* $F = Y \cdot G$. *Let the rows of the matrix* R *be a basis of the module of syzygies of* G. *Then the rows of the matrix*

$$Q = \begin{pmatrix} I_m - F \cdot Y \\ \cdots\cdots\cdots\cdots \\ R \cdot X \end{pmatrix}$$

are a basis of the module of syzygies of F.

Problem 4

For a (finite) number of algebraic equations

$$f_1(x_1, \ldots, x_n) = 0, \ \ldots \ , f_m(x_1, \ldots, x_n) = 0 \tag{5}$$

we want to determine the common solutions of this system of equations. Usually one wants to find these solutions in the algebraic closure $\bar{K}$ of the ground field K.

Problem 4:
given: $f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$,
find: the set of common solutions of $f_1 = \cdots = f_m = 0$ *over* $\bar{K}$.

Before we set out to generate the solutions to this problem, we might want to know whether the system of algebraic equations $f_1 = \cdots = f_m = 0$ has any solutions at all. This question can easily be answered once we have computed a Gröbner basis for the given polynomials.

Theorem 4: *Let* $F = \{f_1, \ldots, f_m\} \subseteq K[x_1, \ldots, x_n]$ *and* G *the minimal reduced Gröbner basis for* ideal(F). *Then the system of equations (5) is unsolvable (in* $\bar{K}$*) if and only if* $1 \in G$.

Now suppose that (5) is solvable. We might want to determine whether there are finitely or infinitely many solutions.

Theorem 5: *Let F and G be as in Theorem 4. Then (5) has finitely many solutions if and only if for every i, $1 \leq i \leq n$, there is a polynomial g_i in G such that $lpp(g_i)$ is a power of x_i.*

For really carrying out the elimination process, we compute the Gröbner basis with respect to the lexicographic ordering. The following elimination property of Gröbner bases has been observed in [Trinks 78]. It means that the i-th elimination ideal of ideal(G) is generated by the polynomials in G that depend only on the variables $x_1, \ldots, x_i$.

Theorem 6: *Let G be a Gröbner basis w.r.t. the lexicographic ordering $x_1 \prec x_2 \prec \cdots \prec x_n$. Then*

$$\text{ideal}(G) \cap K[x_1, \ldots, x_i] = \text{ideal}(G \cap K[x_1, \ldots, x_i]) \quad \text{for } 1 \leq i \leq n,$$

where the ideal on the right hand side is formed in $K[x_1, \ldots, x_i]$.

As an example let us consider the same polynomials as in the example to Problem 1. The minimal reduced Gröbner basis G does not contain 1, so by Theorem 4 the system of equations $f_1 = f_2 = f_3 = 0$ is solvable in the algebraic closure of Q. Furthermore, by Theorem 5, this system of equations has finitely many solutions. The variables in the Gröbner basis G are totally separated. An approximation of a root of g_3 up to ± 0.000001 is -0.128475. This solution of $g_3 = 0$ can be continued to solutions of $g_2 = 0$ and $g_1 = 0$, yielding the approximation $(-0.128475, 0.321145, -2.356718)$ for the original system of equations.

Problem 5

A plane algebraic curve is the variety of a single bivariate polynomial [Walker 78]. We want to compute the intersection of two plane algebraic curves.

<u>Problem 5</u>:

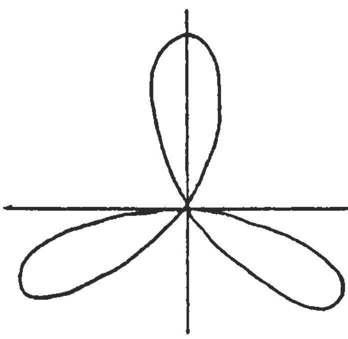
given: *bivariate polynomials f_1, f_2 specifying two plane algebraic curves C_1, C_2,*

find: *intersection of the two curves C_1, C_2.*

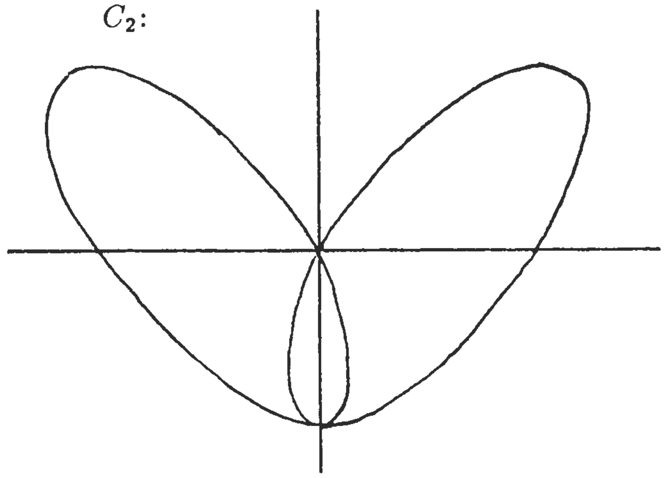
The points of intersection of the two curves C_1, C_2 are exactly the points satisfying the system of equations $f_1 = 0$, $f_2 = 0$. So Problem 5 is reduced to problem 4.

As an example let us consider the two curves

C_1:



C_2:



$f_1 = x^4 + 2x^2y^2 + y^4 + 3x^2y - y^3$

$f_2 = y^4 + 2y^3 + y^2 - 3x^2y + 2x^4 - 3x^2$

We compute the Gröbner basis G of $\{f_1, f_2\}$ w.r.t. to the lexicographic ordering, getting the basis polynomials

$$g_1 = x^{12} + \frac{1003}{81}x^{10} + \frac{3724}{81}x^8 - \frac{250}{3}x^6 + \frac{2500}{81}x^4,$$

$$g_2 = x^4y - \frac{137214}{17689565}x^{10} - \frac{1173797}{17689565}x^8 - \frac{9239511}{17689565}x^6 + \frac{3202095}{3537913}x^4,$$

$$g_3 = y^2 + 3x^2y - \frac{20439297}{141516520}x^{10} - \frac{253511161}{141516520}x^8 - \frac{464023179}{70758260}x^6 + \frac{162950991}{141516520}x^4 - 3x^2.$$

So the intersection variety of the two curves consists of finitely many points. The x-coordinates of the intersection points are the solutions of the univariate equation $g_1 = 0$. This equation has 9 different solutions, 4 of which are complex. One of the solutions of $g_1 = 0$ is $\bar{x} = -\frac{5}{9}\sqrt{2}$. Substituting $\bar{x}$ into $g_2 = 0$ and solving for y yields $\bar{y} = -\frac{5}{9}$.

Problem 6

Many properties of an algebraic curve depend on the number and position of its singular points. A singular point is a point of multiplicity higher than one.

Problem 6:

given: *a bivariate polynomial f specifying a plane algebraic curve C,*
find: *the singular points of the curve C.*

The singular points of C are exactly those points, at which f and the partial derivatives of f vanish. So the singular points are the solutions of the system

$$f(x,y) \underset{\wedge}{=} \frac{\partial f}{\partial x}(x,y) = \frac{\partial f}{\partial y}(x,y) \underset{\wedge}{=} 0.$$

Thus the detection of singularities is reduced to the solution of a system of algebraic equations.

As an example we consider the tacnode C_3 (this is the curve C_2 from Problem 5, shifted by 1 along the y-axis), specified by the equation $f(x,y) = 2x^4 - 3x^2y + y^2 - 2y^3 + y^4 = 0$. For computing the singular points of C_3 we have to solve the system

$$\begin{aligned} f(x,y) &= 2x^4 - 3x^2y + y^2 - 2y^3 + y^4 = 0, \\ \frac{\partial f}{\partial x}(x,y) &= 8x^3 - 6xy = 0, \\ \frac{\partial f}{\partial y}(x,y) &= -3x^2 + 2y - 6y^2 + 4y^3 = 0. \end{aligned}$$

A Gröbner basis of this system w.r.t. the lexicographic ordering ($x \prec y$) is

$$\{x^3,\ xy,\ y^2 - y + \frac{3}{2}x^2\}.$$

So C_3 has two singular points with coordinates $(0,0)$ and $(0,1)$.

Problem 7

Although an algebraic curve in general is given as the variety of a polynomial, for some applications – for instance drawing the curve on a screen – it is desirable to describe the curve by a suitable parametrization. Rational parametrizations have been investigated extensively in algebraic geometry. An irreducible curve C, i.e. a curve given by an irreducible polynomial $f(x,y) = 0$, is *rational* iff there exist rational functions $\phi(\lambda), \psi(\lambda)$ of a parameter λ such that (i) for all but a finite set of values λ_0 for the parameter $(\phi(\lambda_0), \psi(\lambda_0))$ is a point of C, and (2) with a finite number of exceptions for every point (x_0, y_0) of C there is a unique value λ_0 of the parameter such that $x_0 = \phi(\lambda_0), y_0 = \psi(\lambda_0)$.

Problem 7:

given: *an irreducible polynomial $f(x,y)$ describing a rational algebraic curve C,*
find: *a rational parametrization $\phi(\lambda), \psi(\lambda)$ of C.*

Let $f(x,y)$ be an irreducible polynomial of degree d describing an algebraic curve C. The *genus* of the curve C is defined as

$$g_C = \frac{(d-1)(d-2)}{2} - \frac{1}{2}\sum r_i(r_i - 1),$$

where the summation is over all points P_i of the curve C and r_i is the multiplicity of the point P_i. (In general, also “infinitely near” points have to be taken into consideration, see [Walker 1978].) The genus is a measure of how much the curve is deficient from its maximum allowable limit of singularities (the genus can never be less than 0). An algebraic plane curve C is rational if and only if $g_C = 0$.

The simplest case in parametrizing a rational curve occurs when the curve C, described by the irreducible polynomial $f(x,y)$ of degree d, has a $(d-1)$-fold point P. W.l.o.g. P can be assumed to be at the origin (otherwise a linear transformation is performed). In this case a line $y = \lambda x$ through P intersects C in exactly one additional point Q, yielding a parametrization of the curve.

As an example we consider the curve C_1 of Problem 5. C_1 is a curve of degree 4 having a triple point at the origin. The line $y = \lambda x$ intersects C_1 at the origin and at the additional point

$$Q = \left(\frac{\lambda^3 - 3\lambda}{\lambda^4 + 2\lambda^2 + 1}, \frac{\lambda^4 - 3\lambda^2}{\lambda^4 + 2\lambda^2 + 1}\right).$$

So

$$x = \frac{\lambda^3 - 3\lambda}{\lambda^4 + 2\lambda^2 + 1},$$
$$y = \frac{\lambda^4 - 3\lambda^2}{\lambda^4 + 2\lambda^2 + 1}$$

is a parametrization of C_1.

A quartic curve of genus 0 can either have a triple point or 3 double points. In the second case one chooses an additional simple point on the curve and passes conics through these 4 points of the curve. There will be exactly 1 free parameter in the equation of the conic, and by Bezout's theorem there will be exactly 1 additional point of intersection. So this additional point of intersection is uniquely determined by the free parameter in the equation of the conic, thereby leading to a parametrization of the quartic curve. This idea can be generalized, see [Walker 1978], [Abhyankar,Bajaj 87].

Problem 8

The inverse problem is to take the parametric equations for an algebraic plane curve C and turn them into an algebraic equation defining this curve.

<u>Problem 8:</u>

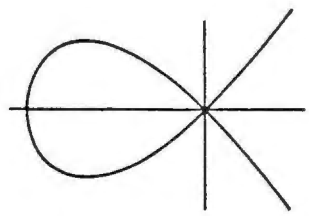
given: $x = \frac{p_1(t)}{q_1(t)}, y = \frac{p_2(t)}{q_2(t)}$,
a rational parametrization of an algebraic plane curve C,

find: *a polynomial $f(x,y)$ describing the curve C.*

This problem has been solved in [Arnon,Sederberg 85]. It requires to find the algebraic relationship between the variables x and y representing the coordinates of the points in the parametric version of the curve C, given the algebraic relationships $x \cdot q_1(t) - p_1(t) = 0, y \cdot q_2(t) - p_2(t) = 0$. So we have to compute a basis for $ideal(x \cdot q_1(t) - p_1(t), y \cdot q_2(t) - p_2(t)) \cap K[x,y]$. According to Theorem 6, this can be achieved by a Gröbner basis computation.

As an example we consider the curve C_1 given parametrically by the equations

$$x = t^2 - 1, y = t(t^2 - 1).$$



Starting from the relations $f_1 : t^2 - x - 1 = 0, f_2 : t^3 - t - y = 0$, we want to compute the relation between x and y. A Gröbner basis for the ideal generated by f_1 and f_2 with respect to the lexicographic ordering $x < y < t$ is

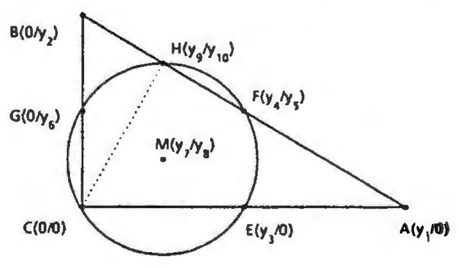
$$\{t^2 - x - 1,\ xt - y,\ yt - x^2 - x,\ y^2 - x^3 - x^2\}.$$

The intersection with $\mathbb{Q}[x,y]$ yields $y^2 = x^3 + x^2$, which is indeed the polynomial describing the curve C_1.

Problem 9

Often a geometric statement can be described by polynomial equations over some ground field K. As an example we consider a special case of the Apollonios Circle Theorem [Kutzler,Stifter 86]:

The altitude pedal of the hypothenuse of a right-angled triangle and the mid-points of the three sides of the triangle lie on a circle.



A possible algebraic formulation of this problem is

$$(\forall y_1,\dots,y_{10})[h_1 = \cdots = h_8 = 0 \Longrightarrow c = 0],$$

where

$h_1 = 2y_3 - y_1 = 0$ (E is midpoint of CA),

$h_2 = 2y_4 - y_1 = 0,\ h_3 = 2y_5 - y_2 = 0$ (F is midpoint of AB),

$h_4 = 2y_6 - y_2 = 0$ (G is midpoint of BC),

$h_5 = (y_7 - y_3)^2 + y_8^2 - (y_7 - y_4)^2 - (y_8 - y_5)^2 = 0$ (length EM = length FM),

$h_6 = (y_7 - y_3)^2 + y_8^2 - (y_8 - y_6)^2 - y_7^2 = 0$ (length EM = length GM),

$h_7 = (y_9 - y_1)y_2 + y_1 y_{10} = 0$ (H lies on AB),

$h_8 = -y_1 y_9 + y_2 y_{10} = 0$ (CH perpendicular to AB),

and

$c = (y_7 - y_3)^2 + y_8^2 - (y_7 - y_9)^2 - (y_8 - y_{10})^2 = 0$ (length EM = length HM).

The polynomials $h_1, \ldots, h_8$ are called the *hypotheses* and c is called the *conclusion* of the geometric statement.

In order to prove the theorem, it suffices to show that c vanishes on all the common roots of $h_1,\ldots,h_8$, i.e. c is in the radical of $h_1,\ldots,h_8 \in \mathbb{C}(y_1,y_2)[y_3,\ldots,y_{10}]$. So this problem is reduced to Problem 2, which can be solved by a Gröbner basis computation. c is indeed in the radical of the hypothesis polynomials (in fact, it is in the ideal generated by the hypothesis polynomials), so the theorem is proven.

Usually a geometric theorem is true only after certain degenerate situations have been ruled out by a *nondegeneracy* or *subsidiary condition*. As for the hypotheses and the conclusion, we require that the subsidiary condition be expressible by a polynomial, this time by a polynomial inequation of the form $s(y_1,\ldots,y_n) \neq 0$. Of course the subsidiary condition should not be so strong as to exclude all cases of the geometric construction. So the problem becomes the following:

<u>Problem 9</u>:

given: $h_1,\ldots,h_m, c \in K[x_1,\ldots,x_n]$,

decide: *does there exist a polynomial s such that*

$(\forall x \in \bar{K}^n)(h_1(x) = \cdots = h_m(x) = 0 \wedge s(x) \neq 0 \Longrightarrow c(x) = 0)$

and

$(\exists x \in \bar{K}^n)(h_1(x) = \cdots = h_m(x) = 0 \wedge s(x) \neq 0)$?

If so, find such an s.

Solutions for this geometry theorem proving problem are described in [Wu 84], [Chou,Schelter 85], [Kutzler,Stifter 86], [Kapur 86], [Winkler 88]. The principal problem with all these solutions to the geometry theorem proving problem is that they can only deal with geometries over algebraically closed ground fields, e.g. complex geometry and not real geometry.

Problem 10

Finally let us turn to the question of deciding problems in real algebraic geometry, where we do not only allow the predicates $=$ and $\neq$, but also comparisons $<$ and $\leq$. This leads to the *elementary theory of real closed fields*, see [van der Waerden 71]. Models for the elementary theory of real closed fields are the field of real numbers R and the field of real algebraic numbers.

The elementary theory of real closed fields is decidable. Decision algorithms have been given in [Tarski 51], [Seidenberg 54], and [Collins 75]. As an example of the *cylindrical algebraic decomposition* method due to Collins we consider the formula

$$\phi \equiv (\exists x)(\exists y)(x^2 + y^2 - 4 < 0 \wedge y^2 - 2x + 2 < 0).$$

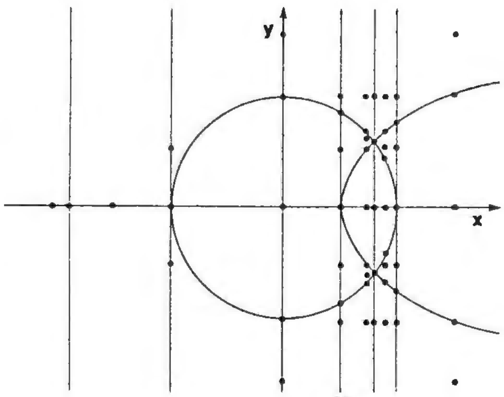
The goal is to decide whether ϕ holds over the reals. In order to arrive at such a decision, the plane is decomposed into connected regions, in which the signs of all the polynomials

$$A = \{y^2 + x^2 - 4,\ y^2 - 2x + 2\}$$

occurring in the formula are constant. In a projection phase the variable y is eliminated by computing resultants and discriminants of the polynomials in A, leading to

$$A' = \{x^2 + 2x - 6,\ x^2 - 4,\ x - 1\}.$$

The roots of the polynomials in A' give a decomposition of the x-axis into connected regions on which all the polynomials of A' have constant signs. So in order to test the signs of the polynomials in A' in any of these regions, a single sample point is sufficient. Now the decomposition of $\mathbb{R}^1$ is lifted to a decomposition of $\mathbb{R}^2$. The polynomials in A are evaluated at the sample points and the roots of the resulting univariate polynomials are computed. From this information sample points for the regions in $\mathbb{R}^2$ can be computed, such that all the polynomials in A have constant signs in every region. In our example sample points, e.g. $(\sqrt{7}-1,0)$, are found which satisfy all the conditions in ϕ. Thus ϕ is valid.



References:

[Abhyankar,Bajaj 87] S.S. Abhyankar, C. Bajaj: *Automatic Parameterization of Rational Curves and Surfaces I: Conics and Conicoids*, Techn. Rep., Comp. Sci. Dept., Purdue Univ. (1987)

———: *Automatic Parameterization of Rational Curves and Surfaces II: Cubics and Cubicoids*, Techn. Rep., Comp. Sci. Dept., Purdue Univ. (1987)

———: *Automatic Parameterization of Rational Curves and Surfaces III: Algebraic Plane Curves*, Techn. Rep., Comp. Sci. Dept., Purdue Univ. (1987)

[Arnon,Sederberg 85] D.S. Arnon, T.W. Sederberg: *Implicit Equation for a Parametric Surface by Gröbner Basis*, manuscript (1985)

[Buchberger 65] B. Buchberger: *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, Dissertation, Univ. Innsbruck (1965)

[Buchberger 85] B. Buchberger: "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory", in: *Multidimensional Systems Theory*, N.K. Bose (ed.), Reidel (1985)

[Chou,Schelter 85] S.-C. Chou, W.F. Schelter: "Proving Geometry Theorems with Rewrite Rules", *J. Automated Reasoning* 2, 253-273 (1985)

[Collins 75] G.E. Collins: "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition", *Automata Theory and Formal Languages (2nd GI Conference)*, LNCS 33, 134-183, Springer-Verlag (1975)

[Kapur 86] D. Kapur: "Geometry Theorem Proving Using Hilbert's Nullstellensatz", *Proc. 1986 Symp. on Symbolic and Algebraic Computation*, 202-208, ACM (1986)

[Kutzler,Stifter 86] B. Kutzler, S. Stifter: "Automated Geometry Theorem Proving Using Buchberger's Algorithm", *Proc. 1986 Symp. on Symbolic and Algebraic Computation*, 209-214, AGM (1986)

[Seidenberg 54] A. Seidenberg: "A New Decision Method for Elementary Algebra", *Ann. of Math.* 60/2, 365–374 (1954)

[Tarski 51] A. Tarski: *A Decision Method for Elementary Algebra and Geometry* 2nd ed., Univ. of California Press (1951)

[Trinks 78] W. Trinks: "Über B. Buchbergers Verfahren, Systeme algebraischer Gleichungen zu lösen", *J. of Number Theory* 10/4, 475–488 (1978)

[van der Waerden 71] B.L. van der Waerden; *Algebra I*, Springer-Verlag (1971)

[Walker 78] R.J. Walker: *Algebraic Curves*, Springer-Verlag (1978)

[Winkler 86] F. Winkler: *Solution of Equations I: Polynomial Ideals and Gröbner Bases*, Lecture Notes, Conf. on "Computers & Mathematics", Short Course "Symbolic and Algebraic Computation", Stanford Univ. (1986)

[Winkler 88] F. Winkler: "A Geometrical Decision Algorithm Based on the Gröbner Bases Algorithm", *Proc. 1988 Interm. Symp. on Symbolic and Algebraic Computation*, Rome, LNCS, Springer-Verlag (1988)

[Wu 84] Wu Wen-tsün: "Basic Principles of Mechanical Theorem-Proving in Elementary Geometry", *J. Syst. Sci. & Math. Sci.* 4/3, 207–235 (1984)

COOPERATIVE SYSTEMS

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 29-43.

HYPER-COSTOC: A COMPUTER-BASED TEACHING SUPPORT SYSTEM

Friedrich Huber

Institute for Information Processing Graz (IIG) of the
Graz University of Technology and the Austrian Computer Society
Schiesstattgasse 4a, A-8010 Graz
(e-mail: fhuber@tugiig.UUCP)

ABSTRACT:

We propose the main architectural features of HYPER-COSTOC (HC), a new computer based teaching support system, parts of which are currently being developed. We have called it HYPER-COSTOC because it is based on a database of lessons called COSTOC and because it has facilities which have a hypermedia flavor. HC is an advanced CAI system which can retrieve and process, in a non-linear fashion, structured molecules of instructional or information material networked together. HC uses a mixture of presentation-type-CAI, object-oriented programming, and query language techniques for university teaching.

HC aims at a practical and viable approach within a university setting. It comprises tools for the creation, distribution, and usage of a variety of courseware in a fashion which is modular, technology independent, easy to maintain, and which includes drill, self-test, exam and other modules. HC should be geared to easy integration with a variety of instructional environments: as a support to traditional lecture-style instruction as well as for the development of exploratory research environments.

This paper is a shortened version of a paper by F. Huber, H. Maurer, and F. Makedón to appear in Journal of Microcomputer Applications.

1. INTRODUCTION

COSTOC is an international project which involves the production and implementation of computer science courseware within university environment [MMO]. Up to now, the word COSTOC has been an acronym for COmputer Supported Teaching Of Computer-Science. Since the techniques inherent to COSTOC go beyond the domain of computer science, we have now expanded this acronym to stand for: COmputer Supported Teaching? Of Course!

The COSTOC system is currently being used in about 20 educational institutions. COSTOC includes a database of hundreds of one-hour lessons, with more than 250 topics within computer science. Starting from this courseware basis, HC should add many new facilities, including better navigation, browsing, window cross-talk as well as aspects of student modelling. It should be made clear, however, that the first implementation of HC will put little emphasis on intelligent-CAI techniques (e.g., intelligent tutoring, intelligent student modelling) because we believe that further developments will be necessary before ICAI becomes broadly applicable.

The important contribution of the ongoing COSTOC project is a massive and growing database of high-quality lessons, mostly in computer science, created by experts around the world. Lesson material consists of text and graphic segments, combined with a variety of different types of frames (help, reference, question-answer, etc). For reasons of portability and maintainability, the COSTOC lessons do usually not include programs or other media, although they have the capability to incorporate such. By the end of 1987, some 250 computer science lessons were available, and this number is expected to grow as high as 500 by 1989. Several labs are currently running in the US with the COSTOC database incorporated in the curriculum. Typical labs consist of a fileserver (a PC with a large hard disk, or a Micro-Vax), containing the database of lessons. Student stations are attached directly to the fileserver or via LAN or other networks.

The main differences between COSTOC and HC can be summarized as follows:

(a) Radical improvements of the authoring facilities
(b) Radical improvements of the delivery system
(c) Incorporation of the ability for simulation and program inclusion within a lesson

2. HYPER-COSTOC: A GENERAL VIEW

HC is a CAI system which provides a spectrum of "hypermedia type" facilities and tools to a lesson database with a choice for online and offline use. The philosophy of HC is both hardware and media independent. It is especially geared to a distributed, microcomputer-based university environment which integrates a variety of university resources in an intelligent way and which is accessible in different modes to a network of off and on campus users.

2.1. A Typical HC Lab Configuration

A typical HC lab consists of a number of student/user workstations which are connected via a network to a database of instructional material. In the browsing mode, the workstation stays permanently online. In the studying mode, a substantial package of instructional material is "downloaded" into the user's workstation and executed locally, i.e., without requiring connection to the database of lessons during this phase. The extent of browsing mode as compared with studying mode depends on the type of configuration used:

(a) The browsing mode should be favored when workstations are connected to a database server via a network with no time charges. The same holds in the case that the workstations lack sufficient local processing power.

(b) On the other hand, the studying mode should be favored in the following cases: If intelligent workstations are connected to a database via a network with time charges, the studying mode should be emphasized in order to minimize connect time charges. Also, slow networks (below 9600 baud) tend to favor this mode and material that is "largely sequential". Another case for the S-mode is that a database server of a fixed size can clearly handle more workstations, as is the case with the Austria-wide network handling 9,000 users [M3] and the CONNEX lab described in [CLM].

The HC execution environment is designed to handle both modes. It is also planned to run on a variety of computers ranging from dedicated micros called MUPID to PC's and workstations like Apollos. The most widely used workstation for HC labs will probably be the PC with an ega card.

2.4. HC Database: A Network of Hypermols

The HC database consists of a large number of usually small teaching units which represent 10 seconds to a few minutes of studying time. We call these units "hyper-molecules" or "hypermols" for short, since they constitute the building blocks of the HC database. Hypermols have the following properties: they are the smallest modules that a user can directly access, they are largely independent of each other, and the user can switch from one hypermol to another hypermol at any time. We distinguish among three types of basic hypermols:

(a) Presentation-type hypermols (PT-hypermols):
They present textual and graphical information which includes dynamic changes of the screen as caused by user-input and also simple animation sequences.

(b) <u>General hypermols (G-hypermols)</u>:
A G-hypermol is a type of teaching unit which can include parts of PT-hypermols and also have other functions defined with it. For example, a G-hypermol function may define a question-answer dialogue between system and user for the purpose of self-assessment or for suggesting on how to continue, or for examination purposes. Another G-hypermol function may define an algorithm which is specifiable within the framework of GLSS [MS2] used for simulation or experimentation purposes.

(c) <u>External hypermols (E-hypermols)</u>:
To keep HC open-ended, which means to be able to incorporate arbitrary software packages, we permit the formation of external hypermols (E-hypermols). An E-hypermol is an arbitrary software package, possibly external to the HC database, which is made available to the user. This software package may be an application program, a driver for digitized pictures, voice, sound or video etc. This flexibility provides the "hypermedia" quality of the HC system.

2.4.1. Composition of Hypermols and Navigation

Although hypermols can be accessed individually, they are usually loosely tied together as a <u>lesson</u>, and groups of lessons compose a <u>course</u>. Thus, a lesson is a network of hypermols , with one node defined as the starting one. Paths which visit the connected hypermols within any given lesson are not part of the hypermols, but kept separate to make a data base of mols highly reusable. Such hypermol routing mechanisms are called <u>H-tours</u>.

An H-tour constitutes a directed graph. Each node in this graph corresponds to a certain hypermol. At the end of this hypermol the student is shown all accessible choices. A choice corresponds to an edge in the directed graph and leads to a new node and the attached hypermol. There are standard types of branches, like <u>sequence</u> (next - back) and <u>index</u> (several choices - back).

All information necessary for navigation is kept in the nodes of H-tours. They allow to build highly individualized ways through the data-base without duplicating mols. Usually each hypermol will be referenced by several H-tours. Figure 2.1 shows a part of two H-tours. H-tour number 1 starts at hypermol A and continues in a straight sequence with B, C, and D. H-tour number 2 also starts at hypermol A. At mol B, however, the student can continue with three different mols: one sequence starting with K, L, etc., a second one with C, D, ... and a third one emanating from R. To keep the figure simple only forward links are shown. As indicated in Figure 2.1 mols C, L, R, and S are also contained in some other H-tours.

There are two types of H-tours (routing through hypermols within a lesson): Static H-tours and dynamic H-tours which depend on the outcome of question-answer dialogues. The trade-off is clear here.

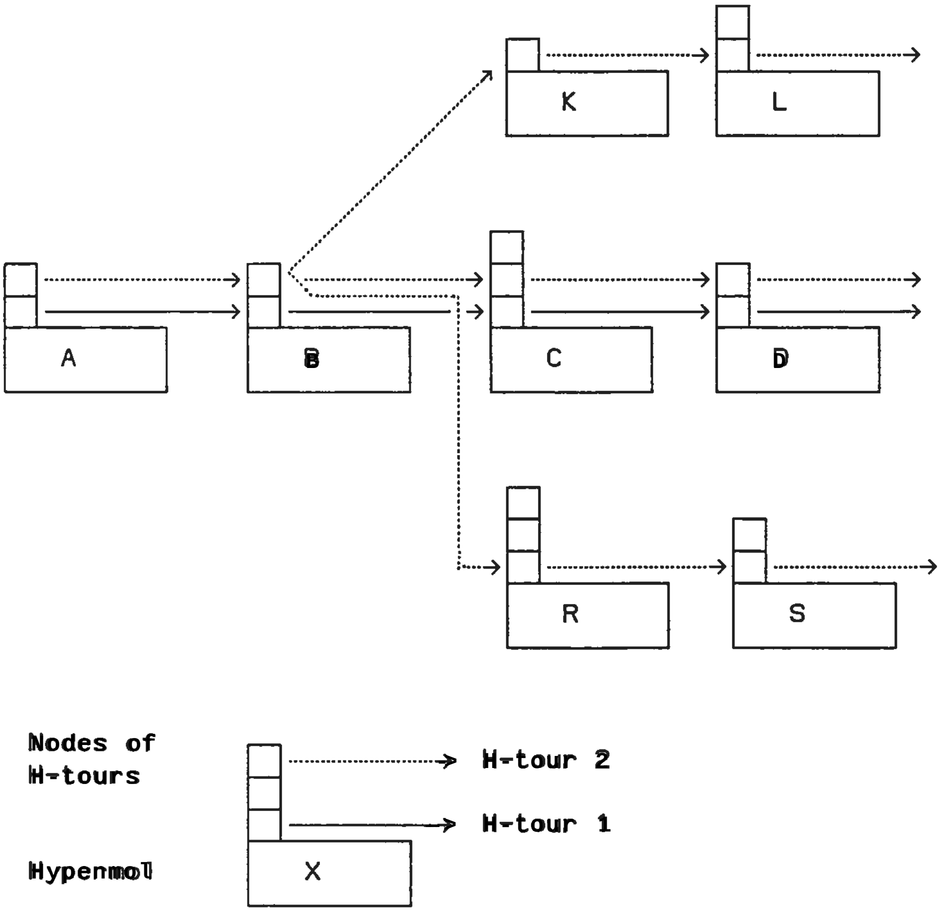


Figure 2.1

2.4.2. Hypermol Attributes

Each hypermol has two types of information associated with it: identification information (ID) and annotation information (AN). The ID and AN information is explained in terms of examples below.

The ID of a hypermol is used to locate the information desired and consists of a number of attributes with associated values. For example, a hypermol of a sorting lesson may have the following set of attributes (with their values indicated in parentheses): molid (sort314), course (sorting), lesson# (3), lesson-name (heapsort), molnumber (14), author (maurer), supervisor (garfield), language (english), domain (computer science), area (data_structures, algorithms), keywords (heap, heap_definition), attribute-type (PT), etc.

2.4.3. The HC Query Mechanism: Some Query Examples

The attributes of mols can be used to search for certain mols or lessons. The search should be carried out in interactive fashion, i.e., the user identifies what kind of items he/she is looking for (mols, lessons, courses, ...) and defines the key for the search. Some examples show how this feature can be used.

In the above example of hypermol with molid(sort314), a user can find this hypermol by specifying

hypermols (keyword = heap_definition).

The result of such a query will be a list of hypermols since some lessons on data structures will also contain the definition of a heap. Moreover, by using a wild card character like $, the user can abbreviate names of keywords. Specifying

hypermols ((lesson-name = heapsort) <u>and</u> (keyword = $heap$))

will give a list of all hypermols of the lesson heapsort where the keyword contains the string heap, such as "heap", "definition of a heap", "heap_definition" or "heapify". The names of all lessons on sorting could be determined by specifying

lessons ((course = sorting) <u>and</u> (lesson-name = $)).

As further example, consider a user working through a lesson on syntax-analysis and requiring the definition of finite automaton as defined in a lesson by, say, Salomaa. A query

hypermols ((author = Salomaa) <u>and</u> (keyword = Finite Automat$))

might well lead to the definition wanted. Observe, finally, that

hypermols ((keyword = Salomaa) <u>and</u> (attribute-type = Facsimile))

could give a facsimile-picture of Salomaa, or

hypermols ((keyword = $automat$) <u>and</u> (attribute-type = PR))

a list of computer programs dealing with automata (for experimentation).

More examples can be found in [HMM]. Since the data-base contains all the necessary information on keywords, attribute-types etc., the query system can be written in such a way that the user only has to select from menus and to enter text merely in some rare cases.

2.5. The HC Annotation Facility

The annotation facility serves three main purposes:

(1) it allows to add notes to each hypermol at different operational levels: as a systems operator, as an instructor, and as a user.

(2) it allows to specify new private H-tours which override suggested lesson-embedded tours and is hence called "active anno-

tation" [MMR].

(3) it provides a communication mechanism between the instructor and the author of the instructional material.

In addition to annotations associated with individual hypermols, there is also a messaging facility of the usual kind. For example, students may send messages to their instructor, and the instructor can operate a bulletin board for one of the classes.

2.6. Cross-process Communication

Another important feature of HC which we call <u>cross-process communication</u> is best explained in a multi-window setting and by running through a few examples.

The main point concerning cross-process communication is that in a multi-window environment a number of hypermols can be active on the screen simultaneously. Activation of a new hypermol (and suspension of the currently active one) is either carried out by the hypermol itself or by user-intervention. As an arbitrary hypermol x is activated from a PT- or G-hypermol y, a command-file belonging to this activation is executed: this execution puts hypermol x into a certain state. A return from hypermol x to the initiating hypermol y after the point of activation can either be caused by a command file, by the termination of hypermol x or by a command by the user.

Let us now study two examples.

EXAMPLE 1:

Let us assume we have 3 PT-hypermols A, B, C, and that each of them consists of some textual and graphical information which is split up into three pieces. For instance, A is of the form A = Ax, P, Aa, P, As indicating that first part Ax is shown, after a key press (P) part Aa is shown, and after a further key press As. Assume that B and C are built up analogously. Suppose now we want to show to the user A, B and C in three different windows in four stages as follows: first Ax,Bx,Cx; then Aa,Ba,Cx; then Aa,Ba,Ca; then Aa,B3,C3; the stages are separated by a key press.

In a truly parallel fashion the definition of these sequences is fairly easy: the author selects all three hypermols and executes them, showing the editor where synchronization points should be established. To a student the three hypermols now appear as if additional pauses had been inserted:

Ax,	Px,	Aa,	P=,	Aa,	P3,		P4
Bx,	Px,	Ba,	P=/		Pa,	Ba,	P4
Cx,	Px,		P2,	Ca,	Pa,	Ca,	P4

EXAMPLE 2:

Different hypermols can also be shown step by step on the screen under direct user control. Consider once more the 3 hypermols A, B, C above and suppose the user wants to see simultaneously first A_1, B_1, C_1 and then A_3, B_3, C_3. The user activates A in one window giving A_1, then activates B in another window and presses a key twice to obtain B_3, then activates C in a third window. Now A_1, B_3, C_1 are visible. By reactivating A and B and pressing a key twice for each mol, the second desired configuration is obtained.

2.7. Checking of Inputs - Filters

Furthermore, observe that certain moduls such as checking the validity of inputs keep re-occuring in many simulation - or experimentation programs. To reduce the work of designing such programs the authoring system should support the generation of G-hypermols checking the input. These G-hypermols can then be used by other G-hypermols, passing the necessary parameters as command file.

In passing we also want to point out that the mechanism necessary to define a valid set of answers, and the algorithms for checking whether a given answer is in the set (as they are used in the answer-judging routines of the execution environment) are also useful for the construction of filters. Thus, many of the algorithms for determining spelling errors, redundant words, using synonyms etc. have applicability far beyond answer-judging: filters are just one example, general user inputs are others: when a user types

lesson-name (course = Data-structures),

and the system comes back with no lessons since the course happens to be named Datastructures (without hyphen) then the system is clearly not sufficiently user-friendly!

3. HYPER-COSTOC FROM A STUDENT'S POINT OF VIEW

3.1. Generally Available Execution Features

A HC session usually starts by identifying oneself and selecting a lesson from the HC data-base via menu pages and/or a simple database query mechanism. The identification is necessary for several reasons: to save the status for later execution when temporarily leaving the system, for exams, to manage private annotations, etc.

In studying mode as many mols as possible are loaded into the memory of the workstation. To determine the hypermols needed the information kept in the H-tour emanating from the title hypermol is used. Lessons come with routing suggestions leading from hypermol to hypermol, as stated in the H-tour by the original author. Whenever desired, execution can be continued with arbitrary hypermols, (potentially in new windows and allowing to just temporarily suspend the execution of the current hypermol), hypermols possibly including question/exam modules, simulation- and experimentation programs, the use of other media, or execution of some other program-package.

In addition to the navigational features discussed so far, further features available including marking a hypermol, returning to a marked hypermol, backstepping (repeatedly if desired) through the dynamically last hypermols, continuing with the hypermol containing the (dynamically last) (sub)table of contents and switching on/off a fast display mode in which a hypermol is displayed in accelerated fashion.

Above features are fairly easy to implement in most environments. However, there is one navigational feature of paramount importance which is not easy to handle fully and hence is often not supported. It is the undo feature, undoing the effect of the last key press in PT-hypermols and when following H-tours: the difficulty in implementing repeated undos in an environment of dynamically changing graphic information is that the only trivial implementation - keeping a stack of memory maps and systems' status - is usually too memory intensive to be useful. More tricky implementations are possible and discussed in [MR].

In addition to be able to collect a mixture of parts of hypermols and own notes just for printing, the information can also be retained in an electronic version as notebook. Indeed usually all information is first put in a notebook (which can later be inspected and edited), and the printing is just one of the options of handling the notebook.

3.2. Annotations

Finally, let us consider the concept of annotations, an extremely important concept of the execution environment.

When the user starts a session and accesses a first hypermol, a list of public annotations is shown. Any of these public annotations and further private annotations (for which the user must know the appropriate password, e.g. because these annotations were created by the same user in an earlier session) can be enabled (and later disabled, if desired). To enable an annotation means that whenever a hypermol is activated, all enabled annotations are shown in parallel with the hypermol in a separate window. If the hypermol causes dynamic changes on the screen step by step, the annotations can also be split up into steps and synchronized with the hypermol in a way described in more detail in [MMR].

Annotations mainly consist of text. They can also create special pointers outside the text window (i.e. in one of the graphic windows) for highlighting. Thus, an annotation may read "Observe how the valve * opens and closes", with the * appearing both in the text and next to the graphic object being explained. How this is done, and how conflicts between various annotations are avoided is explained in [MMR].

Public annotations are usually written by an expert commenting or elaborating the work of the original author, or by a teacher as individualized information for a particular class ("... a further good book might be ...", "... this proof won't be on the exams a workbook.

Annotations are a way to allow a certain individualization and customization of instructional material. This possibility is much enhanced by one further aspect of annotations which we have not particularly considered yet. It is this aspect which causes annotations in HC to be called active annotations, to differentiate them from the static annotations found in many document systems.

Annotations may contain routing suggestions of the kind "To continue press 1" where the solicited action leads to any hypermol selected by the annotator, in particular to hypermols created by other authors. In connection with an automatic annotation activation and return facility, hypermols can be linked together in entirely new ways both by the teacher and the user.

A final word is appropriate concerning private annotations; to assure privacy and to conserve storage in the data-base, private annotations are usually kept on the student's directory or even on floppies and not in the data-base of lessons itself. The user, on returning to material studied before, is then shown his personal annotations either from the private directory or from the floppy.

4. HYPER-COSTOC FROM AN AUTHOR'S POINT OF VIEW

It is of crucial importance for any CAI undertaking how easy it is to create and maintain lesson material. We have discussed at length (in [MM2]) that this is one of the main reasons supporting PT-CAI and, more general, CAI data-bases containing small pieces of instructional material (be it PT-CAI, experimentation or simulation software) which are as independent of each other as possible and hence can hopefully be designed and tested more or less as independent modules.

The hypermol concept supports this philosophy. PT-hypermols are usually small and quite "context independent" modules; the same holds true for G-hypermols except that certain G-hypermols may already become substantial in size and hence are harder to create, debug and maintain. The E-hypermols, on the other hand, are supposed to be developed outside the HC framework. Their design can be made easier, to some extent, by shifting some of the effort to standard G-hypermols whose generation is supported by HC authoring as will be discussed below. IT is, for example, possible to use standard input-hypermols to remove the task of designing interfaces for requesting (and validity checking of) student inputs from E-hypermols.

4.1. HC Editors

Basically, HC authoring provides a <u>presentation facility editor</u> (for creating PT-hypermols or presentation segments of G-hypermols), a <u>question-answer dialogue editor</u>, a <u>routing editor</u>, a <u>standard hypermol editor</u> and a <u>program editor</u> (allowing to create and test program segments written in a subset of Pascal as specified in GLSS [MS2]). As important feature, the program editor allows to use presentation-type constructs as developed by the presentation facility editor. Detailed descriptions of the various editors can be found in [HMM]. We just describe here the important aspects of the different editors.

The HC authoring system which we propose should be <u>extensible</u> and <u>customizable</u>. Let us consider some examples to get a rough idea of what this means. The crucial aspect of extensibility is that new objects can be introduced (including the specification of all interfaces required by the author, such as menus for input prompts and calculation or other procedures where required). Once introduced, such new objects behave exactly like standard objects: After an object "perpendicular bisector", for example, has been introduced and is then used by the author, a prompt for the input of the endpoints of a line segment appears. After the endpoints have been defined by the author, the line segment and its perpendicular bisector are drawn automatically. A more complex extension would be to introduce "and-gates" and "or-gates" with inputs and outputs as new objects in such a fashion that, when drawing such gates and their connection lines, the system always automatically shows the author how many inputs and outputs still have to be dealt with.

Customizing and extending the editor also involves actions such as renaming or deleting objects (to simplify menus or to make them more appropriate for the task at hand; typically, the term "vector" is more appropriate than "arrow" in mathematical contexts, but not in others) or pre-setting certain parameters such as color combinations, defining the layout of a table-of-contents-hypermol to be used as kind of template, etc. Consult [HM1], [HM2], and [H] for further details and possibilities.

Question-answer dialogues come in a number of varieties, namely <u>multiple choice</u>, <u>free-text</u>, <u>form-filling</u>, <u>drill</u>, <u>exercise</u> and <u>exam</u>.

In case of the drill a view option (which can be enabled by the author) can be used to first look at all questions and the corresponding answers before the actual drill starts. Essentially drill forces the learner to answer all questions repeatedly, until they can be sure, that they know all the answers. There exist different strategies for choosing questions from the set of all possible questions, from simple picking at random to more sophisticated methods, like in [MS4].

In the exercise mode, questions from the set chosen are presented only once in a random order with feedback showing the correct solution if the user has not found it after a number of tries. At the end, the percentage of questions answered is shown to the user for self-assessment.

The exam mode is similar to the exercise mode except that a time limit specified by the author is given, and correct answers are only shown after the exam is finished. Also, there is an identification procedure at the beginning, and results are recorded in a file only accessible to the instructor (graphical presentation of points obtained, cut-off points for grades, printed listing of students and grades).

The routing-editor shows the author a graphic presentation of the network the mols of a lesson form. It allows to link hypermols, suggesting to users which out of a selection of hypermols to view next. Some of the routes suggested (or even enforced) depend on the outcome of question-answer dialogues.

Message, notebook and calculation facilities are available to both authors and students. The facility to collect information from hypermols, to edit it, to add personal notes and finally print it, is particularly valuable for authors as a means of easily producing a printed course-documentation. HC authoring also supports the extraction and replacement of text-pieces from hypermols for translation purposes, full-text searches and string replacements accross the range of lessons and other text-related functions.

5. HYPER-COSTOC FROM AN INSTRUCTOR'S POINT OF VIEW

As first step an instructor has to select material of interest for the specific situation in the HC data-base. This is usually done on the basis of the printed documentation of the courses.

Once a course looks promising, the instructor checks through it to determine which parts to use. Often, an instructor decides to use a few lessons for replacing some class-room teaching and some further just for "recommended reading". Usually the instructor will go through the material once, making annotations for the students ("... learn up to here for the midterm ....", "also look at the book lessons more widely usable.

Finally, instructors want to know how often hypermols have been accessed. For this purpose, HC presents comprehensive statistical information on the use of the data-base. As a matter of fact, in addition to the "statistical" package (just presenting tables of what has been used when) a sophisticated "monitoring" package should also be available to record all keypresses of students (in an anonymous way). Such data, together with special utilities, will provide good insight into how the data-base is used, which parts of questions cause problems, etc.

6. SUMMARY

HC is a major undertaking whose aim is to provide the design and part of the tools to set up a comprehensive CAI lab to support teaching activities at various levels. In addition, a large database of instructional material is available in some key areas (e.g. computer science) in good quality from the outset.

Acknowledgement:
Support of part of this research by the Austrian Federal Ministry of Science and Research (Grant Zl. 604.508/2-26/86) and the Fonds zur Förderung der wissenschaftlichen Forschung (Grant P6042P) is gratefully acknowledged.

REFERENCES

[BC] Bonar, J. and R. Cunningham, "Bridge: An Intelligent Tutor for Thinking about Programming", Learning Research and Development Center Technical Report, 1986.

[BCS] Bonar, J., Cunningham, R., and J. Schultz, "An Object-oriented Architecture for Intelligent Tutoring Systems", OOPSLA'86 Proceedings, SIGPLAN Notices, Vol. 21, No. 11, pp. 269-276.

[BB] Burton, R., and Brown, J., "An Investigation of Computer Coaching for Informal Learning Activity", Appears in Intelligent Tutoring Systems, ed. by Sleeman, Brown, Academic Press.

[C1] Carbonell, J., "AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction", IEEE Transactions on Man-Machine Systems, Vol. MMS-11, No. 4, Dec. 1970.

[CLM] Cheng, H., Lipp, P. and Maurer, H., "GASC: A low-cost, no-nonsense Graphic and Software Communication System", Electr. Pub. Review 5, 1985, pp. 141-155.

[C3] Conklin, J., "Hypertext: an Introduction and Survey", IEEE Computer, Sept. 87, pp. 17-41.

[DS] Delisle, N., and Schwartz, M., "Context - A Partitioning Concept for Hypertext", Conf. on Computer Supported Cooperative Work Proc., Austin, Tx., Dec. 1986, pp. 147-152.

[GM] Garratt, J., Maurer, H., "Autool-2 Manual for COSTOC Authors", IIG Report 244, Graz Univ. of Technology, Austria, 1987.

[GSM] Garrett, L. N., Smith, K. E., and Meyrowitz, N., "Intermedia: Issues, Strategies and Tactics in the Design of a Hypermedia Document System", Conference on Computer-Supported Cooperative Work Proceedings, Austin, Tx, Dec. 1986, pp. 163-174.

[G2] Goodman, D., "The Two Faces of HyperCard", Macworld, Oct. 1987, pp. 122-129.

[HMT] Halasz, F., Moran, T. and Trigg, R., "Notecards in a Nutshell", CHI and GI Conf. Proc.: Human Factors in Computing Systems and Graphics Interfaces, 1987, pp. 45-52.

[H] Huber, F., "On Customizing a PT-CAI Editor", IIG Report 249, Graz Univ. of Technology, Austria, 1988.

[HM1] Huber, F. and Maurer, H., "On Editors for Presentation Type CAI", Applied Informatics, No. 11, 1987, pp. 449-457.

[HM2] Huber, F. and Maurer, H., "Extended Ideas on Editors for Presentation Type CAI", IIG Report 240, Graz Univ. of Technology, Austria, 1987.

[HMM] Huber, F., Maurer, H., Makedón, F., "Hyper-COSTOC: A Comprehensive Computer-Based Teaching Support System", to appear in: Journal of Microcomputer Applications.

[MM1] Makedón, F., Maurer, H., "CLEAR: Computer Learning Resource Center", UTD CS Technical Report, 1986.

[MMO] Makedón, F. Maurer, H. Ottmann, T., "A Methodology for Presentation Type CAI in Computer Science Education at University Level", UTD CS- Report (1987) to appear in Journal of Microcomputer Applications.

[MMR] Makedón, F., Maurer, H. and Reinsperger, L, "On Active Annotation in CAI Systems", in preparation.

[MSI] Marchionini, G., and Sheiderman, B., "Finding Facts vs. Browsing Knowledge in Hypertext Systems", Computer, Jan. 88, 1988, pp. 70-80.

[M3] Maurer, H., "NatiSearch Intermediaries", J. Am. Society for Information Science, Vol. 34, 1983, pp. 381-404.

[M3] Maurer, H., "Nation-wide Teaching Through a Network of Microcomputers", IFIP-World Congress, Dublin, North Holland Publ. Co., 1986, pp. 429-432.

[MM2] Maurer, H., Makedón, F., "COSTOC: Computer Supported Teaching of Computer Science", UTD-CS Technical report 1986.

[MR] Maurer, H., Reinsperger, L., "Complex Execution Features of

CAI Systems and Their Execution With Limited Resources", IIG Report, Graz Univ. of Technology, in preparation.

[MS2] Maurer, H. and Stubenrauch, R., "GLSS: A General Lesson Specification System", IIG Report 241-87, Graz Univ. of Technology.

[MS3] Maurer, H. and Stubenrauch, R., "Filters for CAI", IIG Report, Graz Univ. of Technology, in preparation.

[MS4] Merrill, P. F., and Salisbury, D., "Research on Drill and Practice Strategies", Journal of Computer-Based Instruction, Vol. 11, No. 1, Winter 1984, S. 19-21.

[M4] Morris, J. H., et al, "Andrew: A Distributed Personal Computing Environment", Comm. ACM, Mar. 1986, pp. 184-201.

[R1] Rickel, H. W., "An Intelligent Tutoring Framework for Task-Oriented Domains", MS Thesis, UTD, 1987.

[TSH] Trigg, R., Suchman, L., Halasz, F., "Supporting Collaboration in NoteCards", Conference on Computer-Supported Cooperative Work Proceedings, Austin, TX., December 1986, pp. 153-162.

[WI] Ward, L. and Irby, T.C., "Classroom Presentation of Dynamic Events Using Hypertext", Twelfth SIGCSE Technical Symp. on Computer Science Education, ACM. SIGCSE Bull (USA) St. Louis, MO, 26-27 Feb, 1981.

[YLC] Yankelovich, N., Landow, G. P., and Cody, D., "Creating Hypermedia Materials for English Literature Students", IRIS, Brown Univ., Techn. Report, Providence, RI, Oct. 1986.

SCDAS - Decision Support System for Group Decision Making: Information Processing Issues

Andrzej Lewandowski
International Institute for Applied Systems Analysis
Laxenburg, Austria

Abstract

Most of research in the field of computerized Group Decision Support Systems is devoted to analysis and support the *quantitative* phase of decision processes using various methods of multiple-criteria analysis. The experience shows, that the *soft side* of the decision process needs also certain support. This relates mostly to distribution of *textual information* which augments the quantitative side of decision process and to providing the linkage between such information and numerical data. This aspect is especially important when the decision support system is implemented in distributed computing environment. In the paper the possible forms of information processed within the SCDAS system are analysed as well as the framework for implementation the software providing such processing functions is presented[1].

1 Introduction

The SCDAS system (Selection Committee Decision Analysis and Support) has been designed for supporting such decision problems, where the group of experts (the *committee*) cooperates to select the best alternative (or to reduce the set of alternatives to some reasonable subset which can be considered for further analysis) among alternatives presented to them by independently acting experts. Detailed assumptions and description of the SCDAS procedure are presented in the paper by Lewandowski and Wierzbicki (1987).

Up to now exist several experimental implementations of the SCDAS procedure (see Lewandowski, 1988). All these implementations have been prepared mostly to investigate the algorithmic and procedural aspects of SCDAS framework as well as to perform experimental applications of this methodology (for such an experimental application see Dobrowolski at all., 1987).

During experiments with existing prototype implementation of SCDAS as well as experiments with participation of decision makers it became clear, that support of quantitative aspects of decision process must be augmented by tools for supporting qualitative phase of this process. The idea that the *discussion* between committee members is one of the most important part of the decision process has been already mentioned in quoted above papers. It was stated by DeSanctis and Gallupe (1987):

> *"...A group decision occurs as the result of interpersonal communication - the exchange of information among members.....The communication activities exhibited in a decision-related meeting include proposal exploitation, opinion exploitation, analysis,*

[1]This paper is the shortened version of IIASA Working Paper WP-88-48

expression of preference, argumentation, socializing, information seeking, information giving, proposal development and proposal negotiations....In this sense the goal of GDSS (Group Decision Support System) is to alter the communication process within groups...."

Huber (1984) also expresses the importance of qualitative support for decision making:

> *"...Information sharing is the most typical of the activities in which groups engage... general GDSS can also enable groups to elicit, share, modify and use professional judgements and opinions in at least as many ways as they do hard data...."*

Without the consensus related to procedural principles and other important aspects of decision process it is not possible to provide any quantitative support. This consensus can be however reached only after *discussion* and *exchange of information* between committee members.

Therefore, the group decision support system should be treated as *information processing* and *information management* system.

2 Documents structuring in SCDAS system

Most existing GDSS is oriented toward processing numeric information (see Jarke, 1987, Bui and Jarke, 1986, Bui, 1987). The user of GDSS can enter numerical information to the system, retrieve this information, share with other users and perform rather complicated numerical procedures to extract important conclusions from this data. However, other types of information are also important for supporting decision processes. It has been pointed out by Huber (1984) that:

> *"...Today's DSS are largely concerned with the retrieval and use of numeric information. In contrast, the environment of most meetings in corporation and public agencies is highly verbal. Thoughts are primarily shared and modified, not numbers. To the extent that the thoughts need to be recorded, they are put into text form.... Meetings are extremely verbal environments, and the most important thoughts with which they deal are put into text form. A GDSS that does not reflect these facts will serve only a fraction of group tasks. For this reason it is important to consider how GDSS can support decision groups by aiding in the sharing of textual information...."*

As it has been pointed out in previous publications (Lewandowski, 1987), the most promising framework for implementation the group decision support system is the distributed computer environment equipped in teleconferencing and office automation software. Such an environment allows smooth transition from the existing practice of office and telecommunication systems utilization to new forms; moreover such an environment requires only small modifications or extensions to the existing software and methodologies to support new functions.

The basic idea of implementing the SCDAS in teleconferencing framework is the *extension of the concept of document*. In the standard office automation and teleconferencing systems the *text* - letters, memoranda etc. constitute the basic information carrier. In the *extended* or *decision* teleconferencing system the concept of *document* has been generalized - besides of textual data, numbers are transmitted between the members of the group. Moreover, the *formalized knowledge* necessary to interpret the data and to structure properly the decision process must be implemented within the system and made available in sufficiently simple and friendly form to the users of the system. Therefore, several types of documents can exist simultaneously in the extended teleconference system - documents which can be different nature and strongly interdependent. These dependencies can reflect logical relationships between numeric and textual data is well as can reflect the users opinion and knowledge related to the information being processed.

In order to design such extended teleconferencing system it is necessary to specify the possible types of documents generated and processed both by users and the system and rules for generating and processing of such documents.

Several types of documents can be distributed during a typical meeting. According to the agenda of SODAS decision conference, some steps of the decision process can be more oriented towards quantitative reasoning based on analysis of numerical or quantitative data, whereas the other require strong exchange of verbal information, a lot of discussion and more qualitative oriented analysis. The insight into the consecutive steps of the SCDAS process leads to the following conclusions;

The first stage. In the existing experimental implementations of the SCDAS system (Lewandowski, 1987) it was assumed, that the SCDAS conference begins with *Phase 0*, when all elements of the decision problem (i.e. *alternatives, attributes, committee members* and a *procedure*) are known; therefore during the *Phase 0* all these informations can be entered into computer. In the fact, reaching such a high level of common understanding and consensus within a committee could require a lot of discussions and information exchange. Neither the list of alternatives, nor their descriptions need be complete at this stage; moreover, this information might be not known to the committee members at this stage, if they wish to avoid the bias in specifying attributes and their aspiration levels. The important issue at this stage that requires discussion and specification by the entire committee is the definition of the attributes of the decision and their scales of assessment.

The questions formulated during this stage of the discussion could include the following:

1. What is the expected product of the committee work and how does it influence the selection of the details of the procedure?

2. What rules for aggregating opinions across the committee should be adopted, in particular, should outlying opinions be included in or excluded from aggregation?

3. Should the committee be allowed to divide and form coalitions that might present separate assessments of aspirations, attribute scores and thus final rankings of alternatives?

The second stage of the the decision process is devoted to *aspirations.* During this phase aspiration and/or reservation levels for all attributes are determined separately by each committee member. After these values are entered into the decision support system, all necessary indicators (disagreement indicators, dominant weighting factors - see further comments) can be computed. During this phase there will be no active information exchange between committee members - everybody should analyse the problem and specify aspirations himself.

The third stage has again two objectives. One is the analysis and discussion of aspirations by the entire committee. These discussions are supported by the computed indicators and their graphic interpretations. In these discussions, the committee might address the following questions:

1. Do the computed indicators accurately reflect the perceptions of individual committee members about the relative importance of various attributes (if not, should the aspirations or reservations be corrected)?

2. What are the relevant differences of opinions between committee members and do they represent an essential disagreement about decision principles?

3. Does the entire committee agree to use joint, aggregated aspirations (reservations), or will there be several separate sub-group aggregations?

The second objective of the third stage is a survey of alternatives. Discussions might centre on the following issues:

1. Are the available descriptions of alternatives adequate for judging them according to the accepted list of attributes? If the answer is negative, additional information should be gathered by sending out questionnaires, consulting experts etc.

2. Which of the available alternatives are irrelevant and should be deleted from the list? Such preliminary screening can be done in various ways. The committee might define some screening attributes and reservation levels for them (of a quantitative or simple logical structure): for example, we do not accept investments which are more expensive than a given limit.

The fourth stage of the decision process is the individual assessment of alternatives. The evaluation of each attribute for each alternative is the main input of committee members into the system. Each member specifies evaluation scores; the decision support system helps him by displaying the evaluations already made and those still to be entered.

When all evaluations are entered, a committee member should proceed to the individual analysis of alternatives, that leads to a ranking of all alternatives for the given committee member. This ranking is the main source of learning about the distribution of alternatives relative to aspirations.

The questions addressed by each member at this point might be as follows:

1. Do the rankings along each attribute correctly represent the individual's evaluations of alternatives; does the achievement ranking, based on individual aspirations, correctly represent the aggregate evaluation?

2. If the committee member agrees with the individual achievement ranking proposed by the system, what are the differences between this ranking and that based on individual scores but related to committee aggregated aspirations? Are these differences significant, or can he accept them as the result of agreement on joint decision principles?

The fifth stage of the decision process relates to an aggregation of evaluations and rankings across the committee and consists of a discussion of essential differences in evaluations, followed by a discussion of disagreements about a preliminary ranking of alternatives aggregated across the committee. These discussions are supported by the system; the system computes indicators of differences of opinion and prepares a preliminary aggregated ranking.

The questions addressed by the committee at this point might be the following:

1. On which attributes and alternatives the largest differences in evaluations between committee members are observed? Do these disagreements represent essential differences in information about the same alternative?

2. What is the essential information (or uncertainty about such information) that causes such disagreements? Should additional information be gathered, or can certain committee members supply this information?

3. Would the results of these discussions and possible changes of evaluations influence the preliminary aggregated ranking list proposed by the system? This can be tested by applying simple sensitivity analysis tools.

4. Does the preliminary ranking proposed by the system correctly represent prevalent committee preferences?

After these discussions, a return to any previous stage of the process is possible. If the committee decides that the decision problem has been sufficiently clarified, it can proceed conclude the fifth stage by the final agreement on the aggregated ranking or selection of one or more alternatives. It is important to stress again that the committee needs not stick to the ranking proposed by the system, since the purpose of this ranking - as well as of all information presented by the decision support system - is to clarify the decision situation rather than to prescribe the action that should be taken by the committee.

Let us analyse the possible types of information which can be processed within the SCDAS system. This information can be categorized according to two attributes: *information access and ownership* as well as *structural properties* of information.

The *access to the information* generated during the SCDAS session depends on two factors;

- *the privileges of the individuals participating* in the SCDAS conference. The rules are simple - the *conference owner* (or *committee president*) is the only person authorized to change the definition of the problem - like adding new committee members or removing them, changing the list of attributes or list of alternatives etc. He also can generate the textual informations relating to the problem definition or to the progress of the conference, which have *read-only status* for other committee members. Moreover, he can decide whether at a given stage of the process this information can be visible to other conference participants or will be hidden.

- *the stage of the process.* Since the SCDAS conference heis some temporal dimension - the decision process advances from the given stage to the next one if all committee members specified all information necessary on the given stage, the access rules can change in time.

With respect to structural properties, the information generated during the SCDAS conference can belong to two classes:

- the highly structured *numerical and qualitative data.* All the information constituting the problem definition, the relating information generated by the participants (aspirations, scores) as well as information generated by computer (values of achievement function, rankings, graph plots, etc.). There exist strong and well defined relationships between these data - we will call these relationships *structural links* in this sense that it is well defined what data are required from the conference participants at a given stage, what properties these data should possess, what actions (and calculations) are necessary to perform when data are entered to the system or changed by the user and what data must be used to calculate other numerical information.

- the unstructured *textual information* - like notes, memoranda, mail notes send to other conference participants. This information is similar to these generated and distributed during the standard conference. The only difference is in the structuring principle - usually, some part of this set of information can strongly relate to the numerical data. Therefore, the numerical data can be treated as the equivalent of *topic* in the standard conference - for every numerical item there can exist the linear list of comments generated by conference participants. Therefore the *hard links* between textual documents can exist - two linear links of comments will be *interrelated* if there exist some links between numerical data which this textual information is associated. We will call these links *hard* since they are *a'pri'ori* determined by the organization of SCDAS procedure.

Summarizing, the information generated during the SCDAS conference can be structured by the *structure of the decision process* itself. It is possible, however, that the second layer of links between numerical data and textual information can exist - namely links introduced by the user in order to reflect his particular, personal view on various aspects of the problem being solved. This kind of relation between documents we will call *soft links.*

The *soft links* can be arranged in similar way like it is done in hypertext system. In this way we have two, parallel layers of links - the *soft layer* and the *hard layer.* Therefore, contrary to the standard hypertext (see Conklin, 1987, Yankelovich at all., 1988) we will have the *primary relevant documents* and the *secondary relevant documents* - depending on the fact whether relevant documents are belonging to the same layer where the root of the search tree is located.

The difference between hard and soft links are not only formal - their existence can support different questions relating to the problem being analysed. If the question addresses the problem *"...how to explain the fact that my favourite alternative is ranked by the committee so low...."* in order to answer it is necessary to know what data directly influence this fact. This can be difficult issue for the user, especially if he does not know exactly the theory backgrounding the system. Since this theory is known to the system developer, he can establish the links which can help to trace data which are relevant to the posed questions. It is clear, that the answer on such question depends on the *state of the system* - understood as the values of data present in the system on the given stage of decision process. Therefore, some of these links can be *dynamic,* i.e. they can be changed during the progress of the decision process. Therefore, the mechanism for creating an updating such links must be built into the system. In order to fully support this function of the system, hard links must be provided for *help documents.* These documents play the role of standard context dependent help, but similarly like dynamic links these documents can also be dynamic - the help given to the user must depend not only the current state of the program (i.e. to address the question *where am I now*) but also address the current state of the system (*what follows from this*). These two functions of the system we will call the *guidance help* and the *explanatory help.*

The soft links play role of the *remainder* - the user can link and browse documents which he, or other participants consider as important on a given stage of the process. Evidently, these documents can contain both the numeric and textual data.

Summarizing, we can view the SCDAS decision conference as the document exchange problem with documents being *procedurally structured* and *contextually structured.*

3 Structured documents in SCDAS system

Let us discuss in details the structured documents which can be generated during the SCDAS session and possible relationships between them. The structured documents can be generally categorized into numerical and textual ones. As it was mentioned in previous sections the structured documents can be *generated by the conference participant* or *computed by the system.* The questions formulated now can be as follows:

- what types of structured data is required from the user on a given stage of the decision process and what the operational rules for handling these data,
- how these data can be used by the system on a given stage of the process,
- what are the possible dependencies between data entered by the user and/or generated by the system on various stages of the procedure,
- what actions are undertaken when a given action related to data is performed by the user.

The structured information required from the user (users) depends on two factors:

- the current pheise of the SCADS process,
- the privileges of the user entering and manipulating data.

The procedural framework presented in paper by Lewandowski and Wierzbicki (1987) specifies all the data created during performing the decision-oriented part of the conference. The data can be split into three following groups;

- Data characterizing the problem being solved. These data are generated by the conference owner and contain all the information necessary to initiate the conference. They include;

  1. List of *alternatives*, together with all documents characterizing these alternatives and necessary to make evaluation,
  2. List of *committee members*, together with *voting power*, specifying number of votes assigned to each committee member,
  3. List of *attributes* together with *numerical or verbal scale* necessary to express the value of attributes together with all relevant documents concerning the given attribute.

- Data created by the user during interaction with the problem. These data include;

  1. Values of *aspiration and reservation levels* specified for each attribute,
  2. Values of *scores* for all alternatives reflecting the subjective *value of alternative* with respect to all attributes,

- Data generated by the system during iteration process. They include;

  1. *Average aspirations and reservation levels* for all attributes,
  2. Values of *achievement functions* computed for scores specified by all committee members and for individual as well as committee aspirations. These functions are used by the system for ranking alternatives. See paper by Lewandowski and Wierzbicki (1987) for formulas and procedural details,
  3. *Ranking data* which reflect the ordering of alternatives according to the information specified by conference participant *individual aspiration* and *individual scores* as well as information relevant to the committee opinion *aggregated aspiration*,
  4. *Status indicator* generated by the system as the response for user's actions. Every user has his *local status indicator*, the system computes the *global status indicator*. These indicators reflect the phase of decision process and are equal to the number of phase being currently processed. The system compares individual status indicator with the global one and on the basis of this information determines what data are accessible for the user and what actions he can undertake. When the user terminates the current phase, his local indicator is incremented; the global indicator is incremented only if all users successfully completed the current phase. The global status indicator can be manipulated by the conference owner - he can, for instance *decrement* this indicator to make the recourse in decision process.

The rules specifying access to data created and analyzed during the decision conference are as follows;

- The *conference owner* has access *to all data* created during the conference with the following access rights;

  1. He is the only person authorized to change the problem definition, i.e. list of committee members, attributes and alternatives *(read - write access)*,

2. He has access to all data generated by the users, i.e. aspiration and reservation levels, scores assigned to alternatives and user's status indicators or computed by the system using user's data, like average aspirations or user's achievement functions (*read-only access*),
3. He can change the status indicator; this action will allow changes of user's status indicators (*read-write access*),

- The *conference participant* has access to the following data:

  1. All data defining the problem, i.e. list of committee members, attributes and alternatives (*read-only access*),
  2. Data computed by the system, like average aspirations and global achievement functions (*read-only access*),
  3. His own data like aspirations or scores assigned to alternatives (*read-write access*, or *read-only access* depending on the current value of status indicator),
  4. All the data located in his own notebook (*read-write access*)
  5. Data created by other user can be accessible by other users only with permission of the conference owner (*read-only access*).

Except of definition of data structures, we should investigate the temporal dependencies between data generated on various stages of decision making process as well as rules for sharing data between users during each stage of decision process. Therefore the following aspects should be investigated:

- What data are generated at every stage of the decision making process,
- How the data access rights are changed during this process,

The rules for data access are relatively simple: on a given stage of SCDAS process the conference participant has free access to data generated by himself and by the system during previous stages. This access is however restricted and such data can be *only inspected*. On a given stage of the process the system requests from the user some data; he can freely modify and read this data until he decides to terminate the current phase. It happens, when the conference participant decides that entered data reflect well his point of view about the problem and can be used for further computations. Since this moment the data is *locked* and available only for reading and inspection. Termination of the session changes the *local status indicator* (see above) which is used by the system for synchronization control. The committee president can change the status indicator what results in unlocking the data generated during previous stages. In this way the recourse in decision process can be performed.

Let us concentrate on details of operations performed during every step of decision making process:

- *Phase 0.* During this phase the conference president can initiate the new conference of update the old one. The standard sequence of actions undertaken during this step consists of the following:

  1. Specification of user's name and verification of access mode (the ordinary conference participant or the conference president)
  2. Verification of the user's name. If such a name is not known to the system, new conference should be initiated.

Exit from this stage of the program is possible in two modes - the *quit* mode and *terminate mode*. In *quit* mode the program terminates, but the data are not transferred to the global data base. Therefore, the user can invoke the program again and perform necessary data modification. It *terminate* mode, all data are transferred to the data base. In this case the *local status indicator* is also updated as well as the *global status indicator*.

- *Phase 1.* In this phaise all users should define aspiration levels for all attributes. The preamble of this phase is similar to the previous one;
  1. specification and verification of user name. The request is rejected if the specified name is not known to the system (i.e. he is authorized to participate in any conference already defined). The request is also rejected if the user terminated the current phase and wants to modify some data - according to the procedure it is not possible without acceptance of the conference president.
  2. creating (or updating) the data base with the list of attributes and numerical data associated with attributes. Similarly like in previous phäse, the program can be terminated in two modes - the *terminate mode* quits the program only; the data are not transferred to the data base and status indicator is not updated. Therefore the user can resume this phase as many times as he requires until he decides that he specified all required data. In such a case he should exit with a *quit* mode, what initiates updating the global data base with new aspiration data. The system checks the status indicators of all users - if all of them completed the current phase, the global status indicator is incremented. In such a case the users can begin performing the next phase of the process.
- *Phase 2.* In this phase the user can perform the analysis of data specified during the previous phase. After verifying the user's name and the status indicator, the user can perform all necessary data analysis. Similarly like in the previous phase, it is possible to exit the workstation program in *terminate* or *quit* mode. The standard procedure for incrementing the status indicator is performed.
- *Phase 3.* In this phase the user must specify scores for all attributes. This is definitely the most complicated and time consuming phase of the decision process which may request rather intensive interaction with other data information systems and services available to the user as well as rather deep analysis of the specified data. After terminating this phase the score table is transferred to the global data base and the standard procedure for incrementing the status indicators is performed.
- *Phase 4.* In this phase the final analysis of the data is performed. When all users completed the previous phase, the *achievement functions* for scores specified by all committee members are computed. Values of these functions are used for ranking alternatives. Since this is the last phase of the process, status indicators are not updated.

4 Unstructured documents in SCDAS system

As it was mentioned in previous sections, except of highly structured information related to alternatives, attributes, scores etc., the SCDAS system supports generation, exchange and analysis of several types of *unstructured* information. This function of the system supports the *soft* side of the decision process – in many cases more important for obtaining final result and usually requiring more effort to complete than just collection of scores and computing of rankings.

The basic element of this side of the process is *the document*. We will understand this term in narrow sense – the document will be *non-active, textual or graphic information*. We will not consider more general case, when the document can be *active* —i.e. distributed together with tools for analysis of this document. This is not necessary, since all data analysis in SCDAS is concentrated in data nodes of the information structure; if the conference participant wants to perform some *what-if* analysis he can easily duplicate the data node and make his private copy; all tools for analysis are available to him all the time (with some natural constraints following from the SCDAS procedure).

Each document can belong to one of four groups of documents:

- *Public documents* which are generated by the committee president. These documents contain general information about the particular aspects of the problem – in the case of attribute it can be, for example, detailed explanation of the meaning of this attribute, in the case of alternative – information about this particular alternative, like curriculum vitae for personnel selection problem, details of the project for project evaluation problem etc.

- *Message documents* which are generated by conference participants as their contribution to the discussion. These documents are available to all conference participants, however the only person which can modify these documents or remove them from the system is the conference president.

- *Private documents (notes)* which contain the information generated by the conference participant and stored for himself for future utilization *(the notebook)*. Any document available to the conference participant can be imported to the notebook, including help, public and message documents.

- *Mail documents* which contain information received from other conference participants or send to other participants. This information is accessible exclusively for a person identified as the receiver and the author.

The documents created during SCDAS conference are *linked*. As it was mentioned in the previous sections, some links have *organizational* character – i.e. they are predefined by the SCDAS procedure. Documents can be also linked by *referential links* pointing the information which not necessary belongs to a given category, but can be interesting or relevant from a given point of view. Usually, these links do not reflect *the logical relationships* between data, but rather *the contextual* relationships. All conference participants have full freedom to create referential links – both between structured and unstructured documents.

5 Implementation

The ideas presented in the paper have been experimentally implemented on the IBM-PC computer and the Vax-Mate (the IBM-AT compatible manufactured by DEC) using the SMALL-TALK/V programming language (DIGITALK, 1986). The main purposes of this implementation were as follows:•

- to prototype the user's workstation for distributed group decision support system based on the SCDAS methodology and utilizing the standard teleconferencing software like the Telecenter developed at IIASA (Pearson at all., 1981, Fuhrmann, 1987) or the NOTES teleconferencing system manufactured by DEC,

- to investigate the feasibility and efficiency of presented concept of documents structuring,

- to clarify the ergonomic aspects of the user interface.

Details of design and implementation of prototype system as well as principles of cooperation between the workstation and teleconferencing software will be presented in a separate publication.

6 References

Bui T.X. and M. Jarke (1986). Communications Design for Co-oP: A Group Decision Support System. *ACM Transations on Office Information Systems*, Vol. 4, No. 2, April 1986.

Bui T.X. (1987). Co-oP: A Group Decision Support Systems for Cooperative Multiple Criteria Decision Making. Lecture Notes in Computer Science, Vol. 290, Springer Verlag.

Conklin, J. (1987a). A Survey of Hypertext. MCC Technical Report No. STP-356-86, Rev. 2, Software Technology Program, December, 1987.

Conklin, J. (1987b). Hypertext: An Introduction and Survey. *IEEE Computer*, September 1987, pp. 17–41.

DeSanctis, G. and R.B. Gallupe (1987). A Foundation for the Study of Group Decision Support Systems. *Management Science*, Vol. 33, No. 5, May 1987.

DIGITALK, Inc. (1986). SMALLTALK/V – Tutorial and Programming Handbook. Los Angeles, 1986.

Dobrowolski, G. and M. Zebrowski (1987). Ranking and Selection of Chemical Technologies: Application of SCDAS Concept. In: A. Lewandowski and A. Wierzbicki, Eds., *Theory, Software and Testing Examples for Decision Support Systems*, WP-87-26, *International Institute for Applied Systems Analysis*, Laxenburg, Austria.

Fuhrmann, C. (1987). TELECTR User's Manual. IIASA Software Library Series, LS-16, *International Institute for Applied Systems Analysis*, Laxenburg, Austria.

Huber, G.P. (1984). Issues in the Design of Group Decision Support Systems. *MIS Quarterly*, September 1984, pp.195-205.

Jarke, M., M.T. Jelassi and M.F. Shakun (1987). MEDIATOR: Toward a Negotiation Support System. *European Journal of Operational Research*, No. 3, September 1987.

Lewandowski, A. and A.P. Wierzbicki (1987). Interactive Decision Support Systems – The Case of Discrete Alternatives for Committee Decision Making. WP-87-38, *International Institute for Applied Systems Analysis*, Laxenburg, Austria.

Lewandowski, A. (1988a). SCDAS-Decision Support System for Group Decision Making: Short User's Manual. *International Institute for Applied Systems Analysis*, Laxenburg, Austria, unpublished manuscript.

Lewandowski, A. (1988b). Distributed SCDAS – Decision Support System for Group Decision Making: Functional Specification. *International Institute for Applied Systems Analysis*, Laxenburg, Austria, unpublished manuscript.

Lewandowski, A. (1988). SCDAS – Decision Support System for Group Decision Making: Information Processing Issues. WP-88-48, *International Institute for Applied Systems Analysis*, Laxenburg, Austria.

Pearson, M.L. and J.E. Kulp (1981). Creating and Adaptive Computerized Conferencing System on UNIX. In: R.P. Uhlig, Ed., Computer Message Systems, North-Holland, 1981.

Yankelovich, N., B.J. Haan, N.K. Meyrowitz and S.M. Drucker (1988). Intermedia: The Concept and the Construction of a Seamless Information Environment. *IEEE Computer*, January 1988, pp. 81–96.

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 57-68.

COOPERATIVE OFFICE

L. Csaba, E. Csuhaj-Varjú

Computer and Automation Institute
Hungarian Academy of Sciences
H-1132 Budapest, Victor Hugo u. 18-22.
Hungary

Abstract

The present paper gives a society model of office information systems, highly motivated from the theory of blackboard-type distributed problem solving systems. A formal, generative system-oriented definition of an office society is introduced, using the fact of formal similarity of usual intelligent office activities to rewriting systems. Some notions are defined for measuring and characterizing descriptive and behavioural properties of offices. Some statements, illustrating the notions and having comparative character are presented.

Keywords

office information systems, blackboard-type distributed problem solving, office modelling, formal languages

1. INTRODUCTION

With the widespread appearance of computer networks computer supported collaborative activities have gained an increasing importance from both practical and theoretical points of view. The claim of intelligent general tools for and approaches to the development and handling of the theory and practice of the intelligent and cooperative office has been pushed for time. Office information systems and their levels have different approaches and models of description, motivated from different practical considerations of computerization and computation ([1]). Most of the approaches are either of syntactic or of semantic character, usually the two types are in conflict. (For a syntactic approach see [2]). The different approaches emphasize different aspects, as procedural [3,2], declarative [4] or object-oriented [5] aspect. A society model of office information systems can be found in [6]. Artificial intelligence techniques are used in [7], where an office task execution is considered to be a problem solving action. We can easily see that usual computerized activities in offices, as real-time computer conferencing, distributed text processing, meeting organization, the usage of electronic bulletin boards, can be considered as distributed problem solving actions, showing formal similarity to the well-known

blackboard-type distributed problem solving systems. (More details about blackboard-type problem solving systems and real-time computer conferencing can be found in [8,9,10] and [11], respectively.) The observation of that cooperative grammar systems are generative paradigms (models) of both of the above mentioned activities and the blackboard-type problem solving systems was presented first in 1987 in [12] and [13]. The idea was developed further in [14] and [15].
In this paper we present an office-oriented version of this generative model and we introduce a formalism for a descriptive characterization of offices and for that of their behaviour. The model is essentially an actor-based society model.(For more information concerning artificial intelligence motivation the reader is referred to [16,17,18,19]. Finally, we interprete some results of [12] and [14] in this frame.

Throughout the paper we assume elementary knowledge of the reader from the theory of formal languages [24].

2. THE OFFICE MODEL

By an office we mean a collection of functionally related communities, called office-societies, which collectively produce a family of problem solutions, called office-problem solutions, or achieving goals, called office-goals. (Goals are understood as missions, achievements of which are represented by problem solutions.) Note that we do not reflect problems being unsolvable from the point of view of the society. As we mentioned in the introduction, problems occurring in and being characteristic for offices are considered in the most general sense, therefore we make a distinction in the terminology, using the prefix "office-" of the corresponding terms.

We assume office-problems to be well-structured, that is partial and final office-problem solutions are considered to be strings (sequentially organized forms) of nondecomposable knowledge pieces (solution elements) which are called elementary office-knowledge pieces (office information pieces). (For example, undecomposable messages in a real-time computer conference are elementary office-knowledge pieces.) For a string of elementary office-knowledge pieces we use the terminology office-knowledge piece. In order to make a distinction between partial(non-final) office-problem solutions and the final solution we asssume that from the point of view of the problem-solvers elementary office-knowledge pieces are divided into two disjoint families, depending on that whether the elementary office-knowledge piece has final or nonfinal (auxiliary) character. A final office-problem solution is composed from elementary office knowledge pieces having final character.
The notion of a final elementary office-knowledge piece expresses that in that moment and in that context from the point of view of the office-agent the knowledge piece contains full information about the piece of the world it refers to. Non-final elementary office-knowledge pieces contain information which should be or can be developed. (Consider the case of computer conferencing.

The message " Stephen Cook is a member of Group 6." is final office-knowledge piece for a participant of the conference who knows what Group 6. means, but from the point of view of a person being not in the possession of this information this message is a non-final knowledge piece.)

Office-societies are collections of interacting office-agents, collectively solving a family of office-problems. Naturally, an office-agent can be a member of more than one office-society. Note that here interaction is considered in a restricted form, that is, it is realized by the contributions to the problem solving process.

The collection of office-problem solutions produced by an office-society is called its office-problem solving capacity.

Office-agents are intelligent office entities, represented by knowledge sources. (For example, persons, expert systems, etc. can be considered as office-agents.) They have condition-action format, where the condition describes the situation when the office-agent can contribute to the problem-solving process. In this case the situation means that the actual partial solution satisfies some conditions. The action-part describes the behaviour of the office-agent. (Consider the case of collective text-creation. Let "Writing an article about computer networks" be an office-problem. For example, a condition-action representation is as follows: a person (an office-agent) can rewrite the second section of the current version of the article (can modify partial office-problem solution by performing the action described in the action-part) if the section contains subsection titled "Local area networks " (the condition-part)).

Referring to the problem-solving processes in the office-societies, we assume that during the process partial solutions are recorded in a common (abstract) space, called the office-board, which can be accessed by any cooperating office-agent. Note that in every moment only the current partial solution is recorded in the space. (We can easily see that the analogy with blackboard-type problem solving systems is obvious.(For detailed information see [8,9,10].) The reason of the different usage of terminology is the more general treatment of the problems. For example, the message space of a real-time computer conferencing system or a text, edited collectively by a group of persons, can be considered as an office-board, respectively.)

The problem solving process begins in an initial state of the office board (office-problem initialization). Then it is continued by consecutive steps, represented by a sequence of partial solutions of the problem, forming the development to the office-knowledge piece being the office-problem solution and being placed in the corresponding office-board. The partial solutions are obtained by contribution steps of office-agents, which are enable to do it, that is, which satisfy some conditions. The process ends in a previously defined state of the office-board, that is by achieving the a final solution of the office-problem.

The contributions of the office-agents to the problem solving processes are defined as follows: in every step the corresponding (the enable) office-agent executes an office-action, that is makes a modification on the actual partial solution being in the office-board. Essentially, the modification means the following: some parts of the current partial solution remain unchanged and some of them are replaced by office-knowledge pieces. Note that we do not require an effective modification of the office-board, the case when every elementary office-knowledge piece is changed for itself is assumed to be a contribution step, too. Also, we have to emphasize that final elementary-knowledge pieces can be replaced by office-knowledge pieces, too. We can see that here is the connection point with generative rewriting systems theory, as, this process shows formal similarity to the notion of a direct derivation step defined there.

As we have mentioned, from the point of view of an office-agent elementary knowledge pieces can be divided in two disjoint families, that is in the family of final and in that of non-final elementary office-knowledge pieces.
The collection of final office-knowledge pieces of an office-society is the collection of elementary office-knowledge pieces which are final elementary office-knowledge pieces of every office-agent.

Office agents of an office-society can work with and without an outer control. The control is used for selecting an office-agent from the potentially enable ones to perform an office-action. Controls can be static or dynamic ones. In the first case the control does not depend on the partial solution obtained in the problem solving process. Dynamic controls depend both on the members of the office-society and on the partial results of the office-problem solving process.

3. A FORMAL MODEL OF OFFICE-SOCIETIES

The previously defined society model is based, essentially, on the notion of an office-society. The following formal model is a generalization of a generative paradigm, that is of the system of cooperating grammars, which was first introduced in [12] and developed further in [13], [14] and [15]. (The notion of a cooperating system of grammars in another form was defined first in [20] and it was generalized in [21] and [22]). Note that in [12] the term "distributed grammar systems" is used instead of "cooperative".)

In the following we define the notion of an office-society. We note that we consider office-agents to be of having finite knowledge, that is containing a finite number of elementary office-knowledge pieces. The model is highly motivated by the versions of [13,14].

DEFINITION 1

An office-society is a 8-tuple $S=(A,C,F,N,START,STOP,B,ACC)$, where

(i) $A=\{ A_i : 1 \le i \le n \}$ is a finite set of office-agents of S, where $A_i=(F_i, N_i, P_i)$, where

(a) F_i is a finite set of elementary office-knowledge pieces, called final elementary office-knowledge pieces of A_i,

(b) N_i is a finite set of elementary office knowledge pieces, called non-final elementary office-knowledge pieces of A_i, F_i and N_i are disjoint sets,

(c) P_i is a finite set of productions of A_i having the form
$p(w)$: if $cond_p(w)$ holds and w_1 is a substring of w then replace w_1 in w by w_2, where w_1 and w_2 are office-knowledge pieces over $(F_i \cup N_i)$, w is an office-knowledge piece over $(F \cup N)$ and $cond_p$ is a mapping from $(F \cup N)^*$ into $\{true, false\}$,

(ii) C is a mapping from $A \times (F \cup N)^*$ into 2^A called the control of S,

(iii) F is the set of final elementary office-knowledge pieces of S such that F is a subset of every F_i, $1 \le i \le n$,

(iv) N is the set of non-final elementary office-knowledge pieces of S such that every N_i of A_i of A is a subset of N, where $1 \le i \le n$,

(v) $START=\{START_i : 1 \le i \le n \}$, where $START_i$ is a mapping from $(F \cup N)^*$ into $\{true, false\}$, called the set of starting predicates of office-agent A_i, respectively,

(vi) $STOP=\{STOP_i : 1 \le i \le n \}$, where $STOP_i$ is a mapping from $(F \cup N)^*$ into $\{true, false\}$, called the set of staying predicates of office-agent A_i, respectively,

(vii) B is a set of office-knowledge pieces over $(F \cup N)$, called the set of office-problem initializations (the initial states of the office-board) such that for ony b in B there is an i, $1 \le i \le n$, such that b is an office-knowledge piece over $(F_i \cup N_i)$,

(viii) ACC is a mapping from $(B \times F^*)$ into $\{true, false\}$, called the set of acceptable final solutions of the initialized office-problems from the point of view of the collection of office-agents of A.

If $C(A_i,w)=A$ for every i, $1 \le i \le n$, and every office-knowledge piece w over $(F \cup N)$ then we say S is defined without control.

It can be easily seen that this formal definition is convenient for modelling some usual office activities. Note that the acceptability of the solutions means that the office-agents have some idea about the solution they should like to achieve.

Example

Note that if we define elementary office-knowledge pieces, condition-action productions, starting and staying predicates, initial states of the office-board in an appropriate way then we obtain, as a special case, the definition of a cooperating (distributed) grammar system defined in [12], [13] and [14].
This can be obtained as follows: in condition (i) of Definition 1

let production p of P_i defined as follows: let $w_1 = X$, where X is an element of N_i. Let $cond_p(w)$=true for all words w in which $z_1, \ldots, z_n$ and X occur as subwords and let $cond_p(w)$=false otherwise. Thus, if $z_1, \ldots, z_n$ and X occur in the office-board represented by word w for which $cond_p(w)$ holds then we can replace X by w_2. Conditions (ii-vi) can be obtained obviously. In condition (vii) S={S}, where S is an element of F_i for some i. Condition (viii) is defined as follows : ACC(S,w)=true for every element w of $F^{\#}$.

Next we define how the office-society works.

DEFINITION 2

Let S=(A,C,F,N,START,STOP,B,ACC) be an office-society, defined as in Definition 1. Let w_1 and w_2 be partial solutions of an office-problem b, initialized in B.
Let A_i be an office-agent in A. Let w_1=xyz and w_2=xvz, where y and v are office-knowledge pieces over (F_i U N_i), x and z are office-knowledge pieces over (F U N). Let p be a production in P_i having the form " IF $cond_p(w)$ holds and y is a substring of w then replace y in w by v". If $cond_p(w_1)$=true then we say that offfice-agent A_i is enable to modify w_1 for w_2 by executing one office-action.

NOTATION
The execution of an office-action is denoted by ->.

DEFINITION 3
Let S=(A,C,F,N,START,STOP,B,ACC) be an office-society, defined as in Definition 1.
An office-problem solving process *proc* (initialized by w_0 and resulting w_k) is defined as follows:

$$proc;\ w_0 \underset{A_{i_1}}{\rightarrow} w_1 \underset{A_{i_2}}{\rightarrow} w_2 \underset{A_{i_3}}{\rightarrow} \cdots\cdots \underset{A_{i_k}}{\rightarrow} w_k,$$ where

(i) w_0 is an element of B,

A_{i_1} is enable to modify w_0 for w_1 and $START_{i_1}(w_0)$=true

(ii) w_k is an office-knowledge piece over F such that

ACC(w_0,w_k)=true

(iv) for every j, 2≤j≤k

(a) A_{i_j} is enable to modify w_{j-1} for w_j and

if $STOP_{i_{j-1}}(w_{j-1})$=false then $A_{i_j} = A_{i_{j-1}}$,

otherwise $START_{i_j}(w_{j-1})$=true

and A_{i_j} is an element of $C(A_{i_{j-1}}, w_{j-1})$.

DEFINITION 4

Let S=(A,C,F,N,START,STOP,B,ACC) be an office-society.
The office-problem solving capacity CAP of S is defined as the set of office-knowledge pieces over F which are results of office-problem solving processes.

We can see easily, that in the particular case of cooperating grammar sysytems, office-problem solving processes correspond to correct derivations and the office-problem solving capacity of the society corresponds to the generated language.

4. OFFICE CHARACTERISTIC

Formal models ensure the possibility of the formal characterization of offices from different points of view. The necessity of formal characterizations is obvious, as from organizational, constructional points of view of a computerized office we need formalized information both about its structure and about its behaviour.
Note that the notions, which will be introduced in the following, were defined and examined for cooperating grammar systems in [12] and for the model of blackboard-type distributed problem solving systems in [15].

First, office-problems can be classified with respect to the simplicity of their solutions,in absolute and relative sense. Examining our model we obtain the following: we can define an office-problem solution to be simple if in order to execute an office-action the contributor office-agent does not need to be in possession of recognizing large connected parts of/knowing too much about the office-board and its starting and staying predicates do not presume too much knowledge, too.
Using the notion of the problem solving capacity of office-societies we can define simple and complicated office-problem classes, also we can introduce the notion of office-agents with simple knowledge application capability .
We define an office-problem for an office-society and an office-problem-solution to be simple, if there is at least one way to achieve its solution such that every partial solution can be obtained from the previous one only by replacing an elementary office-knowledge piece by an office-knowledge piece and the conditions of the applied productions presume the recognition only the elementary knowledge piece being replaced, starting and staying predicates of the collaborating agents are assumed to be having the value true for all office-knowledge pieces and the set of the acceptable final solutions consists of all office-knowledge pieces composed of final elementary office-knowledge pieces. We can easily see that this notion covers independency, and compared to cooperating grammar systems theory, it corresponds to a context-free derivation.

We say an office-agent to be simple if its condition-action rules are of type of the previously defined ones. A class of office-problems is a said to be a simple class of office-problems if every problem has a simple solution. (Naturally, this notion corresponds in generative terminology to a context-free language).

The next point of characterization is the possibility of defining descriptional complexity measures of offices. Formalized office-societies can be described by their size and structural properties, being characteristic of them and their behaviour. The notions are motivated by notions introduced for formal grammars in [24].

Size measures for example can be introduced as follows: the number of office-agents of an office-society, the number of condition-action productions of an office-agent (the maximum of the numbers of office-agents of the office-society), the number of non-final elementary office-knowledge pieces of an office-agent, the minimal number of non-final elementary office-knowledge pieces needed to produce the same office-knowledge capacity by an office-society, etc.

Structural complexity measures are the following: the number of groups of strictly collaborating agents, where by a strict cooperation we mean the following: two office-agents are in strict cooperation if there is at least one problem solving process when one of them developes a knowledge piece formerly produced by another one. An interesting structural complexity measure can be the maximal number of non-final elementary office-knowledge pieces occurring in partial solutions of a problem during the problem solving process.

Note that size measures have special importance from the point of view of economy of the creation of office-societies, giving the chance of achievements of our goals in a more economic way by using a more appropriate reconfiguration and reorganization of available resources. Structural properties express information about the way of problem solution, giving a possibility of defining office-problem classes. The second structural complexity measure defined above can be interpreted as the number of open questions being involved in the current office-board.It is obvious, that from practical point of view the number of open questions in a moment during the problem solving process is a relevant property. Note that this notion is a generalization of the well-known notion of the index of the derivation in formal language theory.

Another important point of examination is the characterization of the behaviour of office-societies, emphasizing the role of office-agents in the cooperative problem solving. Office-agents and office societies can apply strategies in the problem solving processes (implicitely, controls define strategies, too). From the point of view of office-agents strategies can be of two types: self-strategies, which are characteristic only for the office-agent and are independent from the behaviour of others, and collective strategies which determine the collective behaviour of the office-agents of the office-society. Collective strategies raise such type of important questions as the question of fair behaviour. Without the aim at completeness we give some examples for self- and collective strategies. A self-strategy can be of so-called "one comment"/"full comment" strategy. In the first case the office-agent modifies one office-knowledge piece in the office-board, in the second case it modifies all knowledge pieces which are available for it but there is no knowledge piece which is modified during this process two times. Two examples far collective stategies are the following: the first, the strategy of step number limitation, when every office-agent can execute consecutively (at least) exactly k-office actions, and the second, the strategy af comparative competency, when that office-agent performs the action which is the most (least, competent of measure of k, etc.) is competent, where competency is defined on the basis of knowledge pieces being modifiable from the point of view of office-agents. Moreover, a collective strategy can be defined en the basis of temporaly monopolization, which means that until an office-agent satisfies the conditions of the strategy (and naturally other requirements) then it continues the contribution process to problem solving. A collective strategy can be free, where the notion covers functioning without any restriction.

Note that both descriptional complexity measures and both strategies are in very strong connection with the efficiency of offices.

A. RESULTS

In this section we interprete some results of [12] and [1A], obtained for cooperating grammar systems, illustrating the notions defined previously, with showing the possibility of achieving general statements for computerized offices defined in an abstract way.
We state the theorems without proofs, the technical details come with simple considerations from the corresponding results of the above mentioned two papers.

The next theorem is an example for the strict connection between strategies and size and structural properties of offices.

THEOREM 1
For any simple office-society S defined without control, where office-agents use self-strategy of "one comment" and S uses collective strategy monopolization until satisfying minimal competency (defined as the recognition of at least one non-final elementary office-knowledge piece) there is a simple office-society S' defined without control, which consists of two office-agents, where office-agents use self-strategy of "full comment" and S" uses free collective strategy such that their office-problem solving capacity is equal.

The next statement shows that in some particular cases the role of an outer control in the problem solving process is not relevant. The theorem is an example for the cooperation of office- agents having fair behaviour.

THEOREM 2
The class of office-problem solving capacities of simple office-societies defined without control, where office-agents use self-strategy of "one comment" and the society uses collective strategy "executing consecutively exactly k-steps" is equal to the class of office-problem solving capacities of the same type of simple offices defined with control.

Before presenting our next statement we note that the "full comment" strategy expresses paralelism and the "one comment strategy" expresses sequential behaviour in some sense. The following theorem says that, in some special cases, sequential organization provided with even some very simple additional features results richer problem solving capacity than a simple, paralel organization.

THEOREM 3
The class of office problem-solving capacities of office-societies defined without control, which consist of simple office-agents, which use self-strategy of "one comment", and the starting and staying predicate of every office agent are defined as the recognition/nonrecognition of finite sets of elementary office-knowledge pieces, respectively, and the office-society uses free collective strategy, strictly includes the class of office-problem solving capacities of simple office-societies defined without control, which use free collective strategy and its office-agents use self-strategy of "full comment".

5. FUTURE

We can observe that our model is highly motivated from the theory of classical(sequential) rewriting systems. The cooperative aspect of generation can be imagined as a useful tool for modeling more complicated structures, for example graphs. We think that the theory of cooperating graph-grammars can be a very promising tool in the modeling of distributed problem solving and can form a theoretical base of handling usual graph-oriented representations in artficial intelligence.

REFERENCES

1. *Office automation* .Topics in Information Systems. D. Tsichritzis Ed., Springer-Verlag, Berlin, 1985.

2. ELLIS, C. A. Information control nets: A mathematical model of office information flow. n *ACM Proc. of the Conf. on Simulation, Modeling and Management of Computer Systems* (Aug.) ACM, New York, 1979, pp. 225-240.

3. CHANG, J. M. AND CHANG, S.K. Database alerting techniques for office activities management. *IEEE Trans. Commun.* 1 (1982), 74-81.

4. GIBBS, S. J. Office Information models and the representation of office objects. In *SIGOA Conf. on Office Automation Systems*, ACM, New York, 1982, pp. 21-26.

5. BOOCH, G. Object-oriented development. *IEEE Trans. Softw. Eng.* 2 (1986), 211-221.

6. HO, CH-S. AND HONG, Y-CH. A society model for office information systems. *ACM Trans. Off. Inf. Syst.* 4 2 (Apr.1986), pp. 104-131.

7. BARBER, G. Supporting organizational problem solving with a work station. *ACM Trans. Off. Inf. Syst.* 1. 1 (Jan.1983), pp. 45-67.

8. NII, H.P. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *THE AI MAGAZINE*, (Summer.1986),pp. 38-53.

9. NII, H.P. Blackboard systems. Blackboard application systems, blackboard systems from a knowledge engineering perspective. *THE AI MAGAZINE*, (Aug. 1986), pp. 82-106.

10. HAYES-ROTH, B. Blackboard architecture for control. *Artif. Intell.* 26 1985), pp. 251-321.

11. SARIN, S. AND GREIF, I. Computer-based real-time conferencing systems. *IEEE Computer* 18 (Oct. 1985), pp. 33-45.

12. CSUHAJ-VARJU, E. AND DASSOW, J. Distributed grammar systems. To appear in EIK.

13. CSUHAJ-VARJU, E. AND KELEMEN, J. Cooperating grammar systems: a grammatical framework for blackboard model of problem solving. Manuscript, 1987.

14. PAUN, GH. Conditional starting and staying tests for cooperative grammars. Manuscript, 1988.

15. CSUHAJ-VARJU, E., DASSOW, J., KELEMEN, J. AND PAUN, GH. Cooperative knowledge, in preparation.

16. HEWITT, C. Viewing control structure as patterns of passing messages. *Artif. Intell.* 8 (1977), pp. 323-364.

17. SMITH, R.G. AND DAVIS, H. Frameworks for cooperating in distributed problem solving. *IEEE Trans. Syst. Cyb.* 11 (1981), pp. 61-70.

18. MINSKY, M. The society theory of thinking. In *Artificial Intelligence; An MIT Perspective 1* , P.H. Winston and R.H. Browns, eds. The MIT Press, Cambridge, Mass., 1979.

19. KELEMEN, J. Two notes concerning the society theory of Minsky. *Computers and Artificial Intelligence* 5 (1986), 43-52.

20. MEERSMAN, R. AND ROZENBERG, G. Cooperating grammar systems. In *Proc. of MFCS'78*, Lecture Notes in Computer Science 64, Springer-Verlag, Berlin, 1978.

21. CSUHAJ-VARJU, E. Some remarks on cooperating grammar systems. In *Proc. of Conf. on Automata, Languages and Programming Systems*. F. Gécseg and I. Peák Eds. Karl Marx University of Economics, Budapest, 1986. pp. 75-86.

22. CSUHAJ-VARJU, E. AND DASSOW, J. Cooperation of grammars. *MTA SZTAKI Working Paper*, SzT29 , 1987, pp.15

24. SALOMAA, A. Formal languages. *Academic Press*, 1973.

25. GRUSKA, J. Some classification of context-free languages. *Inf. and Control* 14 (1969), 152-179.

MAN - MACHINE SYSTEMS

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 71-80

HIBOL-2 AS A PARADIGM FOR END-USER-ORIENTED COMPUTING

Roland T. Mittermeir and Heidemarie Wernhart
Institut für Informatik
Universität Klagenfurt
UniversitätsstraBe 65
A-9020 Klagenfurt/AUSTRIA

This paper presents a brief introduction into the design rationale of HIBOL-2, a programming language designed for end-user computing in office environments. The diffenences in the construction process of physical systems versus software systems is demonstrated and used as justification for several design decisions made in HIBOL.

KEYWORDS: Software-Ergonomics, Man-Machine-Interface, Language Design, HIBOL-2

1. INTRODUCTION

The advances in information technology and in micro-electronics brought relatively inexpensive but highly powerful equipment into the hands of broad sections of the workforce, notably of clerical workers. However, the advances made by the informatics industry outpace the learning capabilities of a high proportion of potential users. Hence, one has to open avenues which bring computing power into offices without disrupting the existing work structures and without requiring to master new paradigms.

This paper reports on research and associated developments conducted at the Institut für Informatik at the Universitat Klagenfurt to achieve the goals mentioned above.

HIBOL-2 is an example of a programming language which is integrated within its own programming environment. HIBOL-2 is designed according to a desk-top metapher. An application is modelled as a desk on which information is flowing between forms layed out on the desk. Describing this flow of information and derivation of new information, the user in fact specifies a program which will then execute either independently or under the guidance of the programming environment.

Funding of the development of the HIBOL-2 environment by the Austrian Forschungsförderungsfonds, grant P 5341, is gratefully acknowledged.

During the first part of this research, a number of interesting aspects of software-ergonomics as well as considerations of language design could be studied. The implementation gave rise to the development of an incremental compiler observing aspects of software reuse and object oriented design.

2. OFFICE SYSTEMS AS A SPECIAL KIND OF MAN-MACHINE SYSTEMS

With the increasing penetration of society by technical systems, offices too became invaded by technology. There, the path of evolution can be followed from simple typewriters and calculating machines via electronic typewriters with memory to textprocessing systems and fully equipped micro-computers or highly powered workstations equipped with powerful text-processing-, data- and information base management systems and decision support software.

Hence, as with the advent of most socio-technical systems, the question of whether systems can only be used as bought off the shelf, or whether such systems can be installed as adaptable shells which will be filled and gradually enriched and adapted by their users because relevant for this environment. Obviously, as also shown in the literature on man-machine systems (e.g. [MUMF 81]) as well as in works on system development methodology (e.g. [FLOY 84]) or office information systems [BRAC 84, HÄGG 88], the latter approach is to be followed.

If systems development or systems evolution is placed into the hands of so called end-users, people who are trained in a subject area other than systems development, one has to search for development paradigms, these people can feel home with. Prototyping, i.e. system development by planned trial and error [MITT 85], can definitely be such an approach. However, we have to see how far it can carry and whether developing prototypes in software is reasonably close to crafting solutions in physical construction.

3. A LAYMENS CONSTRUCTION OF PHYSICAL OBJECTS

If physical objects are constructed by laymen, i.e. by their end-users, the developer enters his hobby-room and looks around in search for objects helping to achieve the desired task.

His eye rests on tools as well as on materials. Concerning the tools, he probably possesses a hammer, a saw, screw-drivers, ..., and if he is more affluent even some mechanical devices. The materials at hand will be nails, boards and other objects which have not yet been used or which are no longer in use. Then, he starts his work by choosing the needed materials and the tools appropriate for adapting or combining them.

As for the equivalence to the development of program(ming)-objects, the need for software-development environments, which provide a coherent set of tools is generally recognized by now. The need to provide the programmer or even the layman with good materials to work with - i.e. predefined solutions, adaptable and reusable components) has been recognized only more recently.

The HIBOL-2 environment strives to support this programming by reuse and adaptation. Its top level leads into an archive of semi-finished components of office procedures and of rules for computing data on the basis of filling in business forms. But there remains still the need to construct such semi-finished components in the first place, to adapt and to combine them properly. To describe this, we turn again to our analogy.

Looking at the development process of simple physical objects, the "hobby worker" usually first strives for obtaining an approximate solution. The direction of approximation is obvious. If one has to cut something, one first cuts just a little bit too little off and then fits the piece by successive trial and cutting. Likewise by bending or by other physical and material operations one approaches the final solution via a sequence of successively better approximations where one strives to never cross the point of the exact solutions because oscillations around this solution are hardly possible.

Only with more complex systems, notably with those whose complexity prohibits construction by laymen, an abstract solution, say a blueprint, is made before the physical solution. Designing such an abstraction requires quite different skills than the gutt feeling needed for a fast sequence of trials. The process is also very different. It is not possible to "fit" a design in the way mentioned above. One rather has to aim for an exactly fitting point solution within a single shot. (Only the infeasibility of obtaining such a fit in certain kinds of system- and development environments lead to the recognition of prototyping as a proper way of developing software systems.)

4. THE SPECIAL NATURE OF PROGRAMMING

Programming tasks differ from other system building tasks in so far as they require to develop abstract objects and as they aim at exact solutions in the first shot. This is at least, what trainees are tought in programming courses and this principal characteristic is in no way invalidated by alternative paradigms of software development such as prototyping approaches or the spiral model [BOEH 88]. These approaches are rather alternatives to the abstraction process involved with software design which becomes necessary whence a certain complexity/size treshold gets surpassed.

One should note though that even pure prototyping approaches are feasible only up to a certain size of system/complexity and that beyond this level some kind of (semi-)formal design has to take place. Thus, point-solutions are required at more than one level and the transition from one level to the next requires a change of representation.

All design methodologies proposed in the literature strive for some kind of system decomposition to ease complexity. Their breakdown strategy - how diverse it ever might be - though, decomposes systems invariably into distinct functions or functionalities which are, considered in isolation complete though. Thus, the development problem becomes smaller but remains invariant with respect to the nature of the intellectual task to be mastered by the developer. There is nowhere room for the gradual convergence of solutions as we see them in building simple physical systems.

5. FORMAL VERSUS INFORMAL ASPECTS OF OFFICE PROCEDURES

Office procedures differ quite markedly from the formal and "up to the point" nature of programming and software design. Not that they would lack any formality; but from organizational theory we know very well that organizations can function at their best only when a proper relationship between the formal organization and whatever kind of informal organization(s) exist. Would it be different, "working to rule" (Dienst nach Vorschrift) would not be one of the most frightening methods of striking. The challenges involved with designing office software become evident if we note that working to rule is the only mode of operation for computers though!

The difference between the "mathematical" exactness of programming and the fuzzyness of office procedures can best be highlighted by observing attempts to formalize law [STAM 86], an area of social organization on which the foundation of all modern industrial life is built.

How does one cope with this problem? In systems brought into operation for parametric users, one solves it by leaving them open for manual intervention. In systems which would allow users also the freedom of adaption, enlargement and even (re-)construction, one should strive for dimensions of decomposition in excess of those for functional decomposition (as above, I encompass under this term data-flow, object-oriented, or other related decomposition approaches, because eventually they all boil down to functionality.)

What are those dimensions? We can draw from classical philosophy the distinction between form and contents. While contents (ontologically) is even in most physical design tasks something one has to settle for in a process of well considered choice, form is the dimension where we are accustomed to gradual shaping.

Considering data processing systems, each, form and contents, can be further split up. With form, we can either refer to the visible external appearance, the appeal something has towards the human sensory apparatus, or to the internal or structural form, something which usually remains invisible, which can only appeal to the technician, collector or to other species of special experts.

With contents, we refer usually to the semantical contents of an information system. The bearer of this contents is data (usually with static semantics) and mappings between data (to describe the dynamics of a system).

The following chapter will show, how these considerations of system construction ergonomics are incorporated in the programming language HIBOL-2 and its development environment.

6. HIBOL-2: AN ATTEMPTED ANSWER TO THE CHALLENGES OF OFFICE PROGRAMMING

HIBOL-2 [MITT 87, WERN 87] is a programming language which has been designed in conjunction with its programming environment to allow software development also for people with relatively little training in programming or abstract design methodology.

Its basic paradigm is a desk-top metapher. It is assumed that a clerk fills in business-forms according to prespecified rules or office procedures.

The basic organization of the desk, a device containing slots for incomming data (input-forms), outgoing data (output-forms) as well as slots for accessing and if necessary changing data stored on files (update slots) as well as for performing dialogs with the agent requesting an immediate response service from this desk (dialog slots) are provided. Likewise, there is space on the desk to hold the coarse rules (sequence of action) to be observed when working on this desk, (see Figure 1)

The rough external form of the desk is standardized, since we felt there would be little use in giving users an undue illusion of working on his own wooden desk instead of an electronic abstraction. The details of this arrangement are subject to the users taste though.

The contents of the desk is given by the contents of the forms bound to the respective communication slots as far as the global aspects of data mapping are concerned. At a more detailed level, the working rule determine the sequence in which forms are processed and the level of aggregation, i.e. whether processing should be done on a form by form basis or whether a complete file of forms shold be processed at once.

Defining desks and the work rule is the first portion of work in writing a HIBOL-program. The next sequence of work-steps are constituted by defining the business-forms attached to a desk.

This specification of business-forms is divided into a sequence of phases for each form with each phase corresponding to a specific aspect of form and contents. The phases are:

- phase 1: layout definition,
- " 2: type definition,
- " 3: result definition,
- " 4: optimization.

The tasks to be completed during the individual phases are (see also Figure 2):

Phase 1 - layout definition: During this phase, the layout of the business-form is designed. The "programmer" can organize the area of the form into a hierarchical set of named subareas as if he would draw a form template. The names of the areas should be speaking descriptions of the data held by form instances at the respective location. Likewise, the user can write strings of text onto the form template, which then will appear on each form instance.

By organizing the areas on the form, the user implicitely defines the composite data structure for the respective form. Special provisions are taken for the specification of repetitive data structures.

Phase 2 - type definition: While layout definition provided implicitly the high level data structure, the typing of elementary items has to be done explicitely. The user can either provide a standard type such as INTEGER, DECIMAL,n, TEXT, STRING, DATE, or can indicate a range of admissible values, thus implicitely also indicating a standard type.

With type definition by range specification, the system will automatically provide a range check.

Phase 3 - result definition: While the previous phases had to do with data declarations, the result definition provides the needed procedure specifications. The user can write into the location, where the result should appear either the name of a standard procedure identifyer, such as INPUT or SYSDATE, or an expression dealing with the names of data areas on this or other forms currently accessible on the desk.

The general perspective on form mapping follows a data driven attitude. Hence, HIBOL-2 does not need any sequencing or control driven loop constructs on the forms level. Loops are driven by the (qualifying) instances of input data. The respective language constructs are the EACH- and the FETCH- statement for instance preserving mappings.

SUM, COUNT, COMBINE, ... are aggregating operators working on complete collections of repetitive data structures. With each of these constructs, conditions can be specified to have only those instances which qualify figuring in the respective result.

The individual procedural specifications are given on the level of the respective form item to which they belong. The overall sequencing of the computations following from these specifications is done according to the implicit mapping dependencies by the system.

Phase 4 - optimization: This phase allows to provide the system with hints as to how repetitive data structures should be implemented. In extreme cases, the user might stick in his own implementation, thus replacing the default implementation.

Considering the four phases of specifying a business-form in the light of the discussion concerning form and contents, the following observations can be made:

After the initial sequence of form definition, which has to be done according to ascending phase number, the application developer is free to make modifications to a form in whatever sequence she or he desires. However, one has to observe that layout definition is purely concerned with the external appearance of the form. The fact, that the internal data structure is derived from this external appearance becomes only evident in so far, as it would be more cumbersome to move items across boundary lines of higher level items than to move them within their parent item or to change their external representation.

Type and range definition concerns a deeper level of "form", because it defines not only the space needed by a value (if going to the extreme).. The range a data value is drawn from may have deep implications concerning its semantics.

Result definition is definitely the description of contents. Hence, it should be preplanned and not subject to whatever trial and error procedure.

Optimization is the phase dealing with the internal structure of complex and voluminous data. It is hence the phase which concerns the aspect which might yield delight to the insightfull observer only. To stay consistent with our aims, optimization is, therefore, optional. Only if dictated by efficiency of large applications, the developer should delve into this phase. For small applications, the non-expert builder is well advised to stay with the default internal structures provided by the development environment.

7. SUMMARY

Although the process of gradually developing physical objects cannot be completely carried over to the development of software, the abstract nature of programming can be reduced by splitting form and contents of software solutions.

The programming language HIBOL-2 and its environment can be considered as an example of incorporating this division. It provides an integration of a programming language into its environment in such a way, that (end-)users are to a large degree relieved from abstracting all details into the sequential text of a classical programming languages.

REFERENCES

BOEH 88 BOEHM B. W.: "A Spiral Model of Software Development and Enhancement"; IEEE Computer, Vol. 21/5, May 88, pp. 61 - 72.

BRAC 84 BRACCHI G., PERNICI B.; "The Design Requirements of Office Systems"; acm Trans. on Office Information Systems, Vol. 2/2, Apr. 84, pp. 151 - 170.

FLOY 84 FLOYD Ch.: "A systematic look at prototyping"; in Budde et al. (eds.): "Approaches to Prototyping", Springer-Verlag, 1984, pp. 1 - 18.

HAGG 88 HAGGLUND S.: "Interactive Design and Adaptive Maintenance of Knowledge-Based Office Systems"; Proc. IFIP WG 8.4 Working Conf. "Office Information Systems - The Design Process", Linz, 1988, pp. 1 - 77.

MITT 85 MITTERMEIR R.; "Requirements Elicitation by Rapid Prototyping", Informatica '85, Nova Gorica, 1985, pp. 105 - 108.

MITT 87 MITTERMEIR R., WERNHART H.; "Benutzerhandbuch zur Programmiersprache HIBOL-2"; Institut für Informatik, Universitat Klagenfurt, 1987.

MUMF 81 MUMFORD E.: "Participative Systems Design: Structure and Method", Systems, Objectives, Solutions, Vol. 1/1, 1981, pp. 5 - 19.

STAM 86 STAMPER R.: "The Processing of Business Semantics: Necessity and Tools"; Proc. IFIP TC 2, WG 2.6 Working Conference "Knowledge and Data (DS-2)", Albufeira, Nov. 1986, pp. U1 - U20.

WERN 87 WERNHART H., MITTERMEIR R.: "Benutzerinterface des Editors für HIBOL-2-Programme"; Institut für Informatik, Universitat Klagenfurt, 1987.

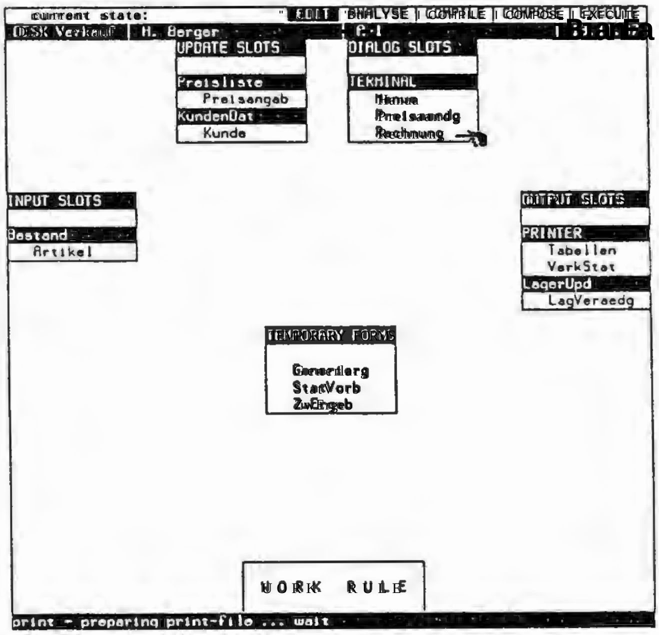


Figure 1: Example of a desk: Bestand, Preisliste, KundenDat etc. are slot names of type INPUT or UPDATE respectively. Artikel, Preisangab, Kunde, etc. are names of business-forms bound to the respective slot.

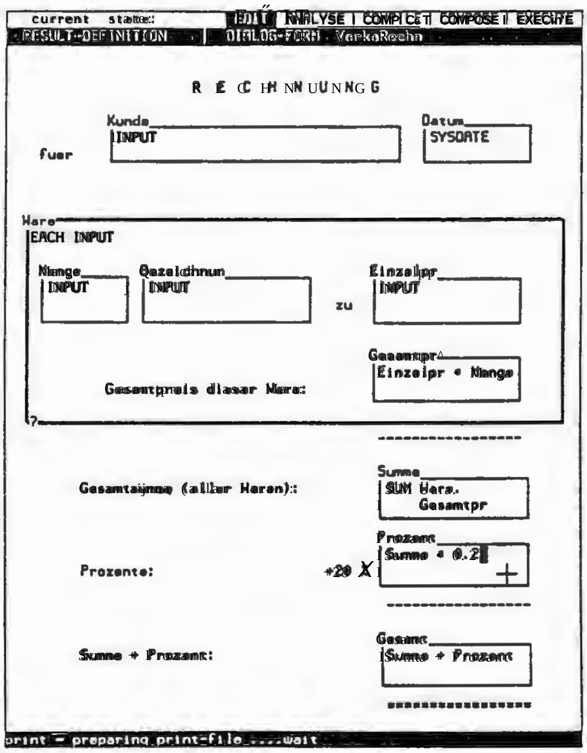


Figure 2: Example of a HIBOL-2 program at the level of a business-form Names on top of the rectangles are identify form elements. The contents of the rectangle define result mappings written into this form during the result definition phase.

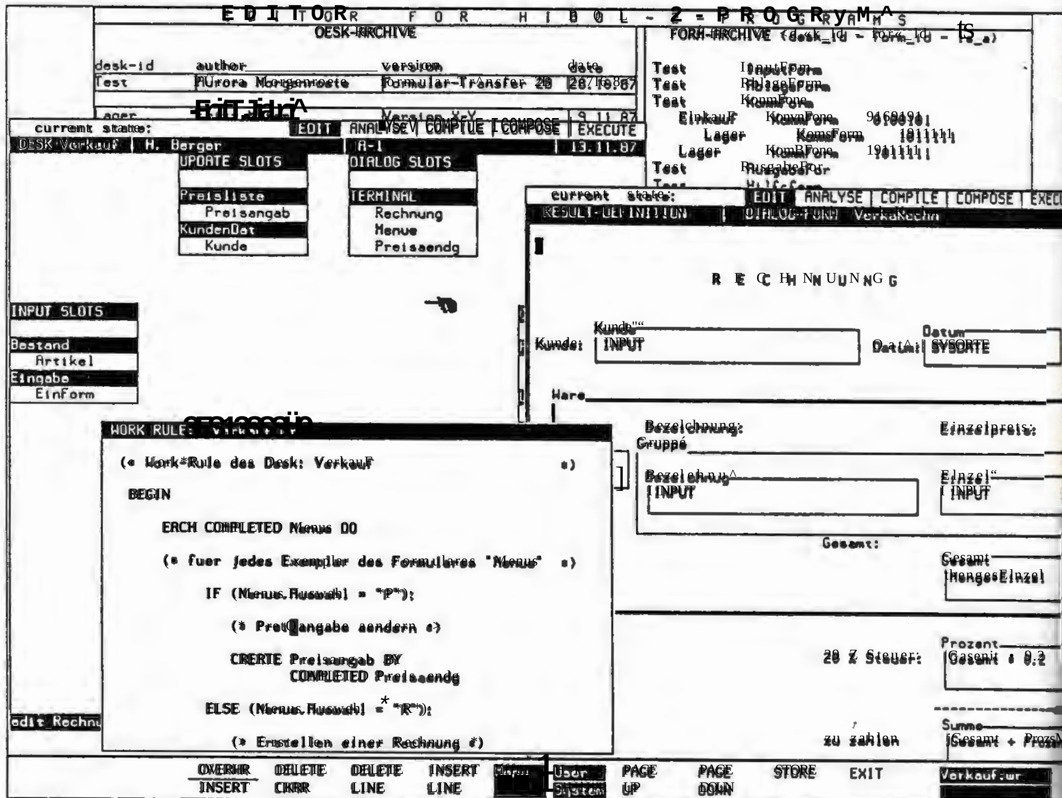


Figure 3: A set of open windows in(to) the HIBOL-2 environment

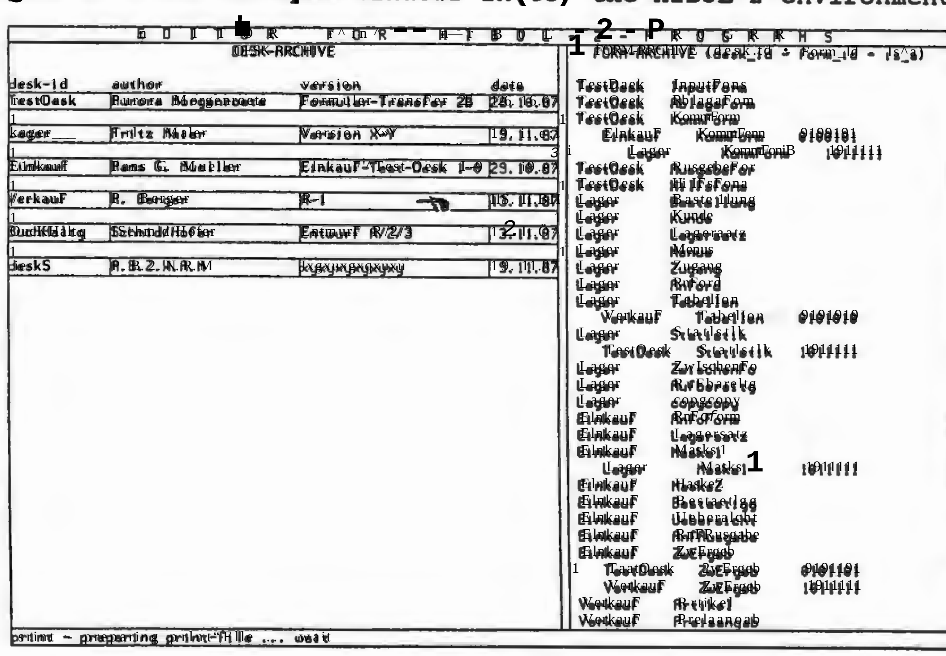


Figure 4: Example view into a desk- and form-archive
The user has verious possibilities to delve deeper and to see more details about the individual entries into this archive

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 81-89.

DEVELOPING USER INTERFACES: A SYNERGETIC APPROACH

G. Haring, F. Penz, M. Tscheligi

Abteilung für Angewandte Informatik
Institut für Statistik und Informatik
Universität Wien

Abstract

The development of man machine interfaces should preferably be based on building prototypes. We introduce an alternative method to create object oriented user interfaces. The various objects are prototyped separately to form independent units of description which demonstrate particular aspects of the entire system. It follows an iterative combination process in which the so far existing prototype units are assembled achieving a more complete description of the intended system.

Keywords: User Interface Design, Prototyping, Object Oriented Design, Software Engineering

1. INTRODUCTION

Due to the technical improvement of hardware (bit mapped displays,pointing devices, graphics processors) the software field is migrating to highly interactive software. Therefore the man machine interface is getting more and more the dominating part of a software system, having great influence on the success of the system. With the growing acceptance of alternative interaction techniques and devices by the user community, almost all companies are developing software with advanced user interfaces.

We are involved in the development of an integrated office automation system. This package will integrate features of word processing, graphics, databases and spreadsheets. The intended user interface is based on the principles of DM (Direct Manipulation, [1]) and the WYSIWYG approach (What You See Is What You Get), known from the Macintosh™ user interface environment:

- continuous representation of the objects of interest
- physical actions (movement and selection by mouse) or labelled buttonpresses instead of complex syntax
- rapid, incremental, reversible operations whose impact on the object of interest is immediately visible
- incremental learning that permits usage with minimal knowledge
- the effect of operations is immediately seen by the user
- the future printout of documents is exactly seen on the screen

This lead to an object orientation of modern interfaces, which are composed of distinct objects presenting the functionality of the application. Our development takes place in the Presentation Manager™ environment, running under the new OS/2™ operating system developed for the powerful 32-bit microprocessors. The Presentation Manager shows explicit object oriented concepts also found in the Smalltalk™ system [2].

Another characteristic of the ongoing project is the separation of the interface design from the application design [3]. This results in the formation of two distinct development groups, reflecting this separation. In the following we only concentrate on the development of the user interface and we are not describing necessary concepts underlying this separation.

The altered profile of interactive software systems has great impact on the life cycle model. The requirements definition and specification phase, in particular for user interfaces, is very hard to do with the historic software engineering methods. It is almost impossible to solve any significant design problem in a single iteration. As an alternative approach prototyping has been introduced by various authors [4,5,6].

Macintosh™ is a trademark of Apple Computer Inc.
Presentation Manager™ is a trademark of Microsoft Corp.
OS/2™ is a trademark of Microsoft Corp.
Smalltalk™ is a trademark of Xerox Corp.

Prototyping can serve various purposes [7]. We are specially interested in the possibility of stabilizing the user requirements for the system and experimenting with new and novel design ideas. The ability to gather "hands-on" experience leads to much more insight into the problem domain, as one can get with paper descriptions alone.

Based on the classification from Weiser in [8] we are building user interface prototypes. Prototyping techniques can be mainly classified into three categories [9]: prototyping for exploration, prototyping for experimentation and prototyping for evolution. As can be seen from the following sections we adopt the experimentation approach to determine the adequacy of a design step. Further we also work in an explorative manner, because the prototype can generate new requirements for the product. The final prototype however need not be thrown away as proposed in most definitions of explorative prototyping.

This paper describes in the following a form of prototyping applied to our specific project demands. To begin with the prototype cycle preliminary functional requirements have to be established. A Synergetic Prototype is then constructed, which finally forms a superior specification of the target user interface.

2. THE SYNERGETIC APPROACH

The synergetic method is comparable to the strategy applied for sticking together the pieces of a puzzle. The player starts with the single parts of the puzzle, showing only a small facet of the entire picture. Figure 1 gives the imagination of such totally unorganized spread pieces.

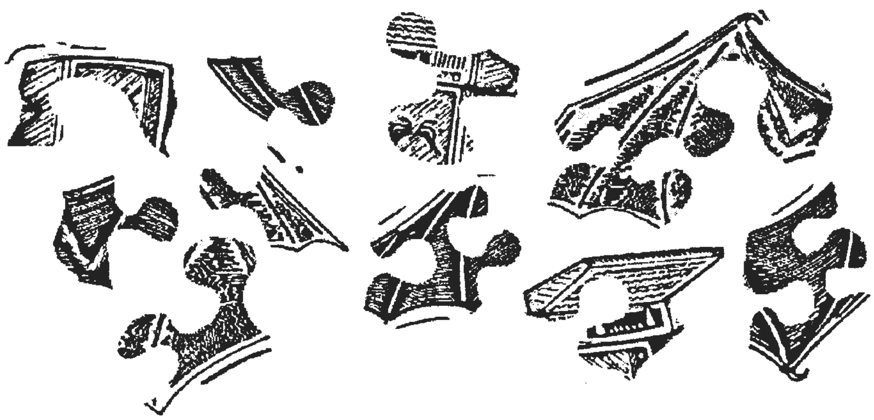


Figure 1

Now the player tries to identify some related pieces to form a more complete picture unit, probably by looking for pieces which show similar colours or shapes. As everyone knows, the dividing line between the units is not very clear determined. The same problem can also appear at the functional decomposition of software systems. In figure 2 we see some assembled picture subpart.

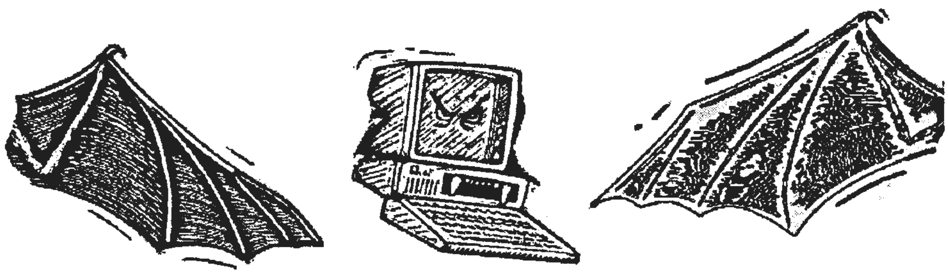


Figure 2

The completion of the picture is done by the successive combination of the so far obtained subparts. This results in a complete picture as shown in figure 3.

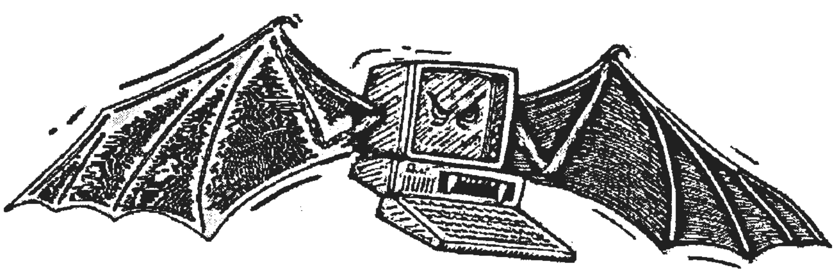


Figure 3 [10]

3. BUILDING UP A SYSTEM MODEL

We use a similar approach for building up the system model as proposed from Budde et al in [11]. The starting point of the user interface development process is the intended functionality of the future product. The first step to obtain this functionality is to determine the so called working objects in the planned user model of the planned application. These working objects are the objects manipulated by the user in his mind. They are reduced to their relevant aspects and then their possible states are identified to form a presentation as elements of the system model.

A typical example for such a working object is a folder in a filing system. The user will imagine a folder as a thing he already knows from his deskwork. A folder is able to hold files which may be documents in the users mental system model. Relevant aspects of a folder are the ability to have a name and to keep the contained documents (files). The state of a folder may be open or closed and is further determined by the contained documents and the folder name.

The functions, the system should perform, are grouped in an object oriented manner with the objects they act on. The operations which alter the attributes have therefore to be identified for each object. The states of an object may only be altered by the operations which are classified to it. The objects forming the users system model should thereby get a functionality which is close to the functionality of the real object in the application domain.

For functions which have a sufficient importance, we introduce so called functional objects, enabling the user an easy access to this functions. An example for such a functional object is a waste-paper-basket which is a presentation for the often needed delete operation. Functional objects can serve as a presentation for an operation (here: delete) applicable to different working objects. The delete operation presented by the waste-paper-basket may for example be valid for a folder as well as for the documents inside the folder.

4. PROTOTYPING WITH THE SYNERGETIC APPROACH

The now obtained system model forms the basis for the synergetic prototyping process. To start the process, the user interface designer has first to produce an object description for every object contained in the users system model planned.

This design process relies on the knowledge and experience of the designer. The screen layouts should be made by a commercial draftsman in cooperation with the interface design group which bears in mind the functional aspects of the system. The commercial draftsman knows how to draw a waste-paper- basket that an user will recognize as a waste paper basket and also introduces aesthetical aspects to the user interface.

Because of the prototyping approach, a detailed elaboration of the object description and a careful adjustment of the interface layout to the system functionality is not as important as it would be within a normal (waterfall) development model. In figure 4 an example of an object description for a folder is given. It is composed of the screen layouts and a short description of the system functionality seen by the user when he is working with the folder.

The object description is used as a supporting document for the initial conversation held between the interface designer and the prototype programmer. It also supports the prototype programmer in his work. In the initial conversation, the interface designer explains the described object to the prototype programmer and so defines his job. Now a well known iterative prototyping process follows. It starts with the realisation of the described objects followed by an interactive evaluation in cooperation between the interface designer and the prototype programmer. With the following correction a new iteration is initiated.

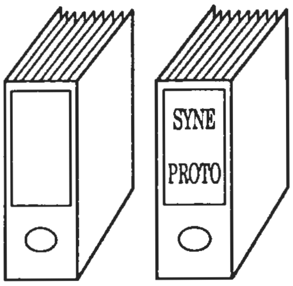
Only the objects and the functions corresponding to a single object are concerned in this first prototyping processes. This functions are named intraobject functions.

When the result of the first prototyping process is satisfying for the interface designer, a prototype exists for each object described. The construction of the synergetic prototype continues by combining the now implemented objects into a new prototype. This combination is done in one or more steps combining objects or groups of objects until a final prototype is achieved.

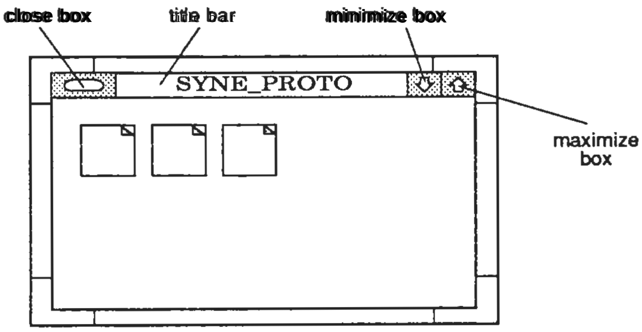
Every now obtained prototype shows more of the functionality which is formed by the cooperation of the assembled prototypes. The functions making up this functionality are named interobject functions. Because of the cooperation of the so far realized prototypes we term the set of produced prototypes a synergetic prototype.

It may happen that an earlier prototype is affected by the correction step. In this case, the affected prototype has to be built again and also the combination of the parts to achieve the latest prototype has to be done again. It is not allowed to alter the presentation or the functionality of an object or a group of objects in a prototyping step later than the step where the functionality was introduced.

Folder closed

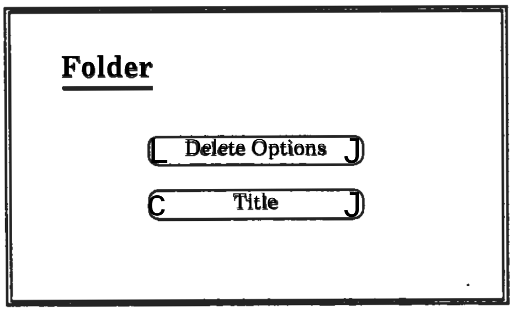


Folder opened



By double clicking on the closed folder the selection menue appears centered to the mouse position.

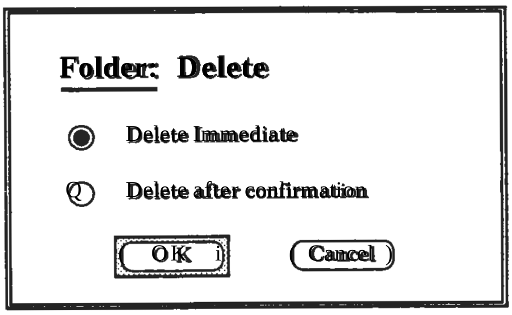
Selection Menue:



If the user clicks the left mouse button:

1) outside the menue box: the selection menue is canceled
2) on the button containing: "Delete Options" the coresponding box appears
3) on the button containing: "Title" the coresponding box appears
4) elsewhere: nothing happens

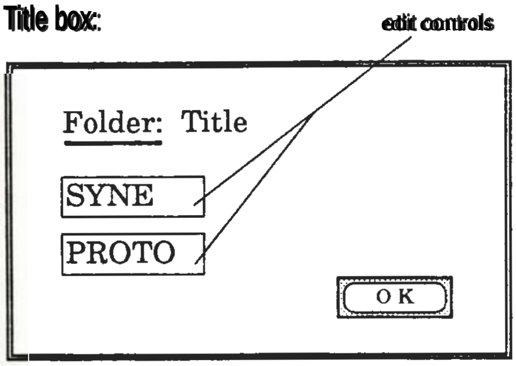
Delete Options Box:



Only one of the two entrys may be selected by a left mouse button click.

Clicking the OK button confirms the adjustment, Cancel reestablishes the state previous to the box activation.

Title box:



The title may be edited in the two edit controls wrapping the cursor from the end of the upper to the beginning of the lower control.

Figure 4

The prototype which shows the functionality of the whole system is called the final prototype. The set of all implemented prototypes is called the synergetic prototype.

5. BUILDING UP AN USER INTERFACE SPECIFICATION

The result we want to obtain by the application of the synergetic prototyping method is a specification of the user interface for the software system in question. This specification is formed by a hierarchical organised set of description units containing the prototypes built.

To get these units of description of the prototyped presentation, every realised prototype is included in a predefined description frame. Every unit of description consists of three parts: The prototype itself which shows the functionality by allowing the reader of the description unit to get a feeling how it works. An explanation text serves as an user manual to explain the functions shown by the prototype which are not shown in any earlier produced description unit. The third part is the programming text which implements the shown and explained functions.

These three parts are maintained by the description frame which also serves as a test environment for the programmer of the contained prototype. The units of description are used as a documentation of the produced prototypes utilized by the designer- and programmer team of every further prototyping step as well as by the implementors of the final product.

The complete set of prototypes, the final synergetic prototype, forms a well structured superior user interface specification. This specification normally will act as milestone preceding the further development steps. An alternative possibility could be the subsequent treatment of the final synergetic prototype to realize the target system. In this case an excessive quality control is necessary to achieve a sufficient system performance and system security. We recommend to apply this quality control not only to the final prototype but also to all single prototypes existing in the synergetic prototype hierarchy. For this task the previous generated description frame can serve as an useful aid.

6. CONCLUSION

We introduced an alternative method for the development of user interfaces based on the concepts of an object oriented software architecture. The construction of our prototyping approach took place according to the features of the underlying user interface environment.

Further research has to be invested in a automatic or semiautomatic support. A Computer Aided Synergetic Prototyping (CASP) could combine several tools. The documentation, realized in our project with the description frame, has to be further improved. The integration of the single prototypes to more expressive ones should also work with a sophisticated automation, minimizing the programming effort.

REFERENCES

1. Shneiderman, B. Direct Manipulation: A Step Beyond Programming Languages. IEEE Computer 16,8 (August 1983), 57-69.
2. Goldberg, A. and Robson, D. Smalltalk-80: The Language and its Implementation. Addison-Wesley, Reading, Mass., 1985.
3. Draper, S. W. and Norman, D. A. Software Engineering for User Interfaces. In proceedings of the 7th International Conference on Software Engineering (Orlando, Fla., Mar. 26-29, 1984). IEEE Computer Society, Los Angeles, Calif., pp. 214-220.
4. Blum, B. I. The Life Cycle - A Debate Over Alternate Models. ACM SIGSOFT Software Engineering Notes 7,4 (October 1982), 18- 20.
5. Gilb, T. Evolutionary Development. ACM SIGSOFT Software Engineering Notes 6,2 (April 1981), 17.
6. McCracken, D. D. and Jackson, M. A. Life-Cycle Concept Considered Harmful. ACM SIGSOFT Software Engineering Notes 6,2 (April 1982), 29-32.
7. Hekmatpour, S. Experience with Evolutionary Prototyping in a Large Software Project. ACM SIGSOFT Software Engineering Notes 12,1 (January 1987), 38-41.
8. Weiser, M. Scale Models and Rapid Prototyping. ACM SIGSOFT SoftwareEngineering Notes 7, 2 (December 1982), 181-185.
9. Floyd, C. A. A Systematic Look at Prototyping. In Budde, R. et al (Eds.), Approaches to Prototyping. Springer Verlag, Berlin, 1984, pp. 1-18.
10. Cowlinshaw, K. C. Computig for the terrified. IBM Perspectives in Computing 6,2 (Fall 1986), 16-19.
11. Budde, R., Kuhlenkamp, K., Sylla, K-H. and Züllinghofen, H. Programmentwicklung mit Smalltalk. In Hoffman, H-J. (Hrsg.), Smalltalk verstehen und anwenden. Hanser-Verlag, München, 1987, pp. 90-121.

BEYOND DATA CRUNCHING:
A NEW APPROACH TO DATABASE INTERACTION

E. Knuth, A.M.Vaina+, Z.Bodó, A. Hernádi

Computer & Automation Institute
Hungarian Academy of Sciences

+ University of Tampere,
Tampere, Finnland

In contrast to traditional data processing ("data crunching"), a number of innovative ideas ('Hyper'-systems, integrated sheets like 'Excel', the whole phenomenon of "MacIntosh", etc.) have emerged lately to provide nonprogramming (generally applicable, turn-key) solutions for office-, management-, and personal information systems. We wish to step further in that logic by narrowing our scope to only one component: the database. A new mechanism is introduced which provides multi-contextual simultaneous data access with inter-session resident context-management and context-associated operation-interface (in contrast to subfunction-oriented language-interfaces of traditional systems).

Keywords: database interaction; context management; multiple views; Hypertext; electronic encycklopedia.

1. INTRODUCTION

Background

We all remember the initial heroic era of computer applications when instruction speed was the only significant parameter to learn, and computers were almost exclusively used for numeric computations: 'to crunch numbers'. Weights have been significantly shifted lately (first of all towards symbolic computations nowadays), as numerous papers in the volume emphasize this. It is not so obvious, however, that though database management is a much younger area of wide applicability, still a similar trend can also be identified here.

Traditional data processing systems (e.g. payroll, banking, insurance, etc.) are characteristically based on batch operations with predefined transactions and frozen schemes. (Such systems can best be realized by *programming* them using DBMS sublanguages. Prominent examples for these aproaches are e.g. [1], [2], [3].) They are born to 'crunch data' - an important thing to do (to free the white collar staff), but the centre of interest in data systems has become radically different in our days.

Forms, accessibility, and exchange of information will play key roles in the society of the future. Trends like hyper media systems, electronic encyclopedia, associative retrieval, distributed knowledge banks, boulletin boards, teletext, etc. will all lead to completely new forms in human working conditions, in cooperative human behaviour, and in information acquisition in general.

To increase the productivity of nonprofessional computer usage, the indefensibility of puzzle solving practice has long been realized [4], [5]. Though recent developments in man-machine communication techniques are impressive (e.g. [6], [7], [9], [10]), it seems that the database community has not yet utilized the power offered in its real nature.

This paper considers one of the fundamental factors common in these trends, the way of *interaction* itself - along the line as emerging new man-machine communication techniques (multi-windows, views, contexts, iconic techniques, the 'MacIntosh phenomenon', etc.) constantly spread. Information in the future (e.g. in public, office, and personal systems) should 'throw itself' at the users, clearly offering all modes of its explorability, in contrast to 'mining in the darkness', (as most language-orinted approaches do).

Phylosophy

Below we give a list of considerations which we believe new approaches should support. Some of them have already been posed by Hyper-text systems too. We claim however to step further by narrowing the scope and dedicating our concepts to the database area.

(1) *Resident working contexts.* When working with a database, several windows can be used at the same time showing different views and or different pieces of information. On logging out, normally all these working contexts are lost and have to be rebuilt when starting work again. The first point is therefore to introduce a kind of a 'Context manager' (with its own resident information base) enabling multi-contextual data-dialogues to survive.

(2) *Making contexts live.* When one of the contexts is used to update the database, it can be useful to see the changes caused on pieces of information displayed in other parallel contexts (in a real time way). So the next point is to make contexts sensitive to changes taking place in the database.

(3) *Reflections versus facts.* Dialog contexts present 'reflections' (from various viewpoints) over the real data (the 'facts'). The world of reflections should obey a selfcontained information model (to be mapped to the database scheme). (The idea is the same as that of 3D geometric modelling.)

(4) *Transformational integrity:* The world of facts should only be available via reflections. Operations acting on reflections may or may not alter the facts themselves (depending on 'modes', see later). As a basic rule, however, nothing may change in the database which is not changing visually on the screen.

(5) *Operational contexts.* Traditionally, operations are associated with database subfunctions (entry, update, query, etc.). They should, however, be associated with reflections in contrast, within a uniform framework providing all subfunctions in one. (In such a way working contexts will not be lost when changing from one database subfunction to another.)

(6) *Nonprogramming paradigm.* The usual method of relying on database sublanguages (database programming) is unacceptable for end-users. The whole interaction should merely be based on templates and icons supposing no syntactic knowledge of the user.

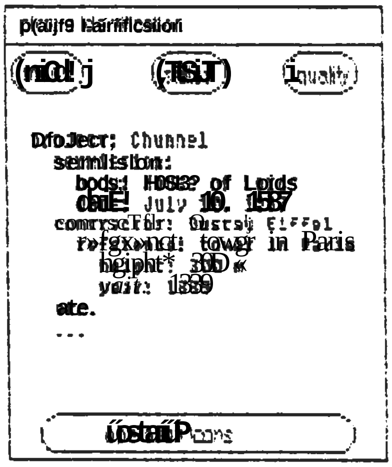
2. SOME NEW CONCEPTS

Here we heuristically outline a set of possible new concepts on which a different type of interaction scheme can be built. For more exact definitions see [8].

Data-picture

Intuitively, a *data-picture* or simply a *picture* is a reflection over the database, to which a meaning is associated. A few examples of possible meanings are as follows: an "entry form for a typical input data", a "transient piece of temporal information on sg.", a "detailed report on sg. to be protected", a "piece of the database scheme from a given context", a "specification of a typical query", etc. All operations to be dicussed later are associated with pictures themselves. (Notice the difference: there will be no operations associated with the special meanings listed above - what conventional systems do.)

Pictures usually show a set of data interconnected in a meaningful sense. They obey a general structural pattern, a hierarchic arrangement (with any number of root nodes) in which each subordination expresses an exisiting interconnection (association). See figure below.



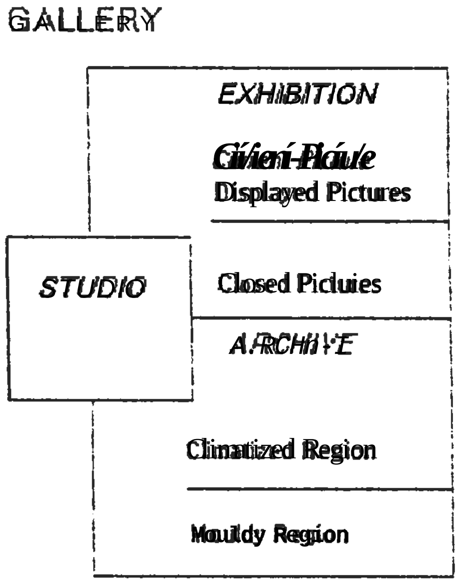
Gallery

As a consequence of our approach, an information base to be managed splits into two separate parts: the database (a conventional part), and the "picture-base" to be referred later as the *Gallery* of pictures. Both have specific integrity rules and properties which have to be maintained partly independently. The Gallery consists of all resident pictures surviving a session (a visit to the Gallery). The Gallery itself is divided into various regions as shown in the figure below.

The so-called *Exhibition* is the region visitors normally enter, where all pictures are thoroughly maintained at all changes of the information base. Pictures of the Exhibition may be *opened* (displayed) or *closed* (not displayed) at a given moment. A distinguished one is the *Current picture*, the one we are just manipulating.

The other major part of the Gallery is called the *Archive* storing pictures which are still valuable in a sense, but which are not maintained regularly any longer. (Typically, it might be e.g. an important long report, which, at a later time, is not necessarily consistent with the actual database.)

The Archive itself is divided into two parts: The *Climatized region* contains pictures which are still in a condition to redisplay them in the Exhibition, on request. (I.e. they are still consistent with all actual integrity properties of the database.) The second, which we call the *Mouldy region* is the store where pictures are abondoned, though not yet discarded. (The restoration of such a picture can be expensive, needing special techniques.)



Finally the *Studio* serves as a restoration area with special equipments (operations), as well as a receiving area to adjust pictures imported from the outside environment.

Qualification

Another dimension of picture classification is the individual qualification associated with each of them. The following classes are considered:

(1) *Sketch* The default qualification for newly created pictures. They are transient ones, and are discarded after the gallery-visit (unless requalified before).

(2) *Property* The normal qualification of resident pictures. Both "sketches" and "properties" are constantly maintained during "performances" (sessions), but neither are protected against "invalidating" them (i.e. data scheme changes contradicting to their structure - see later.)

(3) *Protected* Such pictures cannot be invalidated. Any action (on the active picture) inducing structural changes on any protected one (i.e. alteration of the referred piece of the database schema) is refused.

(4) *Master piece* These pictures are strictly protected. Not only their structure, but even no data value referred by them are alterable. (Creating protected

pictures and master pieces is also a way to localize transformability in large virtual databases.)

In addition, there is a fifth class called the *Standard* one being the set of the "system defined" pictures. This qualification is not available for the users, standard pictures themselves are, however. (These can also be used as typical fragments, "seeds", from which pictures can conveniently be composed by enriching them.)

Status

The following set of properties is directed towards the contents of the pictures displayed. They express important characteristic properties with respect to the mapping which interconnects pictures with the real database.

(1) *Valid.* A picture is called 'valid' if, in all respect, it conformes with the actual database contents. (I.e. its subordinations correspond to existing associations in the database, and all their actual data exist exactly in the referred context. For a formal definition see [8].)

(2) *Filled.* A picture is 'filled' if it does not contain unresolved open references. (I.e. all its references refer to values whenever those exist. See more formally in [8].)

(3) *Saturated.* A picture is 'saturated' if all its nonleaf nodes are referentially complete.

(4) *Evaluated.* A picture is 'evaluated' if all its domain fields are of value type. (Typically, evaluated pictures are results of queries.)

3. OPERATIONS ON PICTURES

There are two universes to be maintained: the "reality" ("facts", i.e. the database itself) and the world of "reflections" we see (i.e. the Pictures). In principle, the "reality" is directly not accessible, but by "reflections" only. Database update is realized by consequences of picture operations exclusively. The world of pictures, however, is also a selfcontained one in a sense. That is, pictures can also be transformed in a number of ways without altering the reflected facts (data) themselves.

Operation modes

serve to designate the type of consequences an operation may cause. Each of the three modes given below can be associated with any of the operations listed (except *status operations*, see later).

(1) *Free mode.* This mode is provided (and can only be used) for temporal transformations. No checks or consequences are made. However, if the validity is not met at last, the resulting modification will not be reaccepted into the Exhibition of Gallery.

(2) *Check mode.* The default mode for all transformations. The validity constraint is checked consistently. An operation is refused whenever contradicts it.

(3) *Enforcing mode.* The <u>only mechanism to update the database</u> itself. It works as follows: Whenever the validity constraint is contradicted, the database is updated so that it will be met again.

We remark that the "enforcing mode" is additionally constrained by the protection of "precious" pictures (see picture qualifications).

Operations on subpictures

This set of operations are combined with a preselection of a subpicture area to act on. The selection procedure, however, is restricted (automatically) so that the selected piece should itself constitute a picture. A stronger version may additionally request that the remaining part must still constitute a picture (i.e. a subhierarchy is extracted from a hierarchy such that the remaining part may not have unlinked branches.)

(1) REMOVE. Removes a selected part from the picture. In the "enforcing mode" the corresponding data objects are also deleted. Notice that it may alter the picture's validity. (Some finer distinctions can also be made concerning the future of the selected piece, see [8].)

(2) MOVE. Moves the selected part a) to another location in the picture, or b) to a location in another picture. Invalidation of the target picture is refused in the "check mode", while validation is enforced by database update in the "enforcing mode".

(3) COPY. Behaves similarly to the above. Naturally, the source picture remains unaltered.

(Remark: there are four additional operations associated with status transitions namely: VALIDATE, FILL, SATURATE, EVALUATE; which also act on subpictures. These, however, are not affected by operation modes.)

Line oriented operations

These operations are combined with the preselection of a particular picture line.

(4) UNFOLD. This operation expounds unshown associations in one step. The selected line is "unfolded" resulting in a new level in the hierarchy. (This operation always preserves the "filled" property. Operation modes do not make differences here.)

(5) NEW. Adding a new picture line to a designated place in a hierarchy. Operation modes act consistently as defined.

Token oriented operations

These operations refer to individual objects contained in picture lines (e.g. like a "value", a "domain", a "relation", etc.).

(6) GIVE. To give a particular instance to the place located. When the place referred to already contains an existing token, it is replaced by the new one (throughout the whole Gallery, depending on the operation mode applied). The new object inherits all the associations its predecessor had possessed.

(7) DROP. To abandon a previously occupied place in the picture. No checking needed in the "check mode". When being in the "enforcing mode", however, the corresponding data is deleted, too.

(8) CLEAR. Clears all data instances from a whole subpicture selected. A skeleton remains (the corresponding part of the scheme). When in the "enforcing mode", all data objects referred to are also deleted.

Organization

Gallery and *Database* are the principal objects visible from outside. The database object itself is accessible as a whole only. In contrast to Galleries, there are no direct operations associated with it. (Visitors are welcomed in Galleries only.) To a given database, any number of Galleries can be assigned satisfying the common demand to ensure personalized views and taylored communication patterns with respect to a common information base.

The *Gallery* object. Whenever a Gallery is opened, its organization is displayed. At this logical level, pictures *as selfcontained objects* are subjects to transactions: create picture, open/close picture, delete/rename picture, requalify picture, move/copy pictures. The target, in case of the last pair, can either be a) a region in the same Gallery, or b) another Gallery.

4. A SIMPLE EXAMPLE

Consider an ultimately simple data model consisting of a set of *atoms* and binary connections over them. An atom is considered as a *(type, value)* pair, while a *connection* as a *relname(atom,atom)* construct, where the participating atoms are unordered. Examples:

atom1: **document** : Declaration of Independence;
atom2: **person** : Thomas Jefferson;

connection: **author** (document ..., person ...);

Relations are defined by factorizations of connections by participating types, (e.g. "author(document,person)" is a relation above where the pair of types is considered again to be unordered). More details on identity and integrity constraints can be found in [8].

Relations or connections will be represented by *identation* in a picture and will be called *'subordinations'*. Notice that the same connection can appear as a subordination in any direction, e.g.:

document: Declaration of Independence
    [author] **person**: Thomas Jefferson

represent exactly the same database contents as

person: Thomas Jefferson
    [author] **document**: Declaration of Independence

Pictures will then be defined as follows. A *picture* is a hierarchy (forest) of *lines*, where

line = [relname], type, domain;
domain= value | expression | empty;

Relation names may appear in nonroot lines only (and can also be the empty string). *Expression* designates a set of values (e.g. a regular expression in the terminology of UNIX), while *empty* represents the universal quantifier in the same sense.

To understand all these properly, remember that pictures are real multi-purpose objects. Entry forms, query specifications, the scheme itself, reports, etc. are all merely "pictures" (that is, the same concept) in this philosophy.

5. EXERCISES

Creating an entry

The most elementary exercise is filing records. One of the possible ways to do it might be as follows:

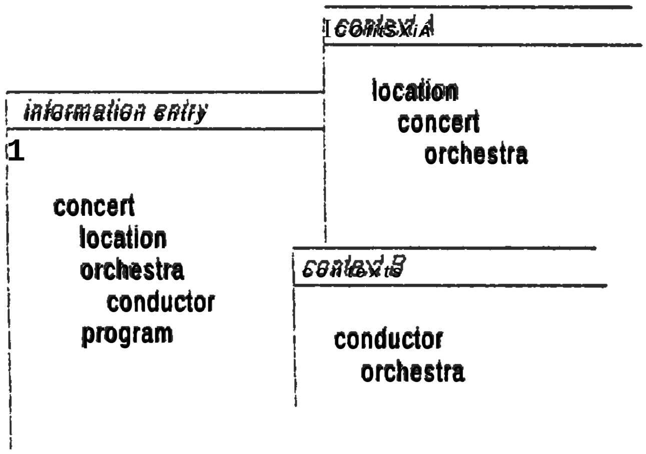
(a) CREATE an Empty Picture;
(b) Enter a prototype record in FREE mode;
(c) VALIDATE;

(Notice that by this step;
(1) The system learns the data scheme;
(2) The actual data is stored too.)

(d) CLEAR; (A skeleton remains now in the picture)
(e) SAVE it as an entry form.

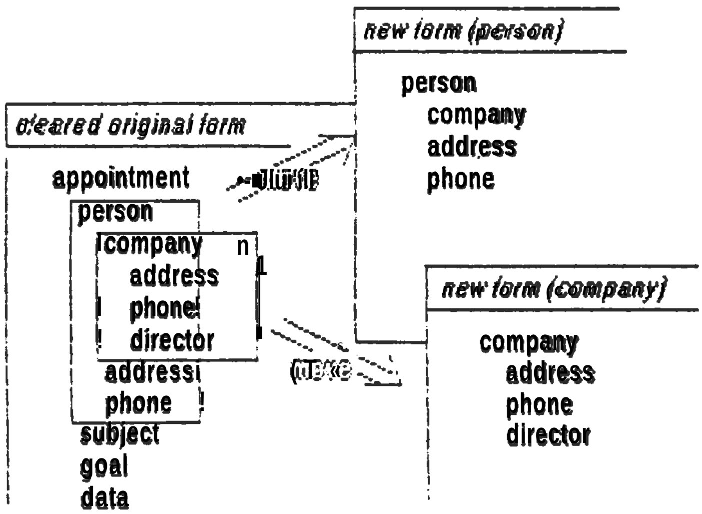
Live contexts

The open pictures of the Exhibition (preferably as selfcontained windows on the screen) live parallel. In principle, they all are refreshed continuously. Therefore, we may watch the consequences we cause (by operating on the active picture) from several inverted contexts. The following figure shows an example for such an arrangement.



Transfer operations

Typically, pieces of information recorded in ad-hoc situations might not be structured consistently enough. For instance, after a particular negotiation, we may record something which looks like the leftmost picture in the figure below. Later, we would hardly need it as a typical entry form with exactly the same pattern. Therefore, it is better to recompose it into three separate forms for future use, as shown.



Moving or copying parts of pictures are also important in the opposite direction, i.e. *constructing* complex pictures *by reusing* typical pieces from others. Think, for instance, of query programming. Specifications of complicated queries are to be stored normally by well identified pictures. At a time, we may need to join them in an unusual way. Such queries can easily be contructed by appropriate subpicture transfers.

Using available tokens

Nothing is more disappointing than failing to identify some existing tokens by mistyping them. Therefore, it is essential to provide a full service to choose available tokens in all possible operational contexts.

Menus for "available tokens" based on the actual database contents are offered for values, types, and relations equally. These menus are always customized for the actual situation they are called. (I.e. using available tokens never invalidates a picture.) (E.g. a simple use is having a field for some keywords belonging to a relatively small but possibly changing set. Typically, we never want to retype (already used) keywords again.)

5. EXTENSIONS

Other data models

The idea of 'Data-picture' relies on the uniformization of interaction patterns corresponding to all possible database subfunctions. Remember that within the definition of 'Picture' we vaguely referred to 'associations' which connect items. Since there was only one kind of association introduced in our simple exaple data model, everything was very straightforward. Naturally, the richer the data model is semantically, the harder it is to find such a uniform representation. Data elements within complex data models can be interconnected in a number of different ways (inheritance, membership, instantiation, attribute, argument, version, etc.).

We think, however, that this kind of ultimate uniformity, though elegant, is not the most important point. If necessary, subversions of pictures covering different functionality may well be introduced. What is really important, is the live management of surviving multiple contexts as listed in the requirements. These requirements are independent of data models.

Concerning the range of data models to be applied in the outlined framework, we concluded the following:

a) The example model contained in section 4. together with slight enrichments (e.g. enriching the available choice of kinds of relations) can well serve for office applications (especially for the non-programmer community).

b) Classical data models (like network, relational) might first of all be considered as programmers paradigms. Such data models do not serve immediately for end-user access. Their user interfaces are normally established by programming methods to satisfy specialized functions. Therefore, these models serve first of all for 'data crunching' problems rather than providing direct interaction.

c) Finally an exciting class called *object oriented data models* (see e.g. [11], [12], [13]) can be mentioned. These models are apriory dedicated to sophisticated users providing exceptionally rich and effective means to handle complex and abstract information structures. Since the full comprehension is essential when working with such databases, a Gallery-like interaction mechanism seems to be an important challenge to realize. We think that finding such a technique might constitute a possible high end within the research we are carrying out with respect to database interaction. A number of open questions are however posed by such an approach.

More advanced technical means

All the techniques mentioned so far can be realized using only textual lines with constant scaling. Since most personal computers go no further in the matter, it was important to fix what the new approach can reach at this level of technical tools. There are, however, a number of future possibilities. A few of them are as follows.

a) Continuous zooming. Providing a full structural view scaling down everything to any appropriate level (even unreadable!) with the capability of continoucs blow up for any desired point of information.

b) Real-time magnifier. A magnifier of adjustable size and enlargement ratio which can move across a picture while blowing up details in a real-time way.

c) Movable unfolding. A temporal version of 'unfold' while dragging down, turning to resident only when releasing the mouse.

d) Real-time inversion. Showing prespecified inverted contexts in a running fashion while dragging down a master picture.

Naturally, all these can be combined with graphic representations of logical interactions.

REFERENCES

1. dBASE III User Manual. Copyright Ashton-Tate, 1984.

2. INFORMIX. Registered trademark of Relational Database Systems, Inc. USA.

3. Date, C.J. A Guide to INGRES. Addison-Wesley, 1987, p.386.

4. Cawkell, A.E. (ed.) Information Technology and Office Systems. North Holland, 1986.

5. Pfaff, G.E. (ed.) User Interface Management Systems. Springer Verlag, 1985.

6. Hopgood, F.R. et al. (ed.) Methodology of Window Management. Springer Verlag, 1986.

7. SUN Microsystems Inc., Programmer's Reference Manual for the SUN Window System. SUN Microsystems Inc., 2550 Garcia Avenue, Mountain View, CA94043.

8. Bodo, Z., Hermadi, A., Knuth, E. "coDE" Common Base Definition. Reference Manual. Version 7. Computer and Automation Institute, Hungarian Academy of Sciences. Budapest, 1988.

9. Inside Macintosh. Vol. I-III. Addison-Wesley, 1986.

10. GEM Programmer's Toolkit. Digital Research Inc.

11. Beech, D. A Foundation for Evolution from Relational to Object Databases. Hewlett-Packard Laboratories, Palo Alto, 1988.

12. Fishman, D.H. et al. Iris: An Object-Oriented Database Management System. ACM Transactions on Office Information Systems. 5,1, pp.48-69, 1987.

13. Maier, D. et al. Development of an Object-Oriented DBMS. Proc. OOPSLA, Portland, Oregon, 1986.

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 105-112.

COMPUTERS AS SCIENTIFIC CO-WORKERS : SCOPE AND LIMITS

Bertram Malle, Günter Schulter

Institut für Psychologie
Karl Franzens-Universität Graz
Schubertstraße 6a/II
8010 GRAZ

Abstract

Development of AI technology has brought about several tools that will soon change a scientist's work. These tools not only help to meet burdensome tasks but also begin to influence the process of scientific discovery. Specialized expert systems will be designed which should represent knowledge that is procured by the exponentially increasing amount of published scientific work and which should make inferences from this knowledge, thus generating new hypotheses and offering interpretations of given research data.
An outline of possible interactions between experimental scientists and specially designed computer systems is given. Desiderata of scientific expert systems (SES) are presented and an example of recently developed SES is introduced.
The use of computer systems in the process of research will have diverse consequences, both improvements and dangers. Some of these consequences for university education and for the structure of research are sketched.
Finally, some philosophical considerations with regard to limits of intelligent computer systems are presented.

1. The Scientist and his Computer : Options of possible support

In the course of research computers have become irreplaceable tools. The main tasks an experimental scientist is confronted with will be outlined in the present section of this report. The focus will be centered on already available and on possible new ways of support by adequate computer systems.
There are 6 main steps of work for an experimental researcher:

(1) Stating the subject of investigation
The primary paradigm of expert systems used in support of scientific investigations is to create new hypotheses and to point out inconsistent research data. A first step to develop such a device was made by SCHULTER, KNOCH & MALLE (1988). This program package was named "Scientific Consultant" (SC) and it will be discussed in Section 2.

(2) Information retrieval
The exponential increase of published scientific work calls for specially designed information retrieval systems, the prerequisite of which are user-friendliness, efficacy and direct access to relevant information. V.HAASE and co-workers have recently

started a project on such a tool named "Smart Assistant for Information Retrieval" (SAFIR).

(3) Reading and understanding the essential research results

Accumulation of theoretical and empirical knowledge should be structured and organized by dynamic knowledge base systems that are continuously updated by several scientists working in similar fields. At present most data bases in use are privately organized, thus lacking the possibility of knowledge exchange. The process of filtering out and understanding the relevance of information can be improved by computer programs carrying out so-called meta-analyses on the vast amount of statistical data that usually emerge from empirical research. Expert systems - such as SC - support the understanding of data by providing implications out of a pool of complex information.

(4) Planning and carrying out an experiment

Although it was recently pointed out by H.A.SIMON* that expert systems will soon be able to plan even crucial experiments much work is yet to be done to achieve such a goal. And it may probably transgress the limits of Psychology or Sociology. Finding an adequate method of measurement for a variable of interest and controlling as many other variables as possible are creative processes and - up to now - they cannot be replaced by inference from methodological postulations and other well-defined premises. The problem of creativity will be elaborated in the last section.

Aspects of statistical analysis shall be omitted here as they are not in the centre of the present conference.

(5) Understanding new research results and writing publications

Experimental researchers find it often impossible to develop sufficient integrative understanding of new results in the context of those already known from before and to interprete them from the viewpoint of a given theory. However, comprehensive expert systems will be able to specify the contextual meaning of a particular result.

Today, the most usual way to write a scientific paper would be to use one of the many word processors that are available on the market. In the future more specific software will offer interactive text processing including text modules that can be accessed like a questionnaire. They will produce standardized and hence comparable research reports helping to build up huge and internationally accessible knowledge bases. This way of publication processing, however, should not replace but support methods used at present.

The essential components of a program network that can meet the needs outlined above are **'knowledge representation'** and **'inference'**. Each of these five steps implies one or both of the latter components and both of them determine the scope of computer-aided scientific work, so a brief discussion of them should complete this first section.

In man, knowledge seems to be represented in mainly four ways (cf. SYDOW, 1983):

1. Sensations
2. Concepts (and their relations)
3. Schemata (scripts, frames)
4. Rules

---

* In a paper presented in Graz in May, 1988

One of the problems of storing information lies in the way knowledge must be represented to guarantuee a maximum of associations within and between the different areas aiming at the achievement of fast and effective access. Interdisciplinary research by cognitive psychologists and information scientists could, yet, help to cope with this problem. Another facet is the blindness of today's computers with regard to sensational knowledge as it is continuously acquired by humans. This leads to the so-called "frame problem", i.e. the impossibility of computers to use common sense knowledge such as human brains do. Contemporary expert systems are not affected by this problem because they are designed for strictly defined tasks. The more global the demands (e.g. planning a complex field study into the ways in which man try to get in touch with each other) the more urgent the problem becomes. However, it would be a vain attempt to try to teach computers what man can already do. It would - on the contrary - be more fruitful to compose networks of human and mechanical expertise which work together in a symbiotic way and where each can do what they do best.

More than with knowledge representation, one is meeting difficulties when it comes to understanding the second essential component, i.e. inference. Todays knowledge of the human ability to make inferences is still limited. Since the early sixties the field of theorem proving by machines has been widely explored (RATZEK,1985; BIBEL,1982,1984). Although deductive reasoning is carried out fairly by computers and research has moved to other fields like multivalued logic (YAGER, 1985), Bayesian statistics (KYBURG,1987) or fuzzy logic (BANDLER & KOHOUT,1985), many problems remain. One of them is the "frame problem", i.e. questionning inference from a limited number of premises without the use of common sense knowledge, the latter being one of the human tools to which the scientist may quite often have to take refuge to. Another aspect is the fact that the process of scientific discovery is not very often the product of deduction from well-definded premises. There is much inductive as well as intuitive reasoning and there are complex approaches by means of trial and error strategies. Finally, emotional, motivational and personality factors must be taken into account. They lie exclusively behind human inferences and they may have great significance for innovation and creativity. But again it must be stated that computers should not be designed as bad imitations of the human mind. The advantage of computers lies in the fact that they are free of prejudice; they are unbiased and they are not very likely to get confused (BAUMGART,1985). Moreover, they have enough capacity to cope with large amounts of premises in a straightforward way. These advantages should be used extensively to support and to complete the aptitudes of the human mind.

2. Scientific Expert Systems: Some desiderata and an example

Perfect scientific expert systems (SES) should cooperate with the scientist at each of the five steps of work mentioned above. To design such a system an epistemology of scientific research would be a helpful prerequisite. It would, therefore, have to answer the following questions:

(1) Which specific knowledge is needed ?
(2) How is the latter represented ?
(3) What are justified structures of inference from this knowledge?

Such an epistemology is not yet procurable. However, a list of desired properties of SES will be introduced below. The first of these properties can be referred to as **'dynamic design'**. All required knowledge must be updated continuously and thus become th background of a dynamic knowledge base. The second property can b called **'cooperative design'**. Not only one single scientist o team, but many teams from a multitude of universities an countries should work together building a wide-ranging knowledg base. The third property is **'public accessibilIty'**. As many scien tists as possible should be able to use SES. The next three pro perties are concerned with the question of knowledge representa tion. Representation must be **standardized** in order to mak possible cooperative design and public accessibility. At the sam time it must be **domain-specific** in order to minimize the loss o information caused by the homogenous way of representation. Th sixth property can be named **'multiple area representation'**. A mentioned before, human knowledge is stored in four main ways Successful SES should at least include three of them: concepts an their relations, schemata, i.e. groups of coherent or coincidin elements and rules. This leads to the seventh property: efficient SES must include **justified rules of inference**, both deductive an inductive, to exhaust the implications of available knowledge.

Some of these desiderata are, at least, partly met by the expert system SC mentioned in Section 1 (SCHULTER, KNOCH & MAILE, 1988) SC consists of three subsystems which will be shortly described.

The first, **THESAURUS**, is an hierarchically structured network o scientific concepts, logically connected to each other by the re lations 'broad term', 'narrow term' and 'related term'.

The second subsystem, **FRAMES**, is a network of scientific concept each of which is surrounded by a frame of theoretical and empiri cal information concerning this particular concept. In the frame central concept is connected to other concepts in various theore tical and empirical ways. Each of these connected concepts are re presented in different SLOTS. All information of one Slot i structured by storing it in different FACETS (e.g. GENERAL LITERATURE or GENERAL-COMMENT). The essential information is re presented in a largely standardized syntax of a Facet calle ASSERTION-INFO. These standardized formulas are called 'Asser tions'. They consist of a few variables connected to each other b different functional phrases: VAR1 CAUSES VAR2 / VAR1 EXPLAI VAR2 / GROUP1 SHOW HIGHER VAR1 THAN GROUP2 / VAR1 IS POSITIVEL CORRELATED WITH VAR2 etc. Each variable or group defines a concep which, on the one hand, creates a Frame and, on the other hand, i a Slot in various other Frames. Thus it was tried to design representation structure that is comparable to the overlapping an associative schemata in human memory.

The third subsystem, **EXPERT**, is the inference system of SC. I consists of a number of rules that should generate new Assertion hypotheses and speculations on the basis of some or all Assertio stored in FRAMES. There are four types of rules: 1) **Formulati rules**, which syntactically transform given Assertions in order t make them available to other rules (e.g. IF (VAR1 IS POS CORREL

TED WITH VAR3) THEN (VAR2 IS POS CORRELATED WITH VAR1)). 2) **Deductive Rules**, which are subject to the laws of classical logic, but they may produce virtual new information (e.g. IF (VAR1 EXPLAINS VAR2) AND IF (VAR2 EXPLAINS VAR3) THEN (VAR 1 EXPLAINS VAR3)). 3) **Hypothetical-inductive rules**, which generate conclusions the truth of which is not guarantueed, even though the premises were true. These conclusions are named 'Hypotheses' and they are labeled by the qualification 'probably' in order to show their limited validity (e.g. IF (VAR1 IS POS CORRELATED WITH VAR2) AND IF (VAR2 IS POS CORRELATED WITH VAR3) THEN (VAR1 IS PROBABLY POS CORRELATED WITH VAR3)). 4) **Speculative-inductive rules**, which generate conclusions that are derived from premises including one Hypothesis. Again these conclusions are of limited validity and their lable is the qualification 'possibly' (e.g. IF (VAR1 IS POS CORRELATED WITH VAR2) AND IF (VAR2 IS PROBABLY POS CORRELATED WITH VAR3) THEN (VAR1 IS POSSIBLY POS CORRELATED WITH VAR3)).

SC is a system shell that can be filled up with knowledge from various areas and which is going to meet most of the desiderata explicated above. The problem of a standardized syntax that includes quantification for statistical data and the problem of both exact and creative rules, however, are not adaquately solved yet.

3. Educational and Scientific Consequences

Scientific expert systems as discussed by now are not predominantly user-oriented because they are primarily used by the designers themselves. If we do want to exploit the educational potential of expert systems we must - sooner or later - transform them into intelligent tutoring systems or computer-based learning environments (YAZDANI & LAWLER, 1985). And if we could manage to make university students familiar with specialized computer systems there will be a handful of positive consequences. With the help of SES novices could be introduced to the conceptual network of a discipline; they would become acquainted with brand-new research results; they could find out about connections between data they would otherwise have learned and stored unrelated to each other; they would get to know ways of evidential reasoning and they could recrute hypotheses generated by the system as topics for a master thesis or doctoral dissertation. But there are also doubts as to extremes of studying with a computer. Detailed reading of papers or books could be substituted by the mere stockpiling of abstracts and assertions; instead of following complex lines of argumentation students could simply ask the system about the probability of an assertion; having hypotheses generated could be prefered to the perceiving of the natural environment; and severe logical inferences could harm the spirit for intuitive and creative reasoning. By now, computers learn from being told and humans learn from experience (YAZDANI & LAWLER, 1985) - maybe soon computers will also learn from experience, but humans should not cease to do that.

The consequences of scientific expert systems for the community of science and the research process are diverse. Scientists will get rid of time-consuming work like searching for litterature or putting in order the information one has collected. There will be more time for the understanding of research data, and for the invention of theoretical models, as well as for the interpretation of the newly acquired data. One of the essential prerequisites will be to interest students or other persons to continuously up-

date the knowledge base of the expert system. Otherwise time will be wasted without much benefit for the creation of new ideas. One of the primary tasks will be to minimize the time that is required for the update of the knowledge base itself. On the other hand only the updated system will guarantuee maximum efficacy.
The community of science may soon be faced with the problem of a **scientific Third World**. At the moment only few nations like the USA, Japan, France or England represent the First World, whereas all other nations can be summed up in the Second World. But for economical and political reasons the First World and several nations of the Second World will be able to procure efficient ex- pert systems, perhaps even by working together in fruitful coop- eration, whereas the rest will soon become the Third World, cut off from the flow of current information and inhibited by old fashioned and slow methods of research. On the other hand, inter- national cooperation with consecutively increased productivity is well within the reach. But this leads to a problem for the research process itself which will be named **'scientific infla- tion'**. The better the expert systems become the more publications scientists will be able to produce. But these publications will mainly represent reviews or piecemeal research reports. This ma host of publications will be fed back into the expert system again and the rate of inflation will rise more and more, thu causing a vicous circle. Therefore the aim must be to design ex- pert systems that increase **quality** instead of mere quantity. Thus SES should not be dedicated blindly to the generation of mo hypotheses and topics, but they should be able to depict essentia topics that are worthy of being investigated.

4. Limits of Artificial Intelligence: Some Philosophical Considerations

As to the power of inference the limits of computers are of philosophical relevance, since they depend fully on the extent knowledge and the rules that researchers may accumulate. Expe systems will soon be able to handle a multitude of data and th will extract relevant implications and derive generalizations fr the latter. Although the outcome does not exceed the potential the data and rules determined by the human mind it is justifiabl to use the term 'intelligent' for this process. But as to t question whether states of consciousness can be developed in computer system or that consciousness is no more than a comm sensical word for a special kind of biomechanical processes th are **quite similiar** to those of future-generation computers serious doubts must be raised. One essential point is that inte ligent behaviour is only **one** component of the complexity of ment systems. Another crucial point is that it is possibly just amon the most basic prerequisites of such a system. Even in the m complex way intelligent behaviour taught to computers is no m than rule-following, determined by explicable premises. But wh sort of rule do we follow when we fall in love with another per or when we stop carrying out a task simply because we are not the mood for it anymore ? And how should we program computers stop solving a problem whenever it does not "feel" like continui ? Motivation, mood and emotion are essential components of cr tive behaviour. And creativity is as undetermined as, for examp emotions are: both are transcending the system's prior programm (MORRIS & JOHNSON, 1985). This unpredictability in human think and acting can be illustrated by the following argument. Us language implies to subject oneself to some standards of c sistency, thus defining a **form** of perception and communication restricting the freedom of perception and thinking (MORRIS & JOHNSON, 1985). Computers always use some kind of restrictive language; however, if one does not presuppose that human thinking is simply a biological process in a strictly defined biological language, some parts of conscious experience must be regarded as amorphous and unrestricted. Such conscious states of, for example, daydreaming or imaginative associating are very often first elements of a creative idea. To look at a problem not only in the traditional and obvious way but, on the contrary, to transcend common frames of perception and thought and to consecutively create a new perspective seems to be an exclusively human potential.

Nevertheless, information scientists and cognitive psychologists should cooperate in order to design systems that are as creative as possible. The Psychology of problem solving and creativity may provide useful insights into the processes that are the basis of human thinking. Examples of such strategies are those of NEBER (1987) and PERKINS (1981). The former surveys 40 years of psychological research and discusses the problems of instruction and prevention in the field of problem solving. PERKINS (1981) wrote a witty book containing "tales" about and research into human creativity.

As far as scientific research requires creative, i.e. unrestricted and indeterminate ways of thinking computers may only be co-workers of human scientists. But they should become the best co-workers we ever have had.

REFERENCES

SANDLER,W. & KÖHOUT,L.J.(1985) Probabilistic versus fuzzy production rules in expert systems. International Journal of Man-Machine Studies, 22/3, 347-355.

BAUMGART,M.(1985) Dem Computer fehlt der "Hausverstand". ibf-report vom 18.Október 1985.

BIBEL,W.(1982) Deduktionsverfahren. In W.Bibel & H.Siekmann (Ed.): Proceedings der Frühjahrsschule Künstliche Intelligenz 1982. Fachberichte Informatik 59, Springer: Berlin, 99-140.

BIBEL,W.(1984) Automatische Inferenz. In J.Retti et al.(Ed.): Artificial Intelligence - Eine Einführung. Leitfäden der angewandten Informatik, B.G.Teubner: Stuttgart, 145-167.

KYBURG,H.E.,Jr. Bayesian and non-Bayesian evidential updating. Artificial Intelligence, 31/3, 271-294.

MORRIS,J.M. & JOHNSON,D.P.(1985) Thinking Machines and Creativity. The Journal of Creative Behavior, 19/4, 241-255.

NEBER,H.(1987) Angewandte Problemlösepsychologie, Aschendorff: Münster.

PERKINS,D.N.(1981) The mind's best work. Harvard University Press: Cambridge,Mass.

RATZEK,W.(1985) Domänen der Künstlichen Intelligenz. Nachrichten für Dokumentation, 36/4-5, 210-211.

SCHULTER,G., KNOCH,U. & MALLE,B.(1988) Wissensdateien und Inferenzsysteme als Hilfsmittel wissenschaftlicher Lehre und Forschung: Entwicklung eines Expertensystems. Berichte aus dem Institut für psychologie der Karl-Franzens-Universität Graz, 1988/6.

SYDOW,H.(1982) Systeme der Künstlichen Intelligenz und kognitive Psychologie. Zeitschrift für Psychologie, 190/4, 392-404.

YAGER,R.R.(1985) Infernece in a multivalued logic system. International Journal of Man-Machine Studies, 23/1, 27-45.

YAZDANI,M. & LAWLER,R.W.(1986) Artificial Intelligence and Education: An Overview. Instructional Science, 14, 197-206.

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 113-118.

A FIRST ORDER THEORY OF THE LEARNABLE AND KNOWLEDGE COMMUNICATION

A. BENCZÚR

Eötvös Loránd University, Computer Centre
Budapest

Keywords. Machine Learning, Symbolic Logic, Complexity Theory, Relational Databases.

Abstract. The paper is concerned with an extension of L.G. Valiant's approach to learning Boolean functions for learning finite models. A general learnability theorem is given based on Kolmogorov's algorithmic information quantity. We describe a cooperative learning with a special knowledge communication.

1. INTRODUCTION

The seminal work of L.G.Valiant (1984) inspired a series of papers treating the computational aspects of learning Boolean functions. In this paper we extend Valiant's theory from Boolean functions to a theoretical framework of learning finite models of a first order language. In doing so, we arrive to a problem similar to that of extracting information from a relational database. Then we can employ the performance evaluation techniques of relational databases based on Kolmogorov's algorithmic information quantity. (A.Benczúr (1987)(1988)) As a starting point, consider the problem of learning Boolean functions. We can turn this into a problem of learning finite models over a two-element universe as follows. Let $f(p_1, \ldots, p_n)$ be a Boolean function and $P(x_1, \ldots, x_n)$ an arbitrary n-place predicate symbol. We assign to f the following first order formula

$$\phi(x_0, x_1) = \prod_{\{0,1\}} \tilde{P}(x_{i_1}, \ldots, x_{i_n})$$

where

$$\tilde{P}(x_{i_1}, \ldots, x_{i_n}) = \begin{cases} P(x_{i_1}, \ldots, x_{i_n}) & \text{iff} \quad f(i_1, \ldots, i_n) = 1 \\ \neg P(x_{i_1}, \ldots, x_{i_n}) & \text{iff} \quad f(i_1, \ldots, i_n) = 0 \end{cases}$$

and $\prod$ stands for conjunction and ranges over all the 2^n such assignments. Suppose now that a model M statisfies $\exists(x_0, x_1)\, \phi(x_0, x_1)$. Then, unless f is a constant function, the universe U of M contains two different elements serving as "0" and "1" in the case of the Boolean function f. The formulas ϕ records completely the behaviour of P on these two elements.

This property of $\phi(x_0, x_1)$ can be extended to the case when we have more than 2 variables. These formulas will be called diagram formulas since they come from the method of diagrams of model theory.

The goal of learning a finite model M of a language L over a universe U is to deduce a program that recognizes whether a formula $\psi(x_1,\dots,x_n)$ of L with free variables $x_1,\dots,x_n$ is existentially satisfiable in M or not. The diagram formula of M shows that it is possible in a finite amount of time: one possible way of learning M is to recognize its diagram formula. To formalize the learning mashine we suppose, that the model M is inside a black box and an ORACLE can answer questions which are sentences from L. The learning machine derives diagram formulas statisfiable in M.

To give a possibility to measure the work of the ORACLE, we can suppose that the model M is represented in the black box as a relational database. Then the performance evaluation model of relational databases introduced in Benczúr (1987), (1988) can be applied.

The main idea of the above papers is to measure the information quantity of relational databases by Kolmogorov's algorithmic information quantity. The same measure can be used for diagram formulas and finite models as well. By the help of this measure we can formalize the intuitively expectable theorem of learnability: the lover bound of the algorithmic cost of learning finite models is proportional to the algorithmic information quantity of the models.

A class of finite models is learnable in the sense of Valiant's learnability if there exists a learning algorithm running in time polynomial in the arities of the predicates to be learnt.

Finding learnable models is a central problem of the theory of learnable. Complex models are practically not learnable, only simple submodels of them can be efficiently learnt. These submodels can serve as a reference knowledge for subsequent, partial lear ning procedure or in cooperative learning.

2. DIAGRAM FORMULAS

Let L be a first order language with predicate symbols $P_1,\dots,P_N$, and without constant, function symbols and equality relation. U is a finite and unknown universe to be learnt using the language L. The interpretation of L in U constitutes a model M. The goal of the learning procedure is to construct an algorithm, that can evaluate a sentence ϕ of L according to the interpretation of ϕ in M.

A substep, or an intermediate goal in the learning process is to recognize a diagram formula, satisfiable in M.

Definition 1.
A diagram formula of the language L with the n variables $x_1,\dots,x_n$ is a conjunction of atomic expression $\tilde{P}_i(v_1^i,\dots,v_{k_i}^i)$ for all possible assignment of $x_1,\dots,x_n$ to $v_1^i,\dots,v_{k_i}^i$ for $i = 1,2,\dots,N$, where $\tilde{P}_i$ is either P_i or $\neg P_i$, choosen arbitrarily in each assignment. □

We denote diagram formulas by $\Delta(x_1,\dots,x_n)$. The idea of introducing diagram formulas is borrowed from the paper J.A.Makowsky and M.Y.Vardi (1986), where they used the method of diagrams see in Chang and Keisler (1977) and McKinsey (1943) to prove characterization of relational dependencies.

A diagram formula $\Delta(x_1, \ldots, x_n)$ determines a unique interpretation of L over the universe A' of the symbols $x_1, \ldots, x_n$. We denote this symbolic model by M_Δ.

In the opposite direction, for every finite model M there exists a (minimal) diagram formula Δ_M, such that M_{Δ_M} is a homomorphic image of M. The method of diagrams of model theory is based on the theorem, that a model M' can be homomorphically embedded into a model M iff the diagram formula of M' is satisfiable in M.

This theorem says, that if a diagram formula $\Delta(x_1, \ldots, x_n)$ is found satisfiable in M, then a fully determined submodel of M is learned, and we can observe n distinguishable elements of U in the roles of $x_1, \ldots, x_n$.

3. LEARNING THEORY OF DIAGRAM FORMULAS

The straightforward and simplest way of learning a satisfiable diagram formula $\Delta(x_1, \ldots, x_n)$ in a model M is to systematically ask the ORACLE the satisfiability of (atomic formulas with or without negation) for the combinations of the k previously checked atomic sentences.

At the end of this process we get for an n-place predicate a k^n long sequence of yes or no values. So we can associate a binary sequence to each predicate symbol in Δ. Concatenating these sequences and adding a prefix encoding the number of variables, p redicate names and the arities of the predicates, we assign a unique binary sequence to each diagram formula. This special code is the same as the canorical form of the relational databases introduced in Benczúr (1987), (1988) and we call it the canonical form of diagram formulas.

This canonical form is suitable the application of Kolmogorov's algorithmic information quantity. (See Kolmogorov (1965), Kolmogorov and Uspensky (1987).

Definition 2.:

Let K_0 be an optimal partial recursive function* mapping finite binary words to finite binary words, such that for every partial recursive function K

$$\min_{K_0(y)=x} |y| \leq \min_{K(y)=x} |y| + C_K$$

for every x, and the constant C_K depends only on K. The algorithmic information quantity of a finite binary word x is

$$I(x) = \min_{K_0(y)=x} |y|. \square$$

The Kolmogorov's information quantity of a diagram formula Δ is the algorithmic information quantity of its canonical form, and we denote it by $I(\Delta)$.

*such function exists according to a theorem of Kolmogorov

From the definition of $I(x)$ it follows.

Theorem 1.

Let G be an algorithm recognizing a recursive class of finite models asking the ORACLE $T(\Delta)$ times to recognize Δ, where Δ is the diagram formula of the model to be recognized. Then there exists a constant C_G such that $I(\Delta) \leq T(\Delta) + C_G$. □

This theorem neglects the algorithmic cost of the ORACLE investigating the model, or what is the same we charge a unit cost for an oracle call. The class of sentences that can be asked are described by the learning protocol. Different learning protocols affect only the cost of a step in the learning process.

From Kolmogorov's theorems for the estimation of $I(x)$ (see Zvonkin and Levin (1970) theorem 1.3) the following can be aesily proved:

Theorem 2:

Let D be a recursive set of diagram formulas generated by the partial recursive function $g(n,k)$ of two variables. Suppose, that $\Delta_k^* = \{\Delta : (\exists i)(g(k,i) = \Delta)\}$ is finite for $k = 1, 2, \ldots$ Then the majority of D_k cannot be learned faster than $O(\log_2 |D_k|)$. □

Corollary 1. Let the parameter n be an encoding of the following size paraméteres of the models: k the cardinality of the universe U, N the number of the predicates and $n_1, \ldots, n_N$ are the arities of the predicates. So, if a set D of diagram-formulas is learnable in time polynomial in the paraméteres, then $|D_n| \leq 2^{P(n)}$, where $P(n)$ is a polynomial of the parameters. Since the total number of the models of this size is $2^{\sum k^{n_i}}$, learnable classes must contain only very few elements. But it is only a necessery condition, the elements of a learnable class must be simple as well according to theorem 1. □

Theorem 2. and the corollary do not contradict to the existence of a polynomial time algorithm in Valiant (1984) for recognizing disjunctive normal form expressions. Indeed, Valiant's algorithm runs in polynomial time in the degree of the DNF. But the typic al degree of a DNF with n variables is near to $n^{-1}2^n$. This estimation can be proved from Valiant's algorithm and theorem 2.

4. COOPERATIVE LEARNING

The results of the previous chapter gives a stress to Valiant's opinion in Valiant (1984). "If the class of learnable concepts is as severaly limited as suggested by our results then it would follow that the only way of teaching more complicated concepts is to bui ld them up from such simple ones. Thus a good teacher would have to identify, name and sequence these intermediate concepts in the manner of a programmer. The results of learnability theory would then indicate the maximum granularity of the single concept s that can be acquired without programming."

In this part we show a step in this direction. Our proposal is to use during the process of incomplete learning simple, learnable diagram formulas.

Finding a small submodel in a larger model might be simpler than learning it as a model.

When a satisfiable diagram formula is learnt it can be used unambigously in subsequent learning or communication. This is because every subset of the universe of the model satisfying it shows a unique behaviour for each of the observers. New observations can be related to diagram formulas so that some of the existentially quantified variables are bound to diagram formulas too.

The roles of variables in diagram-formulas can be identified by their index. Another way to reference these roles in the use of a symbolic first order language L_Δ, consisting of the predicate symbols used in Δ and the equality relation. The interpretation of L_Δ is fixed over the universe $X \equiv \{x_1,\dots,x_n\}$ of the variables by the diagram formula. This means that an assignment to a predicate P is true if the same atomic sentence is not negated in Δ. The language L_Δ can be used to communicate knowledge about subsystems satisflging Δ. Two observers or learners can exchange knowledge in a shorter form, or using L_Δ they can describe the roles of some variables in a formula of L.

The goals of a learning process can be not only submodels but derived, simpler models as well. We call a new n-place predicate P_ψ a derived predicate associated to the formula $\psi(y_1,\dots,y_n) \in L$, if $y_1,\dots,y_n$ are the free variables of ψ and for every interpretation of L $\forall(y_1,\dots,y_n)(\psi(y_1,\dots,y_n) \equiv P_\psi(y_1,\dots,y_n))$. The definition of diagram formulas can be easily extended for derived predicates in the following way:

Definition 3:

A k-variable diagram formula associated to ψ is defined by

$$\Delta^\psi(x_1,\dots,x_k) = \prod_{\substack{1 \le i_j \le k \\ j=1,2,\dots,n}} \tilde{\psi}(x_{i_1},\dots,x_{i_n}),$$

where $\tilde{\psi}$ is either ψ or $\neg\psi$. □

A derived predicate P_ψ is learnt in a model M of an n-element universe if we have found an existentially satisfiable n-variable derived diagram formula Δ^ψ that is $\exists(x_1,\dots,x_n)(\Delta^\psi(x_1,\dots,x_n))$ is true in M. A derived model M' of the model M is given by a system of derived predicates $P^{\psi_1},\dots,P^{\psi_k}$. The model M' can be investigated by the derived language L' consisting of the derived predicate symbols. The diagram formula of M' in the language L has the form

$$\Delta^{\psi_1}(x_1,\dots,x_n) \wedge \Delta^{\psi_2}(x_1,\dots,x_n) \wedge \dots \wedge \Delta^{\psi_k}(x_1,\dots,x_n).$$

Using this notation we can describe a formal model of cooperative learning. Let M_1, M_2, M_3 be derived models of a model M, given by the three systems of formulas $\{\psi_{11},\dots,\psi_{1N_1}\}$, $\{\psi_{21},\dots,\psi_{2N_2}\}$ and $\{\psi_{31},\dots,\psi_{3N_3}\}$. The derived languages are L_1, L_2 and L_3. Two observers A and B are learning models M_1 and M_2 respectively.

Both of them know the derivation formulas of their own model and that of the model M_3. Observer A uses the language L_1 and B uses L_2. They can only communicate existential sentences of L_3 proved to be true in M_3. The goal of the cooperative learning of A and B is to learn separately model M_1 and M_2 and learn together model M_3. Knowledge of M_3 helps in learning model M_1 and M_2 as well. Obviously, the kind of models and the extent they could be learnt depends on the systems of the derivation formulas. Quantitative analysis of the learning process is likely to be based on the concepts and theory of Kolmogorov's conditional information quantity.

Acknowledgement.

I am gratiful to Lajos Ronyai for calling my attention to Valiant's paper and his help provided in the preparation of the paper.

REFERENCES

A. Benczúr (1987). "Information measurement in relational databases" Lecture Notes in Computer Science, 305 proc. of 1st Symposium on Mathematical Fundamentals of Databases Systems (1987), 1-10 Springer-Verlag, 1988

A. Benczúr (1988). "Performance evaluation model of database systems based on Kolmogorov's algorithmic information quantity." (In Hungarian) Thesis for the degree of Doctor of Science, 1988

C.C. Chang, J.H. Keisler (1977). "Model theory". North-Holland P.L. 1977

A.N. Kolmogorov (1965). "Tree approaches to define the concept of information quantity" (in Russian) Problems of Information Transmission, 1965,I.1. 3-11. Moskow

A.N. Kolmogorov, V.A. Upensky, (1987). "Algorithms and randomness." (in Russian) Probability Theory and its Applications, XXXII,3.1987. 425-455. Moskow

J.A. Makowsky, M.Y. Vardi (1986). "On the expressive power of data dependencies". Acta Informatica 23, 231-244 (1986)

J.C.C. Mc Kinsey (1943). "The decision problem for some classes of sentences without quantifiers." J.Symb.Logic 8, 61-76 (1943)

L.G. Valiant (1984). "A theory of the learnable" Comm. ACM, 27(11), 1984. 1134-42

A.K. Zvonkin, L.A. Levin (1970). "Complexity of finite objects and foundation of the concept of information and randomness by the help of the theory of algorithms." (in Russian) Uspechi Mat.Nauk, 1970.XXV. 85-127.Moskow

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 119-123.

HUNGARIAN AND GERMAN SPEAKING COMPUTERS FOR THE BLIND

A. Arató, T. Vaspöri

Central Research Institute for Physics,
Hungarian Academy of Sciences

G. Olaszy

Linguistic Institute of the Hungarian Academy of Sciences

Abstract. The authors discuss possible methods displaying computer information for the Blind. For many cases speech output is the most efficient method for working by the Blind. The Hungarian made BraiLab talking computer family is introduced in the paper. The BraiLab is the school computer of the Hungarian Society for the Blind and the Hungarian Primary School for the Blind. On the basis of our Hungarian experiences we developed a German text-to-speech system and a first variant of the German speaking BraiLab.

Keywords. Aid for the Blind, Speech Synthetizing, Human Engineering, Education.

1. INTRODUCTION

The authors discusse possible methods displaying computer information for the Blind. The cheapest method is the speech synthesizing. For many cases speech output is the most effective method for working by the Blind. The Hungarian made BraiLab talking computer family is introduced. The BraiLab is the school computer of the Hungarian Society for the Blind and the Hungarian Primary School for the Blind. There is more than two year practice of using these about 100 talking BASIC computers. On the bases of our Hungarian experiences we developed a German text-to-speech system and a first variant of the German speaking BraiLab.

2. DISPLAYING INFORMATION FOR THE BLIND

There are several microcomputer based aids which are available for blind people; these are very useful for handicapped programmers, lawyers, teachers, philologists and other specialists. The aids which support tactile interfaces generally fall into one of two types.

The first type is the optical to tactile converter. With this device a reading speed 90 to 100 words/min can be achived. The advantage of this device is that any ink-printed material is readable by the blind user.

The other kind of tactile interface is Braille. The reading speed is faster (about 250 to 270 words/min).

Both types are more or less international, though the Braille coding has some language specifics. Tactile aids are very expensive.

A new and promising area of development for aids for the visually handicapped is using computer synthesized voice output. Only a text-to-speech based system can be taken in consideration. The fix vocabulary system can be used only very tightly.

3. SPEECH OUTPUT SYSTEMS DEVELOPEMENT

Text-to-speech systems are language specific. A Hungarian speech system was not available on the market. Research was carried out in the field of formant analysis and synthesis for the Hungarian and German languages by the Linguistic Institute of the Hungarian Academy of Sciences.(1) (3) Finally a text-to-speech system was developed in the Central Research Institute for Physics of the same Academy. On the basis of these results a project was initialized to build a talking microcomputer for the Blind.

In the development of the project, many problems which could only have been resolved with semantic analysis were simplified with an intermediate Braille coding system.

An MEA-8000 formant synthesiser was used in the project. This is a very low cost integrated circuit capable of producing any language. Initially the ASCII-to-speech converter program was written in macro assembler language for an Intel 8085 based microcomputer. After that a "speech parameters" editing system was written using the high level programming language C. The speech parameters could be loaded into the development microcomputer in the proper form, suitable for the text-to-speech converter. This was a particulary helpful working system which helped us to develope the project more quickly.

The German text-to-speech program was produced using the Hungarian developement system.

A blind research worker took part in the development. For this reason an SDK-85 was completed with a fixed English vocabulary using the Digitalker synthesizer. She could program the MEA-8000 chip with the help of Digitalker.

4. THE BRAILAB TALKING COMPUTERS

A talking personal microcomputer was built using the text-to-speech system. The goal of the project was to develop a very low cost personal microcomputer for Hungarian blind people. The input/output system of the Z80 based microcomputer was modified so that even the screen editing system can be used by visually handicapped people. The following talking programs can be used on the smaller version of the BraiLab computer: the talking BASIC and the talking Assembler with Disassembler and monitor. 48 Kbyte free memory is available for programming. This smaller version is usefull for teaching purposes.

The advanced version of talking personal microcomputer developed for the Blind is called BraiLab Plus. This machine is supplied with a talking version of CP/M compatible operating system. A talking text editor and a talking data base system is ready to use. The BraiLab Plus is usefull for working purposes. It is supplied with 386 Kbyte floppy drive, 64 Kbyte RAM memory, 186 Kbyte built in RAM floppy, serial (RS232C) and parallel CENTRONICS interfaces. The speech sinthetyzer is integrated into the computer. Both versions are transportable.

Only the BraiLab Plus version has been developed for German language because the German text-to-speech system is much larger than the Hungarian one.

The Hungarian Academic of Sciences-Soros Foundation finances the developement of a reading machine for the Blind. An experimental version the BraiLab already reads inkprinted pages with the help of the character recognition system of Computer Research Institute.

5. HUMAN ENGINEERING ASPECTS

An aid for the Blind must serve rehabilitation goals. It means that blind people have to be able to work independently as far as possible. The productivity of their work must be near to the sighted persons' one.

Using a reasonable tempo of voice output, a user can achieve about 320 to 350 words/min reading speed, which is faster than using Braille. That is why the speech output can be used more effectively in some cases than Braille. Although the BraiLab is able to sing, Braille music notations can be read faster with tactile method of course.

Most of the people who lost their sight as an adult can read Braille very slowly. Their integration into the society can be resolved easier with computer speech output.

The BraiLab can be used rapidly because blind people do not need to switch off-line for reading the screen of the computer. During text editing, echoing functions are in operation. At the end of each line when Carriage Return is pressed the computer reads out the whole line as connected text. Numerals at this point are not read character-by-character but as wholes (e.g. thirty-nine rather than three, nine). The echoing of characters, words and lines are performed automatically during typing and editing.

The computer is able to speak as soon as it is switched on. All the messages and error messages besides appearing on the screen can be heard by the Blind. The speed of the speech can be changed from normal to double or half. The speech output can be interrupted very simply.

The BraiLabs have a video output connector. A standard monitor can be connected for visualising the 24 lines by 80 coloumns for sighted people. This is for helping on the integration of the Blind. They can cowork with sighted persons.

6. EXPERIENCES WITH BRAILABS

In the Hungarian Society for the Blind several fundamental courses of computer technics were organised for beginers. About 40 people finished these courses. The Users' Manual for BraiLab has been published on cassette tape and in Braille print as well. (2)

In the Hungarian Primary School for the Blind, pupils enjoy almost every subject when they use BraiLab as an aid. One of the special features of BraiLab is that it can also sing. To make the computer sing, the user has to specify the correct rhythm and the correct sequence of tone. The melody has to be given in relative sol-fa letters, accordind to Zoltán Kodály's method.

7. CONCLUSIONS

There are two types of computer interface available for the Blind. The first is a tactile type, and the other is based on voice output. Tactile interfaces are more or less international, whereas the synthesized voice systems are language specific. A Hungarian-speaking computer called BraiLab is used widely in schools and the Hungarian Society for the Blind. Some special human engineering problems were resolved in the BraiLab computer.

Voice synthesis is a very promising way of creating aids for the Blind. Talking microcomputers are low cost devices which can be used widely in education and in the work of visually handicapped people. In the text processing field, such talking systems can be used at least as well as their tactile counterparts if the human engineering problems are resolved properly.

References

(1) Kiss G.—Olaszy G. An Interactive Speech Synthesizing System with Computer and OVE III. Synthesizer MFF 10. 1982, 21-46. Linguistic Institute of HAS.

(2) Arató A.—Vaspöri T. Guide to Using BraiLab's Talking BASIC. VGYOSZ 1985. Hungarian Society forthe Blind

(3) Bolla K. —Valaczkai L. The articulatory and acoustic features of German speech sounds. Synthesizer MFF 16. 1986. Linguistic Institute of HAS.

Budapest 8-aug-1988

13
A

GRAPHICS AND DATABASES

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 127-138.

Towards a simple yet expressive picture description language

W.D. Fellner, J.K. Stögerer

IIG, Institutes for Information Processing Graz,
Graz University of Technology and
Austrian Computer Society (OCG)

Abstract

Integration of geometric data into current available relational data base systems has been of major research interest in the last few years. Various solutions to this problem have been presented including extensions at the front-end, designing application-independent kernel systems and/or changes at the conceptual level.

In this approach we restrict ourselves to two-dimensional geometric data stored in a standard file format (CGM). Within this environment we concentrate on the graphical data trying to find a relational front-end formalism for a simple graphical description. This leads us to the picture query language $\mathcal{VIC}$ based on relational-calculus which can be easily extended for handling of non-graphical data.

$\mathcal{VIC}$ is not primarily intended for user-interaction and could best be used as a subsystem within a device- and application-independent graphics system. It can be seen as a generalized pick-function supporting graphical retrieval and manipulation operations. As basic components of a $\mathcal{VIC}$ predicate the set of graphical primitives and their attributes - maybe hierarchically nested and in combination with graphical variables - can be used in a simple yet expressive way.

Keywords: non-standard databases, query languages, graphical databases, CGM, non-procedural languages, $\mathcal{VIC}$

1 Introduction

Search and replace functions in non-graphical applications like text editors and relational data base systems are standard operations since quite a long time. They have undergone significant changes conceptually, in style and functionality within this period starting with very simple and limited features, e.g. a few special characters like ***don't cares*** and ***wild cards***, to end up with powerful and convenient languages or language fragments (regular expressions, various query languages based on relational calculus or relational algebra).

However, in graphical systems the scope of a search function, called the ***pick-function***, is mostly limited to the current picture within the edit process. In recent time more sophisticated graphical systems, mainly CAD applications, geographical information systems, chip design layout systems etc., have increased the functionality and allow restricted graphical retrieval over a large set of files or graphical libraries. Although there are big differences between them at the user interface (relational front-ends, mostly SQL compatible [8,1,25], application specific forms, query by example, etc.) the bulk of these so-called ***non-standard data base applications*** [13] uses the same approach: the graphical query statements are used as input for a transformation function which maps the geometric query to the underlying conventional (non-graphical) relational data base system. Sometimes this is a very crude process which does not yield best performance [12]. Therefore research efforts try to overcome the shortcomings of the relational model with respect to graphics and/or special applications. This is done by proposing extensions [20,22,9] or by designing more general models [5,19].

In this paper we present a simple yet very powerful and expressive picture description language which can be used for efficient graphical retrieval and manipulation. It can be seen as a ***generalized pick-function*** and is designed to work within a special environment. So we first concentrate on the basic assumptions and constraints. Then we present the key features of the query language ***VIC*** : the basic geometric data types, syntactic structure and semantics of ***VIC*** predicates and the variable concept.

2 Basic Assumptions and Constraints

In the following we restrict ourselves to two dimensions and therefore we call the objects, we are interested in, <u>pictures</u> and not <u>graphics</u> deliberately. A picture is composed of n geometric objects $O_1, O_2, \ldots, O_n$ which are stored in a standard file format. In the display process they are interpreted sequentially. Hence we can speak of a time axis t

(order of display) and of n independent layers of a picture.

O_1	O_2	$\cdots$	O_{n-1}	O_n

$\longrightarrow$ order of display

There are two types of geometric objects:

- primitives; e.g. circle, rectangle, line, polygon etc. The set of available primitives is defined by the graphics standard [15,16,17,18] used by the application.
- pictures; an object in a picture may be another picture within the database. We call this a **subpicture**. A subpicture can be composed of subpictures itself.

This implies a structure of arbitrary complexity, however, recursion is not allowed. This structure can be implemented by some sort of USES/USEDIN lists associated with every picture. A typical example is a library including standardized pictorial units which are used – together with primitives – to compose new pictures.

The basic graphics standard used in this paper is CGM - Computer Graphics Metafile [17]. CGM became ISO standard in 1987 and provides a means for the exchange of graphical information [14]. With its advent the computer graphics industry for the first time has a versatile and application-independent standard for the capture, transfer and archiving of multiple, device-independent picture definitions. An extension to it – CGEM – will enable GKS-applications [15] to store pictures in CGM-format and is currently in preparation [18].

The question, which layer of presentation the search and replace function process should operate on, is of principal importance and has major influence for the resulting query language. A picture can exist in various different formats:

level 0: device independent format on background storage. Within the context of this paper it is the CGM format (in CGM coding can be done in one of three equivalent encodings: binairy encoding, character encoding, and cleartext encoding).

...

level *i*: intermediate formats within the graphics editor using internal data structures.

...

level n: device dependent level: e.g. a raster graphics display. The picture is defined as an array of pixels with different colour and brightness.

In order to achieve general applicability and to avoid problems of portability it is absolutely necessary to keep away from device and application dependencies. Therefore the proposed query language *VTC* operates on level 0 and is an **object-oriented** and not a **pixel-oriented query language**. Pixel-oriented languages, better known as ***image processing languages (IPL)*** [3] within the area of **pattern matching**, operate on layer n. IPL's build the main stream of research activity and many of them are currently in use in various applications (medicine, mining, ···) [2,23].

On the other hand there are only a few publications concerning object-oriented retrieval languages [4]. The basic units of an object-oriented query language are the **geometric objects** as well as **geometric operators and functions** which can be used to compose or transform objects.

3 The Query Language *VTC*

3.1 General Overview

VTC is a non-procedural picture query language encouraging thorough optimizations [7] by the system. Its syntax is keyword-oriented and very similar to (the quasi-standard) SQL. It concentrates on **'what'** has to be found rather than **'how'** the information has to be searched for. *VTC* can be used easily within an imperative host language. Therefore we can identify two system parts:

- **precompiler:** Embedded *VTC* statements within a host language like PASCAL are transformed by the precompiler into calls compatible to the runtime modul *VTX*. Within predicates constants and variables can be used. A lot of time consuming work can be done during parsing (choice of data access path, optimizations, bindings, ...). Furthermore, during execution it is possible to assign the results of *VTC* statements to variables.
- **runtime system *VTX*:** *VTX* is not a general interpreter. From a library *VTX* routines are linked to the current application.

In *VTC* there are two types of statements:

- **declarative:** declarative statements serve two purposes: the specification of variables and the definition of control parameters controlling the search process.

  Examples: SET, DEFPIC, DEFMAC, DEFOBJ, ...

- **executable:** this type of statement immediately provides a result, e.g. the GET statement searches for the picture(s) matching its predicate. Besides the GET statement there are some manipulation statements for update operations and some *"cursor"* statements for the usage within a host language (this corresponds to the *cursor-concept* of conventional relational data base systems).

3.2 Structure of *VIC* Statements

Each retrieval statement traversing the underlying picture data base calculates a **result set** of n matching pictures, $n \geq 0$. The **geometric characteristics** of the determined pictures are defined by the **predicate**. A predicate is a boolean expression of arbitrary complexity which has to evaluate *"true"* for a picture to be included into the result set. A predicate is composed of **tokens** which are combined according to certain combination rules (see table 1 for unary and binary operators). A token has the following structure:

(*objid*, [*attributes*])

or:

([*objid*], *attributes*)

The brackets [ ] denote optional existence, ( ) are used as delimiters.

The first parameter *objid* is the **object identifier** and denotes one of three geometric types, OBJ, MAC, and PIC, respectively:

- **primitive:** (type OBJ) the set of primitives is defined by the underlying graphics standard, e.g.: CIR, REC, ARC, POL, LIN, ...
- **macro:** (type MAC) elements of a picture may be combined to define a part of the picture: a macro. Essentially, every sequence of tokens defines a macro, i.e. the identification of pictures is done by identification of macros.
- **picture:** (type PIC) each CGM file may hold n pictures, $n \geq 1$. A picture with an empty USES list is entirely made up of primitives, otherwise there are references to subpictures. A picture with a non-empty USEDIN list is referenced as a subpicture elsewhere.

Macro and *picture* are composite types and made up of primitives, macros, and/or pictures. This means that through a macro or picture identifier a series of tokens is referenced implicitly. The **nesting** of tokens can be done explicitly, too. Therefore the parameter *objid* of a token can be an arbitrary series of tokens itself.

$$\langle \langle\ldots\rangle\langle\ldots\rangle\ldots\langle\ldots\rangle, \ldots attributes\ldots\rangle$$

The evaluation rules for such complex tokens are treated in detail in [24].

$\langle E_1\rangle\{n,m\}$	n to m occurrences of object E_1, $n \leq m$ and $n,m \in \mathbb{N}_0$. Instead of n and m the $*$ may be used (see next lines).
$\langle E_1\rangle\{2,*\}$	at least two occurrences of E_1.
$\langle E_1\rangle\{*,5\}$	at most five occurrences of E_1.
$\langle E_1\rangle*$	arbitrary number of occurrences of object E_1.
$\langle E_1\rangle+$	at least one occurence of E_1. ($\langle E_1\rangle+ \Leftrightarrow \langle E_1\rangle\{1,*\} \Leftrightarrow \langle E_1\rangle$)
$\langle E_1\rangle 3$	exactly three occurrences of E_1 (equivalent to: $\langle E_1\rangle\{3,3\}$)
$\langle E_1\rangle 0$	negation: the object E_1 must not be an element within the picture
$\langle E_1\rangle ?$	option: 0 or 1 occurrences of E_1; (equivalent to: $\langle E_1\rangle\{*,1\}$)
$\langle E_1\rangle !3$	complement: the number of occurrences of E_1 must not be equal 3
$\langle E_1\rangle\ \langle E_2\rangle$	existence of the objects E_1 and E_2 (at least one occurrence each). The order of E_1 and E_2 within the picture (with regard to time axis t) is of no importance.
$\langle E_1\rangle\hat{}\langle E_2\rangle$	sequence: sequential order (with regard to time axis t) of E_1 and E_2 ('AND' is equivalent to ^).
$\langle E_1\rangle \vert \langle E_2\rangle$	alternative: inclusive or ('OR' is equivalent to $\vert$).
$\langle E_1\rangle$ XOR $\langle E_2\rangle$	alternative: exclusive or
$(\langle E_1\rangle)$	grouping of tokens

Table 1: token operators

From a functional viewpoint (an approach selected in [21]) the second parameter, *attributes*, can be seen as an operator upon the object(s) designated by the object identifier *objid*. However, we will take a more conventional view, treating them as additional conditions that have to be fulfilled by the object(s). Those aspects which are of no importance for the user can be omitted and are treated as don't cares. Attributes may be categorized within three classes:

- **shape oriented:** attributes defining line type, brushing, colour, interior style, ...
- **position oriented:** attributes defining the position of objects within the picture
- **relational attributes:** this type of attribute is particularly of interest for retrieval purposes. Attributes of this class enable the user to define complex geometric relations between picture elements in a natural way (relative position of an object with respect to other objects, an object hides (or is hidden by) other objects, an object intersects other objects, ...)

A few examples may illustrate the features presented so far.

```
GET ALL PICTURES WHERE (LIN, DEFXY=(x1, y1, x2, y2)5);
```

In *VTC* it is possible to state the exact position of objects by specifying the definition points. Above the postfix 5 establishes that pictures including lines, that have x,y-values differing ±5 units are still included into the result set. However, this way to define a position will rather be the exceptional case because of being too detailled.

```
GET NAMES FROM PICTURES(1..10) WHERE
    (CIR, AREA = REC(x1,y1, x2, y2)) SORT ASC ON DATE;
```

This token defines circles, that are fully contained within the surrounding rectangle (x_1, y_1, x_2, y_2), i.e. the result of the above query statement are the ten oldest pictures that have at least one circle in the specified area. It is possible to define more than one surrounding area and combine them via set operators.

e.g.: `(LIN, AREA <> REC( ... ) OR REC( ... ))`

```
GET ALL PICTURES WHERE
   (MAR, MTYPE<>[3,1], MCOLOR = [*9], MSIZE<5){*,5};
```

The above token defines polymarkers of special type (not equal to 1 or 3), special colour (not colour 9) and special size (less than 5). The query finds all pictures containing at most 5 of these special polymarkers.

```
GET ALL PICTURES WHERE
   (OBJ, COLOR=red, INTERSECT((CIR, COLOR=blue)1)){*,10};
```

In the above token a relational attribute INTERSECT is used. It defines that at most ten arbitrary red-coloured primitives (OBJ) that intersect exactly one blue circle have to be found in a picture to be included into the result.

```
GET PICTURE(1) RESTRICT (, AREA=REC(x1,y1,x2,y2), COLOR=green);
```

From the first picture that fits the predicate only the elements stated in the predicate are taken (all green elements that are fully contained in the rectangular area). Instead of the usual WHERE-clause the above RESTRICT-clause provides a projection mechanism.

```
GET ALL PICTURES WHERE (REC, NOINTERSECT(SPL,)+))+;
```

The resulting pictures contain at least one rectangle that does not intersect any spline (at least one has to exist).

3.3 Variables in *VTC*

Until so far the presented picture selection capabilities of *VTC* correspond to the power of regular expressions. The postfix of a token allows the user to define the number of picture elements and relate them with binary operators and/or relational attributes. However, to be able to specify really complex geometric dependencies between picture elements we need an extension mechanism that goes beyond the presented concept of predicates and tokens. For example the following query cannot be expressed by the *VTC* language features presented until now:

Find all pictures with the following properties: above (in this context *above* stands for a larger *y*-value) a red rectangle contained in subpicture A there is a green circle, which has at most twice the size of the red rectangle.

Such complex queries can only be resolved via the introduction of variables. *VTC* provides two ways for the definition of variables:

- **explicitly:** a unique variable name can be declared in a DEFPIC // DEFMAC // DEFOBJ statement declaring a variable of type PIC, MAC, and OBJ, respectively. The following lines show an example for the definition and use of PIC variables.

  ```
  DEFPIC pid AS GET ...  ;
     ...
  GET ...  WHERE (pid,);
  ```

  The first statement defines a query and attaches the unique variable name *pid* to it. This declarative statement does not result in any search operation on the picture data base. However, when it is encountered in the precompile phase, it is parsed and prepared for execution (determination of the access paths, optimization, ...). The execution of the predefined query is done in the successive GET statement, searching for all pictures which include at least one reference to the *n* subpictures *pid* defines.

Although a reference to the physical sequence of objects within a picture is of importance only in a few situations this can be achieved in *VTC* via postpostfixes '.' (relative offset to the start of the picture) and '!' (relative offset to the start of a picture within an object type or class of objects). This is used in the next example using PIC variables in a more practical context.

Find a circuit, in which the first 5 NANDs, the first 10 FLIPFLOPs and the first 3 NORs are positioned within the area (x_1, y_1, x_2, y_2). If there are remaining NANDs they should be in the right half-plane (x_3, y_3, x_4, y_4).

```
DEFPIC nand AS GET ••• ;
DEFPIC flipflop AS GET ••• ;
DEFPIC nor AS GET ••• ;
GET ALL PICTURES WHERE
     ( (nand,){1,5}!  (flipflop,){1,10}!
     (nor,){1,3}!  WHERE AREA=REG(x1,y1,x2,y2)) AND
     ( (nand,)* WHERE AREA=PLANE(x3,y3, x4,y4)};
```

A half-plane is specified by two points and lies to the right of that vector. The above example utilizes the fact, that nested tokens are always maximized whereas tokens on the outmost level are minimized. This can be seen in the following two lines:

```
GET ALL PICTURES WHERE ((CIR,)+, FCOLOR=red);
GET ALL PICTURES WHERE (CIR, FCOLOR=red)+;
```

The token in the first query statement specifies that all circles within a picture must have colour red. Whereas the token in the second query statement says that there has to be at least one red circle, i.e. circles in other colours are allowed.

- **implicitly**; within a predicate of one *VTC* statement the tokens are assigned ascending numbers within the current level of '( )'-parenthesis. On each level the numbering process starts again with number one.

$$((\ )_{1.1} (\ )_{1.2}, \cdots)_1 (\ )_2 (((\ )_{3.1.1} (\ )_{3.1.2}, \cdots)_{3.1}, \cdots)_3$$

Essentially an implicit token identifier is the path of the corresponding tree structure. Implicit token identifiers are generated for each *VTC* statement again, so they can be seen as local variables within a statement.

Find all pictures which contain a red equilateral triangle and a brown rectangle. The triangle is centered above the rectangle.

```
GET ALL PICTURES WHERE
(REC, FCOLOR=brown, XMID=XMID(2))1
(POL, #EDGES=3, REGULAR, FCOLOR=red)1;
```

Find all pictures which contain a blue rectangle and a blue circle. The circle is centered above the rectangle with distance k and its diameter is less equal the width of the rectangle (character 'i').

```
GET ALL PICTURES WHERE
(REC, FCOLOR=blue, XMID=XMID(2))1
(CIR, FCOLOR=blue, YMIN=YMAX(1)+k, XLEN≤XLEN(1))1;
```

We conclude this section by presenting the solution of the query stated as a motivation example at the beginning.

```
DEFPIC subpicA AS GET ••• WHERE ••• ;
DEFMAC redrec AS RESTRICT subpicA TO (REC, FCOLOR=red)1;
GET ALL PICTURES WHERE (subpicA,)1
    (CIR, FCOLOR=green, WHERE YMIN>YMAX(redrec),
    SIZE ≤ 2*SIZE(redrec))1;
```

Due to space limitations a lot of details and further possibilities of *VTC* , e.g. manipulation statements, embedding in host languages, etc. have been omitted. The interested reader is referred to [24].

4 Summary

The presented object-oriented picture query language *VTC* is a first step towards a generalized pick-function. It has been designed with a special environment in mind, however, the principal concept is open to extensions for other applications. *VTC* is a relatively simple language based on only three basic geometric types and a well defined token/predicate concept. Within this framework various attributes together with the concept of variables establish the power of *VTC* .

References

[1] Blasgen M.W. et al.: *System R: An Architectural Overview.* IBM Systems Jnl., 1981, Vol.20#1, pp.41-62

[2] Chang S.K. et al.: *An image processing language with icon assisted navigation.* IEEE Trans, on Software Engineering, SE-11#8, Aug.85, pp.811-819

[3] Chang S.K./Fu K.S.: *Picture Query Languages for Pictorial Data Base Systems.* IEEE Computer, Nov.81, pp.23-33

[4] Chang S.K./Ichikawa T./Ligomenides P.A. (Eds.): *Visual Languages.* Plenum Press, 1986, 460pp.

[5] Chen P.P.: *The Entity Relationship Model. Towards a unified view of data.* ACM TODS, Vol.1#9, 1976

[6] Encarnacao J./Krause F.J. (Eds.): *Files structures and data bases for CAD.* North Holland, 1982

[7] Fellner W.D./Stögerer J.K.: *PIC - eine objektorientierte grafische Abfragesprache.* Proceedings of the Austro Graphics, Sept. 1988, ACGA'88, Wien

[8] Finkelstein R./Pascal F.: *SQL Data Base Management Systems.* Byte, Jan.88, pp.111-123

[9] Fischer W.E.: *Datenbanksysteme für CAD Arbeitsplätze.* Informatik Fachberichte 70, Springer 1983

[10] Güting R.H.: *Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems,* in: J.W. Schmidt/S. Ceri/M.Missikoff (Eds.): *Advances in Database Technology,* EDBT'88, Proc. of the Int.Conf. on Extending Database Technology, Venice, March 1988, pp.506-527

[11] Güting R.H.: *Modeling Non-Standard Database Systems by Many-Sorted Algebras.* Univ. Dortmund, Forschungsbericht Nr.255, 1988

[12] Guttmann A./Stonebraker M.: *Using a relational data base management system for computer aided design.* IEEE DB Engineering, Vol.5#2, June 1982, pp.21-28

[13] Härder T./Reuter A.: *Architektur von Datenbanksystemen für Non-Standard Anwendungen.* in: Blaser A./Pistor E. (Eds.), Lecture Notes in Computer Science 94, Springer Verlag, 1985, pp.253-286

[14] Henderson L./Journey M./Osland C.: *The Computer Graphics Metafile.* IEEE Computer Graphics and Applications, Aug.86, pp.24-32

[15] ISO International Standard: *Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) - Functional Description.* ISO Standard IS 7942, Oct. 1985

[16] ISO International Standard: *Information Processing Systems - Computer Graphics - Interfacing Techniques for Dialogues with Graphical Devices (CGI).* Part 1-6, ISO TC97/SC21 N1179, Working Draft, May 1986

[17] ISO International Standard: *Information Processing Systems - Computer Graphics Metafile for the storage and transfer of picture description information (CGM).* ISO IS 8632, ISO/TC97, Aug. 1987

[18] ISO Draft International Standard (Addendum 1): *Information Processing Systems - Computer Graphics Metafile for the storage and transfer of picture description information.* Addendum 1, ISO 8632-PDAD1, ISO/TC97/S24/N19-N22, Nov. 1987

[19] Lamersdorf W./Schmidt J.W.: *Rekursive Datenmodelle.* Informatik Fachberichte 72, Springer Verlag, 1983

[20] Lorie R.A.: *Issues in data bases for design applications,* in: [6]

[21] Lucas P./Zilles S.N.: *Applicative Graphics using abstract data types.* Research Report RJ 6198, IBM Almádén Research Center, Apr. 1988, 55pp.

[22] Pistor P./Hansen B./Hansen M.: *Eine sequelartige Sprachschnittstelle für das NF^2-Modell.* in: Informatik-Fachberichte 72, Springer Verlag, J.W. Schmidt (Hrsg.): *Sprachen für Datenbanken,* 1983

[23] Preston K.Jr.: *Progress in Image Processing Languages,* in: Duff M.J.B. (Ed.): *Computing structures for Image Processing,* Academic Press, 1983, pp.195-211

[24] Stögerer J.K.: *Suchen und Ersetzen in Bilddatenbeständen.* Inst. f. Informationsverarbeitung, Technical Report, 1988, Graz University of Technology (to appear)

[25] Stonebraker M./Wong E./Kreps P./Held G.: *The design and implementation of INGRES.* ACM Trans.on Data Base Systems, Vol.1,#3, 1976, pp.189-222

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 139-147.

"ORBIS PICTUS" AND THE INTEGRATED CAD/CAM SYSTEM

L. Cser, P. Tamàs

Technical University of Budapest

Abstract: The application of colour graphics in CAD/CAM mechanical engineering systems is for visualization (e. g. representation of temperature and strain stress-fields etc.). The presentation is about a practical application in wich selected technical information of casting in the integrated CAD/CAM systems is expressed by the colour and layer techniques.
The product design process, that is the finished component - casting - casting pattern in the integrated CAD/CAM system and the CNC machining of cast patterns require non-traditional design methods.
The application of the described method needs a new form of the cooperation between design and CAPP.

Keywords: Casting pattern, CAD/CAM Colours, Layer technics.

1. THE ENVIRONMENT OF COMPUTER-TECHNICS

The development of computer-technics created a new situation on the field of CAD/CAM application. The application of the new graphical means that is the colour display, the colour hardcopy, shading technique changed the way of usage on a number of different fields. These are e. q. the visualization of FEM results, computer design (e. q. machine production) textil design etc.

The colour-technics applications mentioned above have two essential advantages:
- It assures a better lucidity.
- There is a figurative representation of the objects.

2. PRE-FABRICATION IN CAD/CAM SYSTEMS

In the system-plan of the integrated CAD/CAM systems there is logically sequential unity of constructional planning, technological preparation, NC programming and CNC production if the production starts from freeformed forged piece or a sawed rod. When there is a more complicated forming operation in pre-fabrication (e. q. cold flowing, die forging, casting), the logical sequence is broken, prefabricate forms, technological procedures and tools have to be designed that is the constructional planning modul has to be used repeatedly. In such cases there is a logical loop in the process.

Hereinafter we are going to present the case of casting as one essential obstacle in the way of CAD/CAM integration.

3. PROBLEMS OF THE PRODUCTION OF CAST SPARE-PARTS

As well Known the steps of casting are as follows:

- First the assembly drawing of the ready spare-part.
- Then making the plan of cast piece on the basis of the construction.
- Thirdly the planning and preparing of the casting pattern usually from wood. The wooden casting pattern forms the shape of the casting in the moulding sand.
- For hollow castings so called "cores" have to be made. The "pattern" of the cores is produced like the casting pattern but in this case the negative of cores has to be moulded.
- The sand forms made by the casting pattern give the shape of casting
- Casting requires well-chosen casting technology.

There is a big contradiction between the highly developed CAD/CAM production and the traditional casting pattern production techniques.

This contradiction appears mainly in the following fields:

- The bottle neck of product output is the casting pattern factories.
- The eventual repeated using of casting patterns causes heavy pattern storage costs.
- The pattern production takes purchase mainly on the experiences and skills of specialists.

The only way of lifting this contradiction is the integration of the planning and production of casting pattern as well as the documentation storage must be solved in the complex CAD/CAM system.

Troubles can be caused if the integrated CAD/CAM system works on a shared intelligence net because it means the common use of the CAD system or the duplication of it.

The first solution infers that the CAD sub-system runs on host computer with several independent work-stations, the latter requires a constructional planning sub-system for the work-station planning the technology of the prefabrication.

4. THE TASKS OF CAD/CAM IN CASTING

When speaking about cast spare-parts technology and construction are not separable from each other. During planning one has to regard

- the function of the spare part,
- the material of the casting,
- the casting technology,
- workability,
- strength,
- constructional restrictions,
- process of forming and pattern making,
- avoiding of spoilage.

Because of diverging requirements the "science" of casting construction and pattern making belongs even today to "arts".

The steps of planning casting patterns are the following:

- making the assembly drawing,
- making the geometry of casting,
- planning the geometry of the pattern,
- forming the cores and core marks,
- choosing the adequate core boxes.

The basic elements of technological preparation are the following:

- choosing the plain of split of the casting,
- determining the upper and lower part of the casting,
- form-modification according to casting technology requirements,

- working tolerances,
- determining cores and core marks,
- rounding off the not castable angles,
- determining the necessary moulding obliquities,
- mounting the casting patterns on sheets,
- regarding the cooling shrinkage,
- optimalizing the casting technology.

5. COLOUR-GRAPHICS STORAGE IN THE CAD/CAM CASTING SYSTEM

As discussed above casting planning and the elaboration of construction must be connected process. In spite of this in practice the design engineer and the casting specialist are two different people working separately.

During a traditional design process the construction and the casting pattern are formed by several cross check talks between the design engineer and the casting specialist.

In the integrated CAD/CAM system one has to find the way of information storage and flow that gives the adequate information to the design engineer about casting problems and to the casting specialist about constructional modifications without interfering with each other's work.

The method to store and convey information and thereby create an iterative connection between the design engineer and casting specialist is the usage of layers and colour-technics.

When casting patterns are being designed the steps of casting pattern planning and technological preparation come one after the other. In every step drawings containing the geometric information are put on a new layer and so modification can be realized only on the given layer. By this technique it is possible to separate or to compare the different work steps and to move back to a preceding step and by doing so insuring the flexibility of the design process.

Standards for drawings containing casting pattern geometry - departing from machine engineering practice - prescribe colour descriptions of information. On the drawing interferences of the casting specialist are represented by various colours.

The software integrated in the CAD/CAM system planning casting technology makes it possible to use colours in a way that design information on the layers give a standard pattern drawing when the layers are put on one another. In a such a way the design engineer is able to separate parts from the whole and having necessary information to make adequate documentation about casting pattern production.

Figures 1-6. visualize the contents of layers of such a planning system.

5. COLOURS AND STANDARDS

In the preceding century colours were used in machine design for the identification of various material types. On figure 7, 8 you can see the originally coloured pages of a locomotive design guide from 1871. Later the development of manifolding technique it became necessary to identify with the help of different line types. But by doing so the lucidity of description was decreased.

The present graphical standards give the possibility to colours to be applied as Information transmitters. The GKS (ISO 7942) defines only line types, proposals for registration of graphical items handed in to ISO deal also mainly with line types and hatchststyles. PHIGS includes a lot of possibilities for design engineering to get more efficient. So the usage of colours in information transmission can probably be introduced and "orbis pictus" marches into the world of product models too.

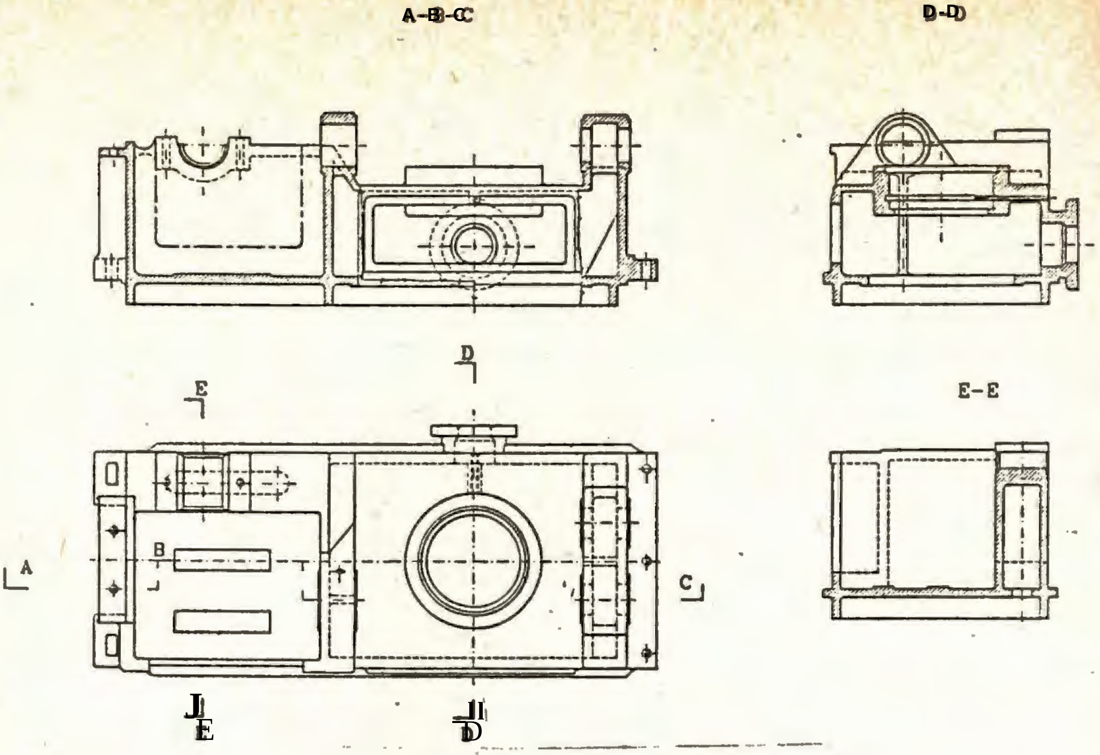


Fig. 1. The casting

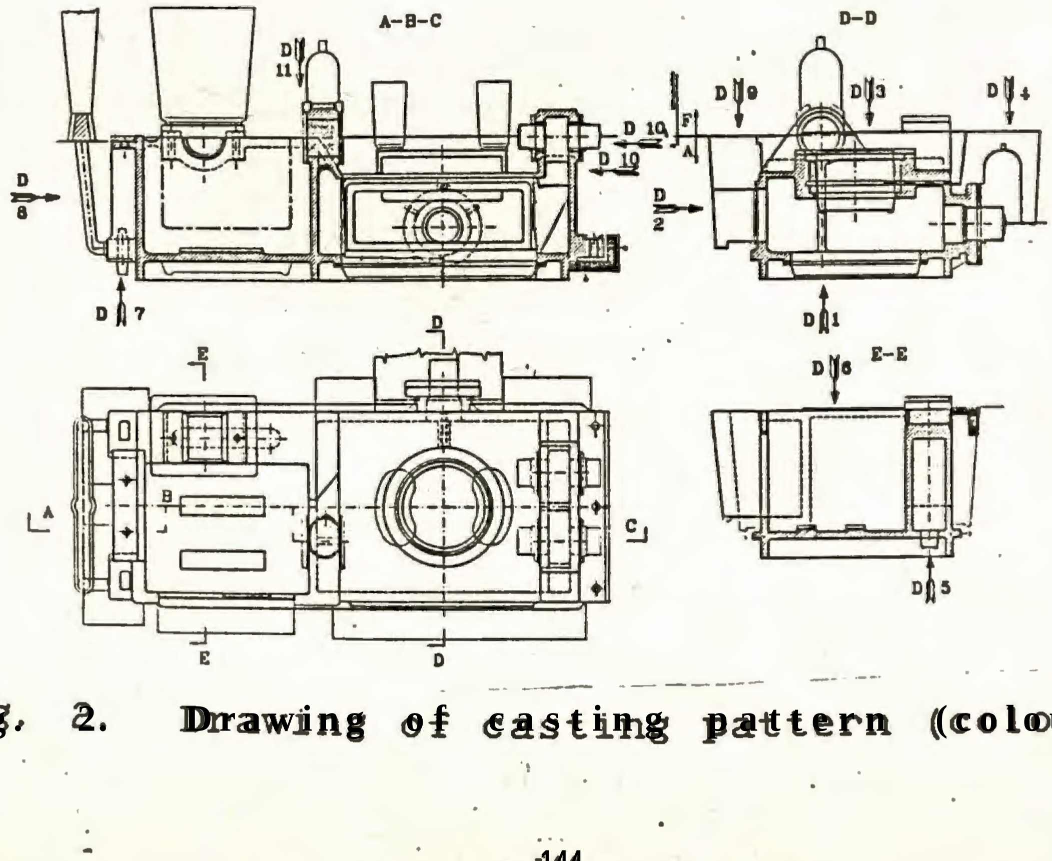


Fig. 2. Drawing of casting pattern (coloured)

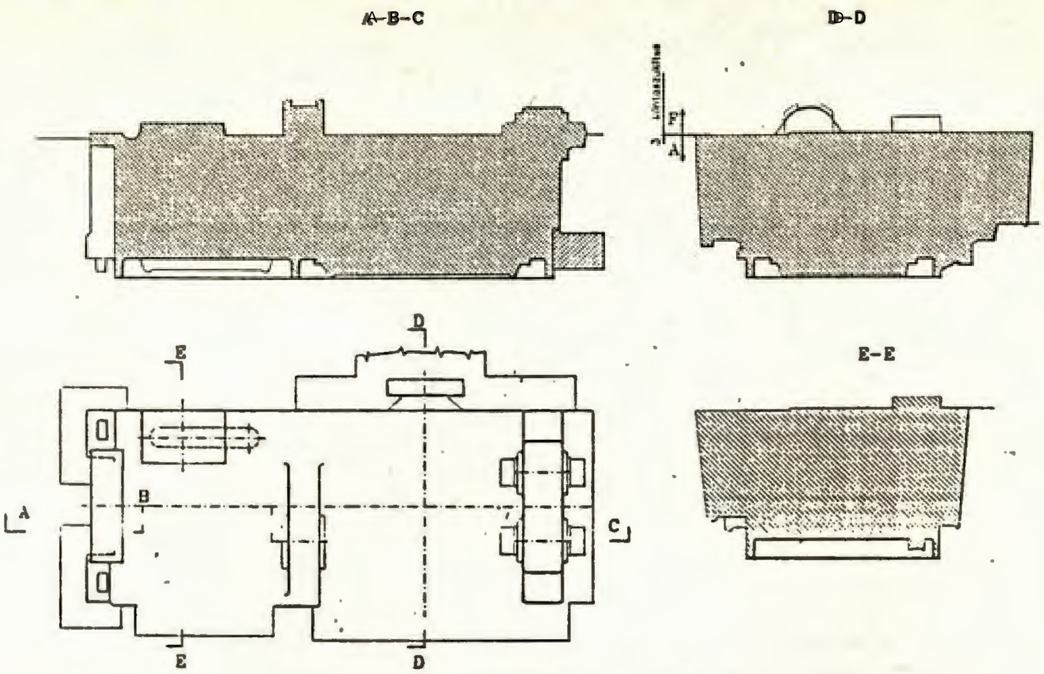


Fig. 3. The layer of pattern (red)

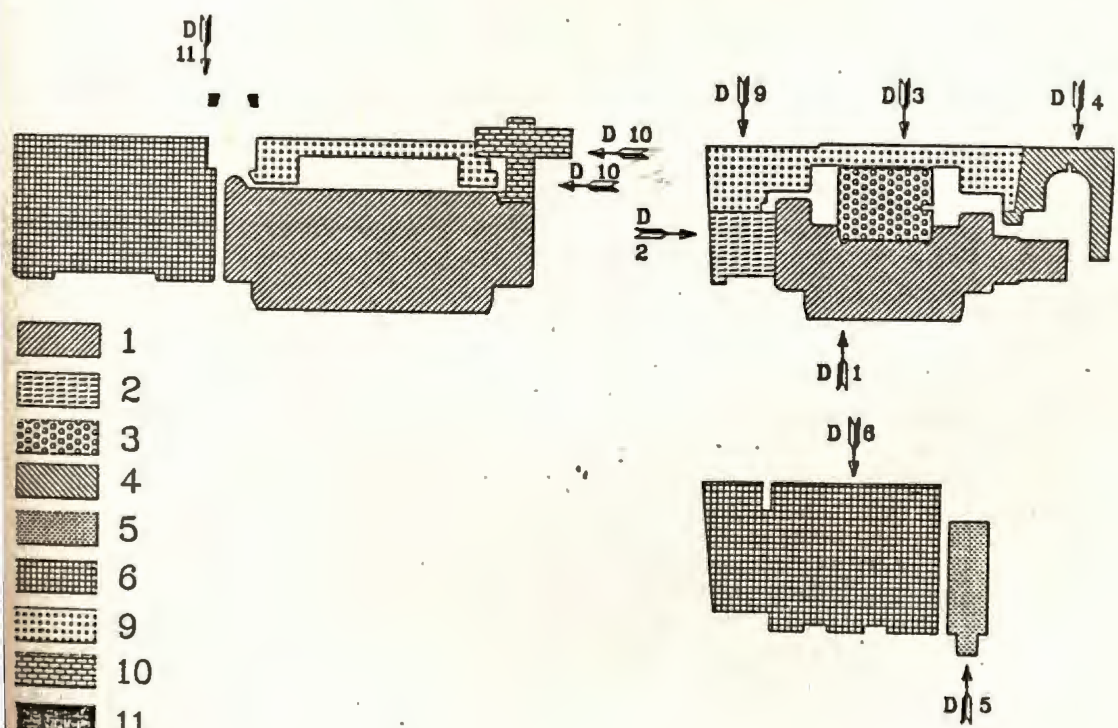


Fig. 4. The layer of cores (blue)

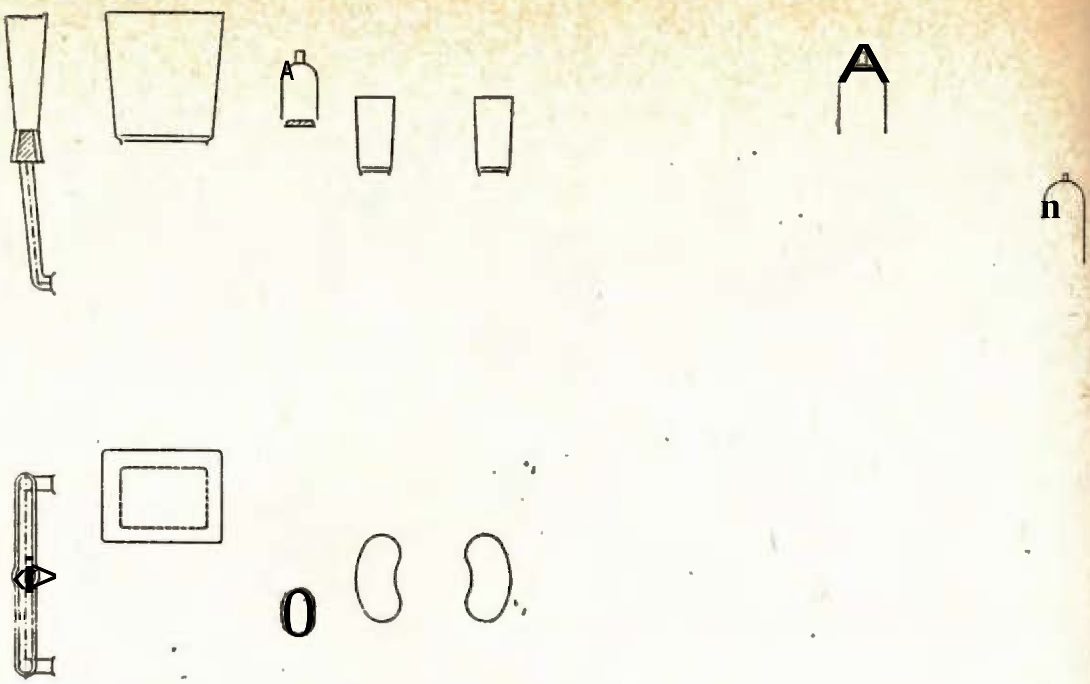


Fig. 5. The layer of runners (green)

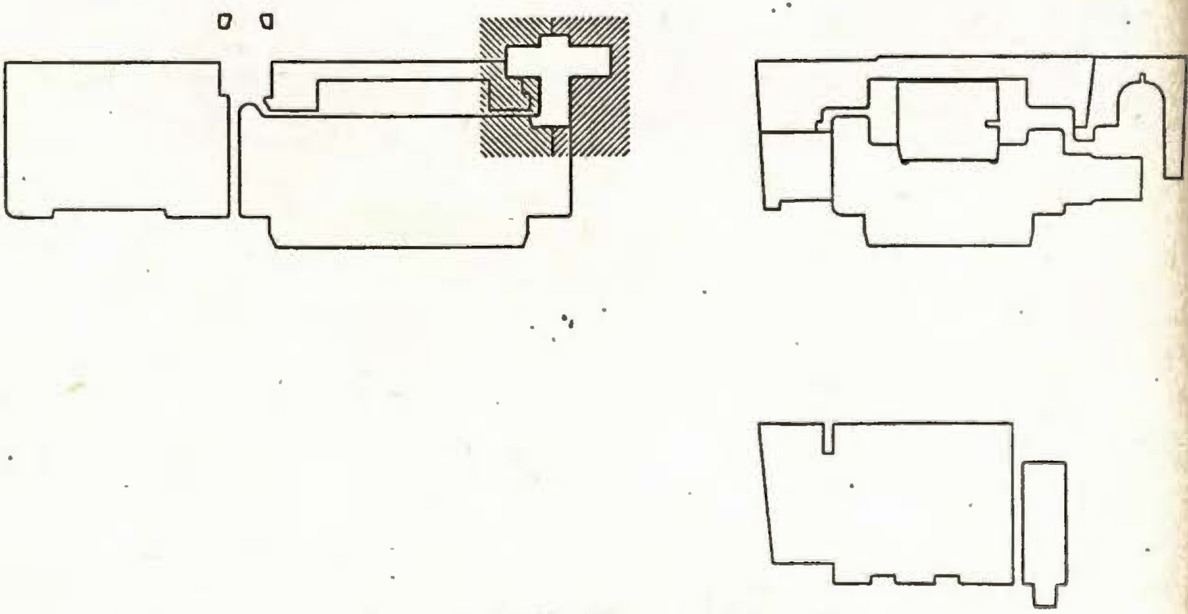


Fig. 6. The layer of coreboxes (red)

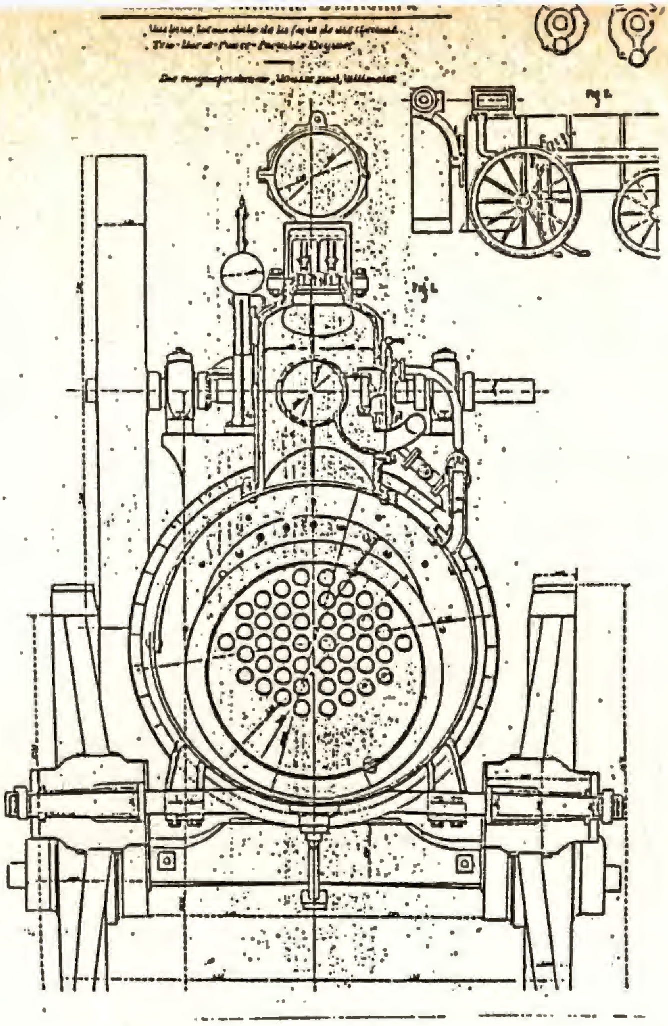


Fig. 7. A drawing of a locomotive design guide from 1871 (coloured)

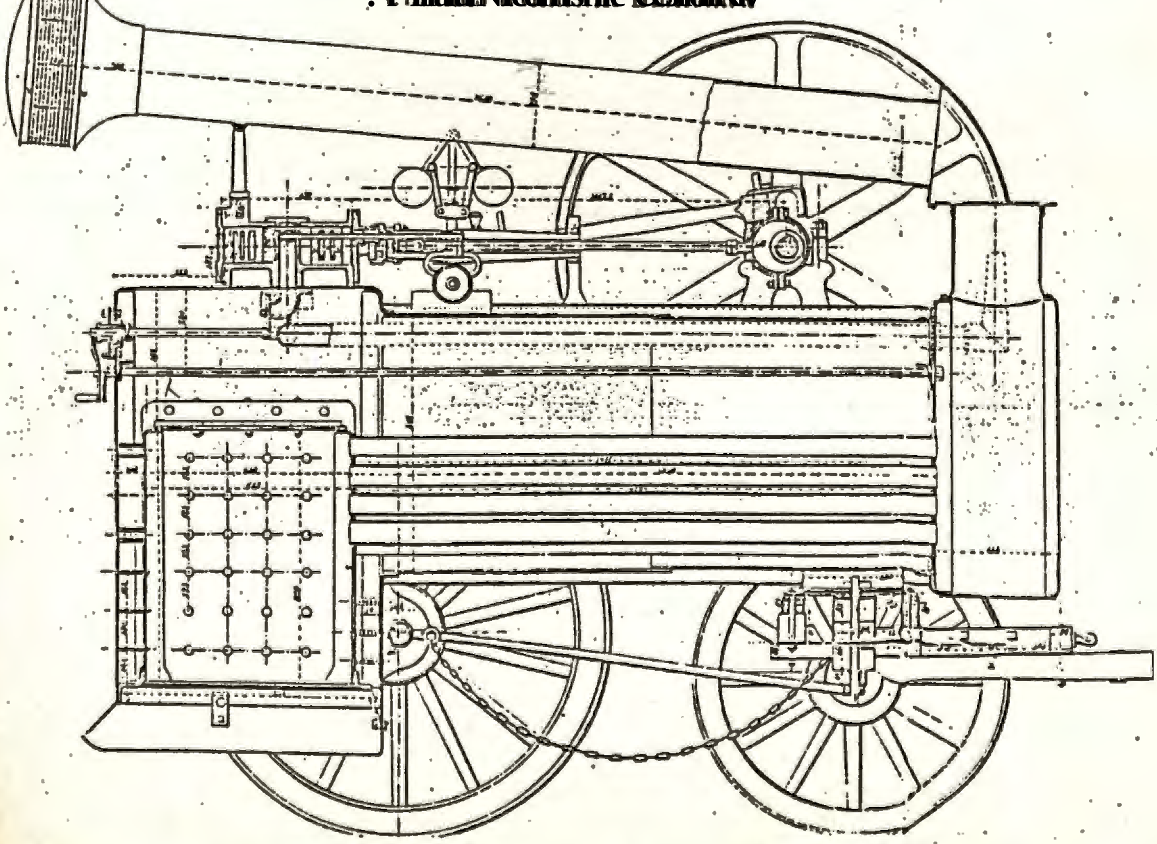


Fig. 6. A drawing of a locomotive design guide from 1871 (coloured)

" '‘V- ■. . i- -

í^'í-í:'iií^<A’S 3

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 149-161.

EDEN – A GENERAL-PURPOSE GRAPHICS EDITOR ENVIRONMENT

W.D. Fellner, F. Kappe

IIG, Institutes for Information Processing Graz,
Graz University of Technology and
Austrian Computer Society (OCG)

Abstract

Systems allowing the creation and manipulation of graphical information (so-called Graphic Editors) have become essential in various fields of applications (e.g. CAD, CAI, DTP, Vtx). At the same time the typical user of such a system has changed. Not computer experts, but designers, secretaries, technicians, teachers etc. are todays typical users of computer graphics, mostly on microcomputers. Obviously it would be desirable to have a common concept of graphics editing covering many applications.

EDEN (short for EDitor ENvironment) is a generic concept of object-oriented graphics editing: Many different application-specific graphic editors can be created using EDEN as their common nucleus. EDEN supports the objects and attributes defined by CGM/CGI, but in addition mechanisms are provided to extend the set of supported objects for a given application.

This paper illustrates the basic features and the extension mechanisms of EDEN-based editors and presents some already existing EDEN-based applications (for CAD, CAI and Vtx).

Keywords: Computer Graphics, Graphics Editors, CGM, CGI, EDEN

1 Motivation

The recent breakthrough of the microcomputer has also influenced the field of computer graphics. While in the past the processing of graphical information was limited to dedicated machines with expensive graphics hardware (such as plotters, digitizers etc.), the competitive marketplace of today forces microcomputers to provide powerful graphics facilities.

This was made possible by the availability of high resolution, multi-color raster displays at a reasonable price. On the input side the mouse made its way to the standard graphics input device of todays micros. And because of the increasing computational power of the microcomputer itself (16/32-bit processors, coprocessors, megabytes of memory...) more graphics applications are possible on smaller machines (think of Computer Aided Design (CAD) or Desk Top Publishing (DTP)).

At the same time the typical user of such a system has changed. Not computer experts, but designers, secretaries, technicians, teachers etc. are todays typical users of computer graphics. This leads to an increasing demand on software as well, especially on programs allowing the creation and manipulation of graphical information by the user - so called *Graphics Editors*.

These editors can be used for various applications, but a set of common features shared by well-designed graphics editors can be defined. From both the users and the software developers point of view it is desirable to have a common concept of graphics editing covering many applications.

EDEN (short for *ED*itor *EN*vironment) is a generic concept of object-oriented [1], interactive graphics editing: Many different applications can be created using EDEN as their common nucleus (Figure 1).

2 Basic Features of EDEN

EDEN itself consists of a set of routines that handle:

- Communication with the user (menus, mouse, keyboard...)
- Driving the graphics hardware (display objects, windows...)

[1] We will not deal with pixel-oriented programs (paint programs) here, although EDEN can also perform as a paint program by providing additional operations on the objects *Cell Array* and *Pixel Array*.

- Access to internal data structures (insert, delete, pick object...)
- Interface to the operating system (select, read, write file...)
- frequently used functions (segment transformations, rubberbanding, zooming...)

Obviously these functions are (in part) machine dependent.

These basic concepts implemented in the EDEN nucleus are incorporated into any EDEN-based graphics editor. The key features of such an editor are described below. In short, these are the features that we think a state-of-the-art and user-friendly graphics editor should offer.

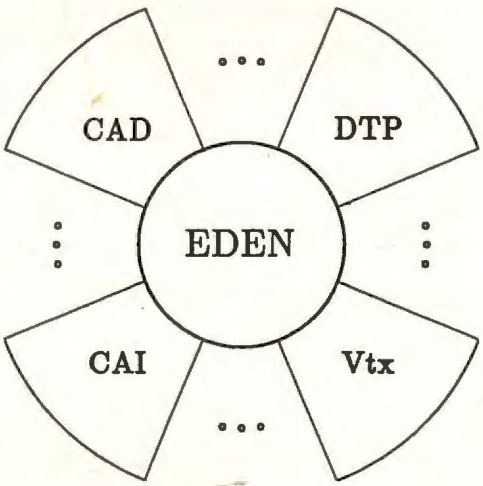


Figure 1: EDEN as a common nucleus for various applications

2.1 The Segment Concept

The creation of high-quality images is a time consuming job, even with sophisticated editing tools. Therefore the user will try to save some time by reusing complex sub-images in other images, if he is encouraged to do so by a system offering adequate functionality.

We will refer to these sub-images as *Segments*. Segments are kept in a kind of database by the system and may be used in arbitrary images by the user. In addition, 2D transformations may be applied to segments, thus increasing their flexibility. Segments may also be nested (segments containing segments).

For an image containing segments only the references and the transformation parameters are stored. Any modification of the segment —segments are edited just like ordinary images —will therefore result in a modification of all images referring to that segment.

Segments usually are created by picking some objects out of an existing image. The objects forming the segment cannot be accessed individually after this operation (the reverse operation – split segment – must be used to obtain the individual objects again). Only 2D transformations (any 3×3 matrix representing a homogeneous coordinate transformation) can be applied to the segment as a whole.

Segments have turned out to be one of the most powerfuls tool for image creation.

2.2 User Interface

A good user interface is essential for acceptance of the system by the user. It is not easy to define what a 'good' user interface is; however, there may be defined several features that a modern, interactive program should support:

- **Menus:**
  The user should not be forced to memorize commands necessary to work with the system. Instead the functions should be offered as *menus*. The structure of the menus, called *menu tree*, should have the following features:

  —*Small depth* (means more width). The user has to provide less input and more functions are offered simultaneously.

  —*Logical structure*. Similar functions should reside in the same menu, this makes functions easier to find.

  —*Uniqueness of functions*. It is confusing to reach the same functions via different paths in the menu tree. Therefore this situation should be avoided (only one way to get to a specific function).

  —*Clarity of function description*. This is a difficult goal to achieve, because there is only a limited amout of screen space available to describe the corresponding function. Usually one (eventually abbreviated) word must be enough. One solution is the use of *Icons* instead of words, but this should be used with care. A different solution is the availability of a *context-sensitive help function* that provides the user with a more detailed description of the functions of the selected menu.

  —*No redundant menus*. It is annoying to get an 'Are you sure (Y/N)?' prompt after every selection of a delete operation. Instead of having to confirm the selected operations it is more comfortable to have a global *Undo* function.

- **Graphical Input:**
  Graphical objects should be entered and manipulated by means of a graphical input device such as a mouse or a digitizing tablet. The user should not have to edit coordinates as text using the keyboard. The selection of menus using a graphical input device is already a generally accepted practice. The keyboard should be used only entering text constants (such as filenames) only.

- **Graphical Output:**
  The quality of the display is importeint for acceptance of the system. In this context quality does not only stand for the hardware characteristics (resolution, colors, refresh rate), but – even more important – for the user-support during manipulating the image on-screen.

  —*Rubberbanding* is a technique that helps the user to define an object. The term is derived from the way a line should be entered: after the first point is fixed, a line is drawn permanently from the fixed point to the (moveable) cursor, like a rubber band of which one end can be pulled over the screen (Figure 2a). The concept can be extended to all graphical objects; rubberbanding is extremely useful when manipulating axes (Figure 2b), axe chords, arc pies or splines (Figure 2c), where it is non-trivial for the user to tell the exact appearance of the object from its definition points.

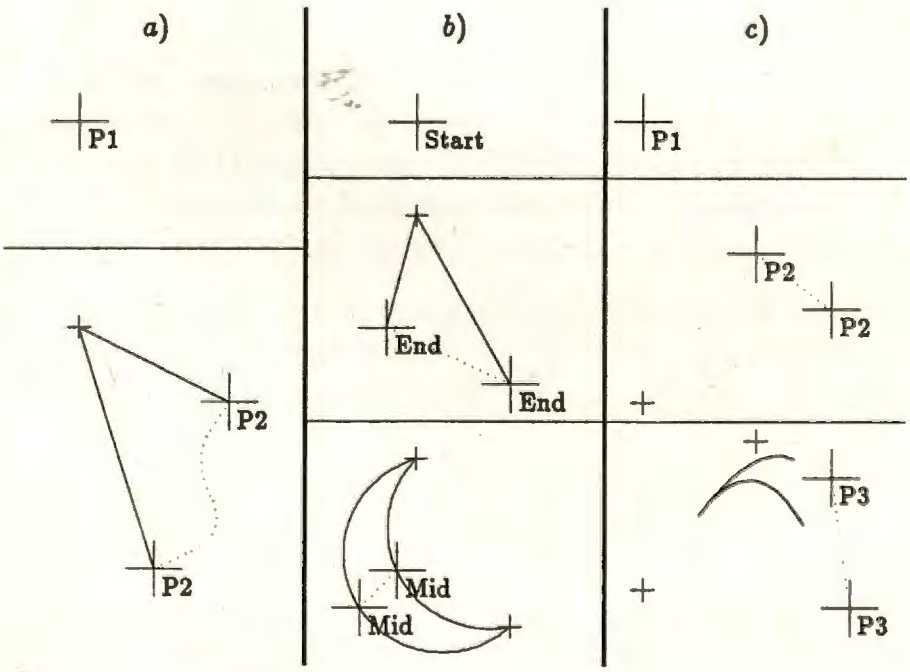


Figure 2: Rubberbanding line (a), airc (*b*), and B-spline (c).

— *Display speed* is not a precondition for rubberbanding only. Sometimes it is necessary to redraw the whole image (e. g. after deletion of an object). Obviously this should happen as fast as possible. Therefore most of the display algorithms should be implemented in hardware.

— *Zooming, Point snapping, Gridding:* These techniques free the user from counting pixels on the screen. Zooming lets you view and manipulate a magnified portion of the image. Gridding superimposes a rectangular grid (of variable size in x- and y-directions) on the image. Point snapping is similar, but instead of displaying a grid the screen positions, where the cursor can be located, are restricted to grid points. Both techniques facilitate editing of regular structures.

- **Undo function:**
  Humans change their mind or make mistakes. A system with high human interaction must be expected to provide a comfortable means of error correction. The number of actions necessary to correct a mistake should not be much more than to make the mistake. The easiest way to achieve this goal (for the user, not for the programmer) is a global *Undo* function that lets you take back any keypress (or mouse button press) and resumes at the situation previous to that keypress.
- **Configuration:**
  Different users may have different requirements. The system should allow the configuration of system parameters by the user.
- **Help function:**
  Most users prefer playing around with the program rather than reading manuals before using it. In order to prevent him from a frustrating experience an interactive, context-sensitive help function should provide the user with the information he needs. This *electronic manual* provides faster and better access to specific information than the printed one, making the latter obsolete.

Additional features such as *Action macros* (lets you group together a series of operations for later reuse), an *Automatic recovery* after system failure, creation of backups etc. can be thought of. EDEN supports all of the features listed above.

2.3 Graphics Standards

EDEN is not designed to support the editing of pictures in a single graphics standard. Existing standards for videotex images, CAI (Computer Aided Instruction) lessons, CAD and DTP applications are too different to support them all in the EDEN nucleus.

But EDEN adopts the basic concepts of existing and accepted standards such as GKS ([6]), CGM ([8]) and CGI([7]). Many of these concepts are found in the videotex image encoding standards CEPT ([1]), ECMA ([3]), and other standards. EDEN supports the following concepts of GKS, CGM and CGI:

- The primitives, GDPs and their attributes.
- The attribute handling (Bundles, ASFs).
- The segment concept.
- The workstation transformation.

3 Creating EDEN-based Applications

The main design goal of EDEN was not to support everything that might be necessary for some application, but to allow extensions of EDEN to do so. When creating an editor for a specific application, it may be necessary to extend the set of supported objects, to supply routines for manipulation of these objects and to provide an interface to existing software and graphics standards for that application. The necessary extension mechanisms are described below.

3.1 Extending the set of supported objects

EDEN provides three different, expandable sets of objects (Table 1):

Object type:	Atom	Compound	Segment
Created by:	Programmer	Programmer/Super-User	User
Definition:	C program	Program [a]	Interactive
Machine dependent:	Yes	No	No
Stored in:	EDEN module	User module	Picture file
Added at:	Link time	Run time	Reference
Examples:	Ellipse, Animation Sound	Arrow, Bathtub, Nandgate	Logo, Phone, House

Table 1: How new objects can be added at different levels.

[a] The definition language for compound objects is currently C, but the possibility of defining them in an object-oriented language (C++ or specially defined) will be further investigated.

- **Atoms** are the basic objects all other objects consist of. There are some built-in atoms (*Polymarker, Polyline, Text, Fill Area, Cell Array* and *generalized drawing primitives (GDPs)* like *Circle, Arc, Arc Chord, Arc Pie, Rectangle* and *B-Spline)* and additioneil application specific atoms. The routines to display and manipulate atoms are organized as a linked list. Only those objects that cannot efficiently be displayed by combining existing atoms should be defined as new atoms to keep this list short.

- Segments need no programming at all. The may be defined by the user. *Local segments* (are accessable only in the picture where they are defined) and *global segments* (may be referenced by any picture) may be created.
- Compound objects can be displayed using atoms but need more functionality than segments. The necessary programming may be done by the programmer of the application or by an experienced user (Super-User). The definitions of the compound objects for an application are machine independent and may therefore be added to the system at rimtime.

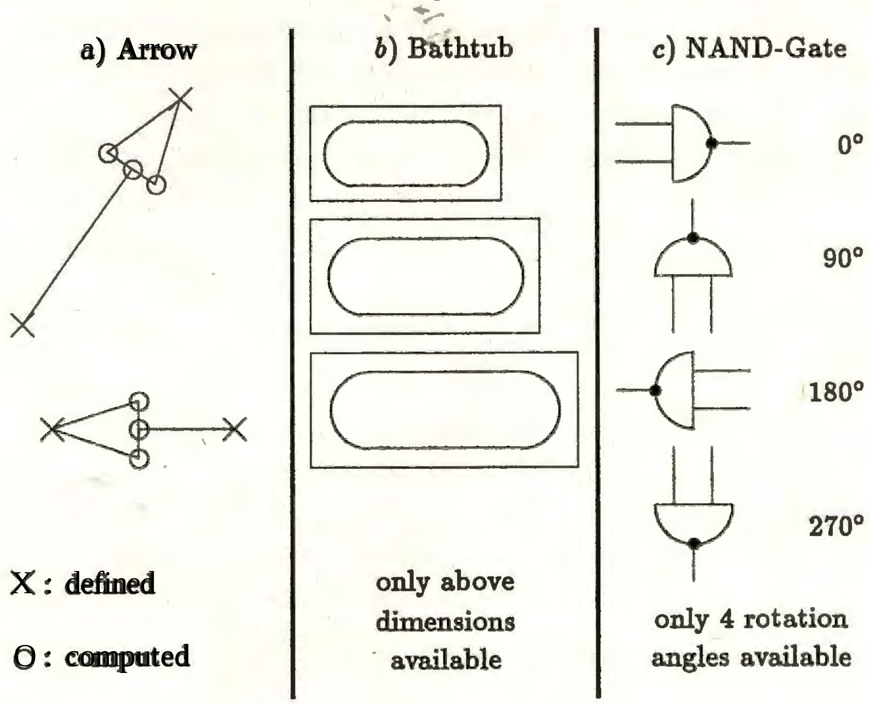


Figure 3: Examples of compound objects

Some examples are shown in figure 3. The arrow needs to have 3 points computed from 2 other given points and can be displayed using line and fill area. The bathtub can only be defined in available sizes. In addition, the display routine might check that the long side is beneath a wall. A price might also be associated with this compound object to compute the value of the installations in a room. In case of the NAND-gate, the size is constant and only 4 orientations are allowed. The display routine could also perform some application dependent checks.

3.2 Interfacing the World

Obviously pictures and segments generated by EDEN-based applications have to be stored somehow. Usually there exists an application-specific file format for this purpose, but use of that format may have disadvantages (for instance, the hierarchical segment structure might be lost, the encoding/decoding is slow etc.). Besides, an application-independent file format is desirable to interchange pictures across applications. Therefore EDEN supports both (see Figure 4): An application-independent file format called *Picture Interchange Coding (PIC)* and the encoder/decoder concept to provide an interface to application-dependent file formats.

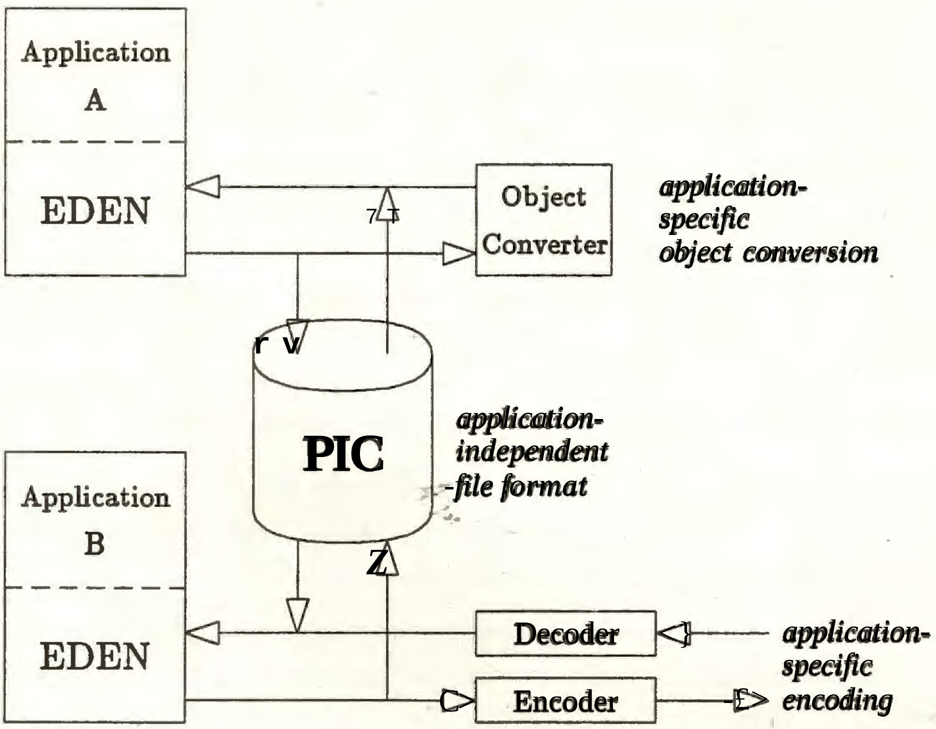


Figure 4: The World Interface

3.2.1. Picture Interchange Coding (PIC)

A file format for encoding pictures can be designed with a number of different objectives:

i) *Compactness:* The encoding provides a highly compact file, suitable for systems with restricted storage capacity or transfer bandwidth (Example: CGM character encoding ([8] part 2), CEPT videotex standards ([1],[3])).

ii) *Processing Speed:* The encoding uses binary data formats similar to the representation used within the computer in order to minimize processing overhead in reading/writing the data file (Example: CGM binary encoding ([7] part 3), PIC).

iii) *Human Readability:* The representation of the data is easy to read, type and edit using a standard text editor (Example: CGM Clear text encoding ([7] part 4)).

iv) *Extensibility:* The encoding allows future growth in a natural way. The coded file should be sharable by a number of applications - even if they do not all understand everything that is in the file (Example: PIC, TIFF ([4])).

The key feature of PIC-Files is that they may be generated by different applications that support different atoms and compound objects. This implies that an application has to ignore all data not related to the internal list of supported objects. All information stored in a PIC-File is contained inside so-called *blocks:*

$$\boxed{\langle block\rangle ::= \langle ID\rangle \langle length\rangle \langle data\rangle\ [0]}$$

The *ID* is a 4-character (a 32-bit longword) token that identifies the type of the block. The *length* of the following data in bytes is also encoded with 32-bit. The meaning of the *data* following depends on the type of the block (the ID). In particular, it may contain other blocks, thus introducing a hierarchical file structure. An optional zero byte is used to align blocks on word boundaries.

This block-oriented file structure has the following advantages:

- The ID is a mnemonic name for the type of the block. When adding new blocks name clashes have to be avoided. The length of the ID (4 characters) is a compromise between ease of adding new IDs and processing speed.
- The PIC-Reader may easily skip over blocks with unknown IDs by using the length-of-data information. It is also possible to scan the file at high speed without interpreting the data.
- The data field is fully transparent (any sequence of bytes may be stored here).
- The smallest unit of processing is a byte. Blocks are word-aligned, increasing the processing speed on most modern processors.
- Blocks can be nested. For instance, pictures may contain segments, the segments are composed of atoms. In addition, so-called *Environment blocks* may be used to define common properties of following blocks on the same level (PASCAL-like scope rules).

3.2.2 Code Converters

The images created by an EDEN-based graphics editor will usually have to be processed by some other application, get printed (plotted) or stored somewhere (e.g. a network) in a standardized code. On the other hand, one might want to edit the output of some other application with the EDEN-based editor.

Because these applications cannot be expected to understand PIC-Files, the images have to be converted to (or from) a different picture description format. These *code converters* can be divided into 3 groups (see figure 4):

1. *Decoders* convert pictures in some application-specific format to standard PIC-Files. If any features of the input are not supported by EDEN, they have to be mapped to something similar or new EDEN objects have to be added by the mechanisms described in section 3.1.
2. *Encoders* convert PIC-Files to some application-specific representation. Encoders can also be used to get a hardcopy of a picture by generating appropriate printer codes.
3. *Object converters* convert PIC-Files to PIC-Files, but replace some EDEN objects by different ones. This might be necessary to process pictures created by different EDEN-based applications (e.g. by splitting compound objects of the 'source application' to atoms of the 'teirget application').

A code generator is a stand-alone program converting data from standard input to standard output. An Encoder, for example, is used by EDEN by piping the PIC output to the encoder rather than to a file. The output of the code converter may be piped elsewhere (e.g to another code converter, a file, a printer or - in the case of an object converter - back to the EDEN PIC-input (figure 4)). Code converters may also be seen as translators from one (picture description) language to another - a compiler - and can be written using compiler generation utilities such as *LEX* and *YACC*.

4 Existing EDEN-based Applications

4.1 Generic Videotex Editor

The first prototype for EDEN-based graphics editors was an editor for creating videotex frames [9]. This editor runs on PC-AT compatibles with a special graphics controller and a special videotex board under MS-DOS. The editor allows editing of CEPT-C0 alphamosaic text not covered by the EDEN nucleus and related objects. This editor also uses a number of code converters:

- A decoder for interpreting alpha-geometric videotex frames (both standards are supported). Because of the coding structure of the two standards the data may be processed by one decoder.
- Two encoders generating videotex encoding acording to the above standards plus an encoder that produces printer codes for a kyocera laser printer. These printer codes may be used with TEX documents[3].
- An object converter for converting alphageometric objects to alphamosaic objects (DRCs).

More encoder/decoder pairs may be added to support more videotex standards or printers.

4.2 CAI Editor

An editor for creating CAI lessons was also modelled using the EDEN nucleus [5]. Basically, this editor introduced the following extensions:

- Special atoms had to be defined for the support of animation, sound and answer judging.
- The editor contains additional functions to organize and maintain a lesson (a lesson is a set of single frames).
- In order to check the interaction of frames a lesson executor is also contained.
- Encoder/Decoder pairs are needed to support different courseware formats.

4.3 CAD Editor

Another editor built on top of the EDEN nucleus was a special CAD-application in the area of energy and communication networks.

Besides porting the system to another hardware (Apollo workstations) and to a multitasking environment the main extensions were:

- The interface to a relational data-base holding all geometrical information.
- The interface to several processes performing various computations on the generated drawings (redundancy checks, application specific computations,...)
- The support of additional input devices (scanner, video camera, special graphic tablets) and output devices like special plotters.

[3]As you might have guessed, this paper was prepared using TEX and EDEN (for the figures).

Literature

1. CEPT: *Videotex Presentation Layer Data Syntax (Issue 1); Part 2 – Geometric Display*; T/CD 6.1, Innsbruck, Austria (May 1981).
2. CEPT: *Videotex Presentation Layer Data Syntax (Issue 2); Part 1 – Alpha-Mosaic Display; Part 4-6 – Define DRCS, Colour, and Format; Part 8 – Reset*; Revision of T/CD 6.1, Cannes (Sep 1983).
3. CEPT: *Videotex Presentation Layer Data Syntax (Issue 2); Part 2 – Geometric Display*; T/CD 6.2, (Nov 1987).
4. Davenport T., Vellon, M.: *Tag Image File Format (TIFF) – Rev. 4.0*; Aldus Corp. & Microsoft Corp. (Apr 1987).
5. Holweg G.: *Ein komfortables Editiersystem für CUU*; Masters Thesis, IIG University of Technology Graz, Austria (May 1988).
6. ISO: *Information Processing Systems – Computer Graphics – Graphical Kernel System (GKS) – Functional Description*; ISO IS 7942 (1985).
7. ISO: *Information Processing Systems – Computer Graphics – Interfacing techniques for dialogues with graphical devices (CGI), Part 1-6*, ISO TC 97/SC 21, N 1179, Working Draft (May 1986).
8. ISO: *Information Processing Systems – Computer Graphics – Metafile for the Storage and Transfer of Picture Description Information (CGM)*; ISO IS 8632 (Aug 1987).
9. Kappe F.: *Design und Implementierung eines EDEN-basierenden Bildschirmtext-Editors*; Masters Thesis, IIG University of Technology Graz, Austria (Feb 1988).

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 163-170.

THE INTEGRATED GEO-INFORMATIONSYSTEM INFOCAM/ORACLE

H. Brückler, F. Brückler

Institute for Machine Documentation
Research Center Joanneum, Graz

Abstract

Existing systems for manipulating geographical data, s.a. maps, plans, contoir lines, satellite-pictures, air-pictures, etc. have often separate modules for geometrical data and thematical data. Some systems handle thematical data with relational databases, but graphics only project by project with normal file-handling. This method causes a lot of disadvantages.

Our system INFOCAM is a common project between the Swiss company Kern, the Institute for image processing and graphics, and our institute. INFOCAM is totaly based on the relational database ORACLE V 5, so geometrical and thematical data are both stored in one and the same database. And therefore we do not have any file-handling to manipulate data.

In this paper we describe the architecture of the INFOCAM-System, and the concept of the database with the most important tables and their relations.

1. INTRODUCTION

1.1 General

The most existing systems for manipulating geographical data, s.a. maps, plans, contour lines, satellite-pictures, air-pictures, etc. often have two differnt ways to manipulate geometrical and thematical data. Some systems handle thematical data with relational databases, but graphics are stored with normal file handling project by project. This method causes a lot of disadvantages.

In the geometrical part of the system you cannot merge parts of existing different projects to one new project (e.g. you cannot merge the housing-estate from project one with planned streets of project two). A second disadvantage often is the very loose connection of geometrical and thematical data. Many systems only allow the manipulation of a whole thematic, but not of a part of it (e.g. it is not easy to show all streets wider than 12 yards and build with concrete). A third disadvantage: You have no direct relational access to geometrical data (e.g. highlighting all areas with more than four edges).

Our system INFOCAM is a common project between the Swiss company Kern, the Institute for image processing and graphics, and our institute. INFOCAM is totaly based on the relational database system ORACLE V 5, that means geometrical and thematical data are both stored in one and the same database. And therefore we do not have any file-handling to manipulate data.

1.2. Advantages and disadvantages of relational databases

Why do we use a relational database to store the data? Other systems often use databases only for thematical data, but not for geometrical information. We have decided to hold the whole information in the database (there exits no file for storing data) because then you automaticaly have following advantages:

- sheetfree working
- same dataarea for all projects (the database), and therefore no problems with combining new projects from older ones
- at any time consistent data
- no file management neccesary
- multiuser capability is totaly managed by the database
- users and their access rights are also managed by the database
- easy way to export and import data, and to exchange data with other systems and databases
- relational access also on geometrical data
- a direct connection between geometrical and thematical data, which means, that you can query information by combining thematical questions with geometrical questions
- hardware independancy for datastorage, because ORACLE is available for nearly all types of hardware
- it is easy to use the data in other programs via the ORACLE programming tools
- distributed systems are directly supported via ORACLE

The disadvantages of using relational databases are:

- first of all, you have to buy a relational database, and so the customer has to pay a little bit more for this system
- on the other side, it is a new concept for the developers; there are no general known datastructiures for storing geometrical or graphical and thematical data; it is a great work to test and develope new datastructures, with no guarantee that it will ever work;

2. COMPONENTS OF THE SYSTEM

The whole system consists of about 9 great modules, of which I will discribe only the important ones.

The main module is called IMAGE, which is used for interactive graphical work. Here the inputs, updates and deletes are done. A submodule to IMAGE is INCOME, a module for manual digitalizing of maps etc.

An other important module is called IMPRESS, to create interactive maps and plans.

ATOS is used to manipulate contour lines or triangulations.

All these modules (and some others not described in this paper) are based on the module INFOCAM/ORACLE, which stores all the data - geometrical and thematical - manages the data-transfer, data-exchange between the modules, the user controll, access control, backup control and a lot of other management functions.

3. COMPONENTS OF THE DATABASE-PART

The part INFOCAM/ORACLE is - as you can see above - a very large and complex module. So we have decided to divide it into three submodules called DB-MANGAGER, DB-INFO and DB-USER.

3.1. DB-Manager

This part is the main control program, which manages the users, their accessrights, the projects stored in the system or backuped on tape, initialises new thematical tables, etc.

3.2. DB-Info

This module is used for fast querying the database to get an overview over the stored data, or to make a fastplot of any geometrical information. It is also possible to retrieve only textinformation with or without corresponding geometrical information. You can also query by graphical input, e.g. you can use the mouse to point on an unknown object, which then is described.

Generally DB-Info is used to retrieve information and data, but not to manipulate any information permanently.

3.3. DB-User

This is the main part of the whole system. With this module all the manipulations, inputs, and deletes are handled. This module has no own interface to the user; it is only an internal module, which manages the data to the manipulating modules such as IMAGE, INCOME, etc.

As the whole system is a multiuser system, this module also handles the access control and the temporary access restrictions, if one user wants to manipulate the data used by an other user at the same time.

4. BASE CONCEPT OF THE DATABASE WITH THE IMPORTANT TABLES

It is not possible to discribe all used tables and concepts in this short paper. But we will explain the most important tables for storing the geographical information, and some aspects of storing the thematical data.

The most important elements you have to manage are: points, edges, symbols, lines, regions and objects.

Points and edges are the graphical elements, symbols, lines, regions and objects are logical elements, e.g. a street is a line which consists of several edges. Also a region consists of one or more edges. An object is a set of symbols, lines, regions and other objects. This definition allows also a hierarchy of objects with no logical restrictions.

Each element from point to object has an identifier called 'id', which is a unique key over the whole database. This 'id' is the connection between the geometrical (symbol, line, region, etc.) and the thematical data (e.g. trigonometric information for a point, area of a region, meaning of a symbol, and any user defined information, e.g. water quality for a line which represents a river).

In the following we describe (in a shorted form) the main tables for the geometrie. (There are many other tables, or tables with more attributes than described here.)

4.1. Point-Table

Table POINT:

ID	Identifier (20 digits, with 10 digits for an approx, x-coordinate and 10 for an approx, y-coordinate)
X	X-Coordinate for the point
Y	Y-Coordinate for the point
Z	Z-Coordinate for the point
NET_ID	Id of an area in which the point lies
P_FLAG	Information about the type of a point
etc.	

4.2. EDGE-Table

Table EDGE:

ID	Identifier
START_P_ID	Id of the startpoint
END_P_ID	Id of the endpoint
BULK	X/Y Coordinates of the startpoint, the points between start- and endpoint and the endpoint (inner points of an edge are not stored separetly in the point table)
POINT_NUM	Number of points in the bulk
NET_ID	same meaning as point net_id
etc.	

4.3. THEM_OF_EDGE-Table

Table THEM_OF_EDGE

ELE_ID	ID of an element (e.g. symbol, line, region, etc.)
E_ID	ID of an edge
ORIENT	Orientation of this edge
etc.	

4,4. TH_ATTR-Table

Table TH_ATTR

ID	Identifier
LAYER	e.g. name of specific theamtic
ATTR1	first description of this object
ATTR2	nearer description of this object
XWMIN	minimum boimding rectangle of this object
XWMAX	
YWMIN	
YWMAX	
etc.	

4.5. Examples

An example of drawing a street may be the following:

Highway A2 as the name of the street would be stored in TH_ATTR.LAYER. A description of this highway is TH_ATTRATTR1 and TH_ATTRATTR2. The maximum area is TH_ATTR.XWMIN to TH_ATTR.YWMAX. With th ID of TH_ATTR.ID you have to go in the table THEM_OF_EDGE.ELE_ID to get the id's of all edges for this object Highway A2 in THEM_OF_EDGE.E_ID. With this THEM_OF_EDGE.E_ID you have to go to the edge-table (EDGE.ID) and with the information in EDGE.BULK and EDGE.POINT_NUM it is possible to draw the wanted object.

A SQL query statement to retrieve this information would have the following form:

```
SELECT    e.point_num, e.bulk
FROM      edge e, them_of_edge t, th_attr a
WHERE     a.layer    = 'Highway A2'
AND       t.ele_id  = a.id
AND       e.id      = t.e_id;
```

To retrieve the pointinformation of all points of this special street, following query would be necessary (let us assume that there is a table point_info with the wanted information of a point):

```
SELECT   p.info1, p.info2, ...
FROM     point_info p, edge e, them_of_edge t, th_attr a
WHERE    a.layer   = 'Highway A2'
AND      t.ele_id  = a.id
AND      e.id      = t.e_id
AND      (p.id     = e.start_p_id
OR       p.id      = e.end_p_id);
```

It is no place and no time in this paper to describe further tables for e.g. user-control, more thematical tables (such as POINT_INFO), or multiuser techniques. Its is also no place to describe the indexes and clusters used by the ORACLE database to have a fast access to the data.

But we will only give a short statement above the time consuming of the above SQL-statements:

It is (much) faster to retrieve data from ORACLE with a query of the above type, than to read the same prepared data from a file under VMS on a Micro-VAX.

5. WHY DO WE USE ORACLE?

First of all, we have choosen ORACLE as our favorite, because it is a great and well known company in the field of database systems.

Second, the ORACLE database is a strict relational database with standard SQL-interface. Also the availability on nearly all hardware- and computersystems is very important for a portable system. So you can change the hardware or use a greater machine without problems.

It was also very important for our project to have several programming interfaces to nearly all commonly used programming languages. A special advantage of ORACLE are the Precompilers, so you can use the known SQL-statements without change in a Fortran or Ada program. This feature speeds up the programming of database using procedures dramatically.

6. CONCLUSION AND AN OUTLOOK TO FURTHER DEVELOPMENTS

We plan in the nearer future the use of a Vax-cluster, with a distributed database, divided into local and global data. Local data is hold in the workstation, global data is stored in a greater central machine. For the user this concept makes no difference to the old one, because the database manages (unvisible for the user) the datatransfer and storageplace.

It is also planed to use this software package on a heterogen net, e.g. the IMAGE-programs on a Vax-GPX and the database e.g. on a UNIX-machine.

References:

[BRÜ] Brückler Helmut, Brückler Eva
IGIS-Datenbankkonzept unter Verwendung von ORACLE V 5
Vortrag, DIBAG/FGJ, Graz, Feb. 1988

[FRA] Frank A.
Datenstrukturen für LIS, semantische, topologische und räumliche Beziehungen in Daten der Geo-Wissenschaften
Inst. f. Geodäsie u. Photogrammetrie a. d. ETH Zürich

[MEI] Meier A.
Methoden der graphischen und geometrischen Datenverarbeitung
Teubner, Stuttgart, 1986

[ORA] ORACLE V5
Manuals
ORACLE Corporation, 1987

[PHO] PE&RS Vol. LIII/Oct. 1987
Photogrammetrie Engineering and Remote Sensing
American Society for Photogrammetry and Remote Sensing
Falls Church, USA, 1987

[SPA] Proceedings of the
International Symposium On Spatial Data Handling
Vol. 1 and 2
Zurich, Switzerland, 1984

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 171-176.

PIGALLE, AN IMAGE AND TEXTUAL DATABASE MANAGEMENT SYSTEM

Kópházi József, Molnár Csaba

Computer Research Institute and Innovation Center

Budapest

ABSTRACT. We introduce in this paper a new PC-software namely PIGALLE (electronic Picture & text GALLEry), that integrates large volume, free-format textual and medium resolution true colour pictorial information into a complex database for later intelligent retrieval. Employing effective database-handling and image-compression algorithms, PIGALLE offers the user a large amount, valuable and well-structured information. PIGALLE can handle numerous input peripheriais, e.g.:xameras, scanners, ultrasonic scanners, and output peripherials, e.g.:WORM, plotters, displays and so on. Due to these rich information sources, elegant displaying environments, and the high capacity archivation media, PIGALLE may serve a wide range of applications in medicine, education, culture and industry.

KEYWORDS. Database management, Image Compression.

1. INTRODUCTION

In recent years the appearance of "fast" PC's and some new personal computer peripheriais makes it possible to develop revolutionally new systems which were previously available only on mainframes or at least on minicomputers. Two types of peripheriais are interesting for us, the frame grabbers and the optical discs. The frame grabber is a plug-in module of a PC, which enables the user to capture an image, more precisely a frame generated by a standard video equipment such that camera, video tape recorder or angiotron. Then the image is to be sampled, digitized, processed and stored. In this way real images become attainable by personal computers. On the other hand, the high capacity optical discs makes it possible to store and manage effectively the large amount of data generated by a frame grabber. On the base of that hardware elements new database management philosophy and database management systems have been developed. We are to draw attention to some problem of this topic, and to outline our solution which has found shape in a real database management system, PIGALLE. In the second chapter we give details the problem of image capturing and compression in order to be able to store a lot of good quality pictures. In the third chapter the structure of the mixed database and the knowledge base will be outlined, and at last in the fourth chapter the natural applications will be listed.

2. ABOUT IMAGE CAPTURING AND COMPRESSION

Images can carry a lot more information than textual explanation. Taking a quick glance at a picture may often tell the observer more than reading through pages of written description. The trade-off is the relatively long time and enormous disc space consumed for electronic processing and storing of images. A frame grabber usually converts the monochrom analog video frame into a 512x512x8 bit digital data. The true colour video frame is converted into a 512x512x24 bit digital data. Even if a high capacity storage media could be used for storage it desirable to compress images in light of the following standpoints:

- the cost of storage, which is in direct ratio to the size of data should be minimized,
- the time necessary to display the image has to be subjectively acceptable.
- the quality of the decompressed images is expected to satisfy the demand of possible applications,
- the hardware requirements of displaying the image be as cheap as possible.

The redundancy contained in the natural images offers the possibility of compression. Numerous compression algorithms are known, but only a few satisfy the requirements listed above. We have chosen two base algorithm, one for camera images and one for scanned images. The first is a block-code which converts constant length blocks of the picture into constant length blocks of the code, independently the actual value of the image. Consequently the rate of the code which is the ratio of the size of the coded data and the original image is constant. The trade-off of this coding method is the image distortion, the advantage is the good compression ratio and the very simple decoding algorithm. The second one is a variable length code, which associates variable length codewords to variable length blocks of the scanned image. Of course, the rate of this code varies in a wide range, depending on the images being encoded. The advantage of this method is that the image can be decoded without any distortion, the trade-off is the relatively complicated and slow decoding algorithm, and the variable rate which may cause buffering problems. In extreme case the compression may result increased data. In the following we explain in detail the characteristics of theese algorithms.

2.1. TRUE COLOUR AND MONOCHROM IMAGE COMPRESSION

The Storage capacity needed to archive 512x512x24 bit digital true colour image without compression would be 768 KBytes, and 256 KBytes for monochrom image. PIGALLE compresses theese datas to 66 KBytes and 32 KBytes respectively, which means 12:1 and 8:1 compression ratios. During information retrieval the coded data may be decoded and the restored picture will be of slightly lower quality, that is hardly noticeable visually. The coding algorithms consist of two steps. The first one is a version of the so-called block truncating coding (BTC) algorithm, which can be considered as a moment-preserving, local quantizer. This means that the picture is divided into blocks, each of whose pixels are individually quantized to two levels such that the block sample mean and variance are preserved. In the second step the gray levels for monochrom images and the colours for true colour images are quantized in the following way: We illustrate the method for true colour images. The three dimensional colour values appearing in the locally quantized image are considered as points in the three dimensional Euclidean space. Then the space is splitted into disjoint cells, and a representative element is chosen for every cell. All the colour values contained in the same cell are encoding into the representative element of this cell. More precisely, the two colours of a block are encoded by the indicies of the representative colours of cells containing theese colour values. We have sketched roughly the coding method which consisits of numerous "little algorithms", but deeper explanation is not possible here. The interested reader can find much more details about theese algorithms in the literature (see ref. [1-4]).

As an option, PIGALLE has two encoding functions.The first one combines the two steps sketched above, the second one uses only the first algorithm. The first function is more economical, as the information requires less space to store, and the decompressed image can be displayed via a single frame grabber, as it contains only 256 different colours. However, this also means a significant loss of color information which is visually noticeable in most cases.The second, not combined method results in a code file 1.5 bigger than the first one, and three frame grabbers must be installed in order to display the decompressed picture. The loss of information in this case is less significant, most times it is visually unnoticeable.

2.2. COMPRESSION OF SCANNED, BINARY IMAGES

Two level, scanned images can be input via a 200 or 300 dpi scanner. The storage requierment depends on the actual resolution. PIGALLE applies the CCITT T.6 standard coding/decoding method. The compression ratio varies between 1:2 and 1:30, without any loss of information. The actual ratio is dependent on the original image.

3. DATABASE MANAGEMENT IN PIGALLE

The primary goal is to make the data accessible to a particular textual and connected pictorial information from a large volume database regarding WORM's and CD_ROM's large storage capacity and their special physical features. First of all it needs a structured database including documents -article texts, pictures, encyclopedic entries ...etc,- dictionaries, indexes. On the other hand we need a filing system for handling these information both on magnetic and on optical storage media optimally.

3.1. FILING SYSTEM

There are two essential standpoints for an optimal filing system in such applications;

- Because of textual data entries the length of an index key can be varying between wide range.
- The applicable indexing techniques depend on both the needs of changes and on the storage media. While in dynamic databases, tree structures, such as B-trees have been used as indices because of their ability to handle growth, in relatively static databases hashing has been used for fast access. Because every picture entry even if it has been compressed increase the size of database in much more degree than other connected information, therefore picture file resides on optical media during database building, make possibility of more disks pictorial database, and all other data reside on winchester. PIGALLE use a B+ -tree indexing technique, handling variable length records and index keys. User can define the order of characters for indexing, matching it to any national character sets and alphabet.-r-

3.2. DATA STRUCTURES

The database of PIGALLE is made up of three logical units:

- Text base (documents,articles);
- Picture base;
- Knowledge base which defines dictionaries, the relations between them, and controls access to the actual contents:

Dictionaries:

- Title dictionary:	Each picture and text element has a unique title which identify directly all entry in textbase and in picture base. The simplest form of retrieval is browsing through this dictionary or its subset matching a mask.
- Descriptor dictionary:	Descriptors are all of "important" words, expressions which a document can be identify with. They carry the most important information for logical data retrieval. Unlike the titles, the descriptors, however, are more than simply a collection of words and expressions, the logical structure of the database can be define by them.
- Synonym dictionary:	Synonyms of all existing descriptors can be defined in this dictionary, and these synonyms will equally be included in the search for the original descriptor - or vice versa. An existing database can thus have multiple query structures by providing synonyms.

Data retrieval uses the connections between database entries. This partly means direct connected elements to a document (article/picture) such as its title, connected other title, its descriptors, abstract. On the other hand it means logical connections between these elements (descriptors) recorded in knowledge base. Descriptors can be arranged in many logical graf structures similarly to "Thesaurus" techniques. Any of them may be marked as group descriptor. Group descriptors, as their name suggests, refer to other descriptors, which may be group descriptors themselves as belonging to one logical group. This result in a logical structure of groups and subgroups of descriptors, which serves multiple purposes:

- To meet the needs of special user applications, the logical structure of the database can be defined before entering the actual data (article/picture). This speed up data entry make possibility of automatic descriptor selecting and indexing as the text is appended to the database.
- The structured access to the descriptors eases data retrieval as well, as it offers all query options related to a given subject when selecting the concrete search creteria. This is especially useful in educational applications, or when retrieval is based on broad, general points of view.
- The structure of the descriptors can be modified, reorganized at the time of data entry as well as in a separate maintenance phase. This way the same database can reveal various internal logical relations.

3.3. DATA RETRIEVAL

Two important questions must be answered before retrieving information from such databases.Firts, are you retrieving text data or documents? Retrieving documents is generally less precise than direct data retrieval, and there's no telling if you'll find the right answer to specific questions such as "What is the second largest river in South America?". From this viewpoint PIGALLE has a document retrieving system. The second quiestion to be considered, "will the retrieval software be geared toward browsing or searching for documents?". Searching is most often used because harnesses the power of data's efecironic nature. You can search for various word combinations and, with a properly indexed database, the software will quickly find the documents that contain them. Then, you can peruse those portions of documents that are likely to contain information relevant to your search. Browsing, on the other hand, allows you to enter the database at any point and view documents on the screen. Searching moves from "what" to "where" (as in what particular word combination is located where in the database.) Browsing moves from "where" to "what" (as you browse in a particular location in the database, you want to see what information is found there [7].) PIGALLE offers both boolean search (you can search for descriptor A AND descriptor B, or for descriptor A OR descriptor B or its combinations) where also any logical descriptor-tree structures can be used for building up searching mask, and browse function in the title list, either in selected parts by searching, or in entire title dictionary (see ref. [5-7]).

4. APPLICATION AREAS

At last we list some of the application areas where PIGALLE can be used successfully:

- Maintaining product-catalogues, visually keeping track of spare-parts, documentation in trading, manufacturing or design companies.

- Filing and combinining X-ray or ultrasonic images with additional information in order to support medical diagnosis or simply to build up client history.

- Creating educational materials, where the pictorial information is combined with textual explanation.

- Cataloging, documenting, archiving precious objects in museums, galleries...etc.

REFERENCES

[1.] *E.J.Delp and O.R.Mitchell*, Image compression using block truncation coding, IEEE Tr. on Commun., vol. COM-27, September 1979.

[2.] *P.Heckbert*, Color image quantization for frame buffer display.SIGGRAPH 1982., Proceedings, pp. 297-207.

[3.] *D.R.Halverson*, On the implementation of block truncating coding algorithm, IEEE tr. on Commun., vol. Com-30, November 1982.

[4.] *G.Campbell, T.A.DeFanti, J.Frederiksen, S.A.Joyce, L.A.Leske, J.A.Lindberg and D.J.Sandin*, Two bit/pixel full color encoding, SIGGRAPH 1986., Proceedings, pp. 215.

[5.] *D.E. Knuth*, The Art of Computer Programming, Vol.3. 1973.

[6.] *C.S. Ellis*, Concurrency in Linear Hashing. ACM Trans. Database Syst. Vol. 12 No. 2, June 1987.

[7.] *B. Brewer*, The Look and Feel ... and Sound of the User Interlace. CD-ROM Review July/August 1987. pp. 26-31.

ARTIFICIAL INTELLIGENCE

LIMITS OF LOGIC - COMPUTER EPISTEMOLOGY *

T. Vámos

Computer and Automation Institute,
Hungarian Academy of Sciences,
1132 Budapest, Victor Hugo u. 18-22.,
Hungary

Abstract. This paper discusses why the analysis of our limits in knowledge representation and world comprehension is so important in our electronic information age and why the very ancient basic problems of philosophy are returning at a new level. The problems are divided into three related areas: uncertainty, logic and language, which all direct toward the complexity issue. The logic theme is discussed in more detail. Last, some approaches are indicated.

NEW RELEVANCE OF OLD PHILOSOPHICAL PROBLEMS

Ancient Greek philosophers realized the difference between the real world, our perception of that and the way how people can express this, communicate the perception of reality. The same basic problems returned at any revolutionary achievement of discovering new facts, relations of nature and at each revolution of communication technology. Science is the history of science itself, this was the idea of Goethe and the New Paradigm concept coined by Kuhn on the turns of scientific methods relates to these revolutionary variations on the same theme.

Computer and communication age provided a radical new and necessary look at this triplet of world and knowledge representations. We represent our perception on reality in a machine program which reacts to our reality in a real time mode or at any rate in a more and more uncontrollable way. This uncontrollability lies in the exorbitant complexity of reality and exorbitant complexity of the programs themselves much beyond human comprehension. Emergency management of nuclear power plants, other large scale highly dangerous processes, monitoring of respiratory and other vital functions during a critical operation, dangerous maneuvers in aviation, decision processes in highly complex economic and social systems are and will be more and more supported, automatically controlled by knowledge-based systems. This means that the investigation of all possible relations of reality and representations was never before as crucial as now. We encounter this problem as our professional

* To appear also in Proceedings of the 6th IEEE International Workshop on Languages for Automation.

responsibility in a period when the scientific and software market is full with overstatements of possibilities and the public opinion generally underestimates its relevance for the near future.

Using the idea of Teilhard de Chardin that the animal knows it but does not know that it knows, we can add that it does not know what it does not know. As human verbalization of experience started, the early beliefs, origins of religions did just that: extrapolated all unknown knowledge to totems, spirits, Gods etc. Writing enforced a much more consistent mapping of knowledge, conceptualization, relational connections - after a long progress came a rapid flourishing, this was the acme of the ancient Greek philosophy. The cognition was used within a small circle of erudited people and mostly for purposes which were not in direct contact with the practical life. Printing distributed knowledge among broad, physically less related communities. Simultaneously achievements of natural sciences in a practical framework of scientific methodologies, metasciences promised an unlimited possibility of comprehending and mastering everything. The realm of unknown was visibly shifted towards the horizons of unlimited infinity, the Age of Reason supposed to enable mankind mastering the Totality. This overlook of a historical trend is of course a terribly superficial sketch and a one-sided interpretation of contradictory, fluctuating and long processes. It is used here as a hint, indicating the intrinsic relations of representation methods and human thinking.

Progress of this century's natural sciences, mathematics in parallel with their enormous results have indicated several limits of knowledge, bounds of the comprehension of human mind, faced with the complexity of nature. We shall not get into details of this subject but more into the other side, how the representation of knowledge is limited by representation methods used by our computers. This is our new and practically very relevant relation to our knowledge about our knowledge. A clear confrontation with these facts provides a research program, direction of extending representation methods, circumvention of difficulties. This attitude is just the opposite of the primitive man frightened by the realm of unknown: it is a conscious and dignified position to contrast with the neoprimitive irrationalism and its counter-part, the arrogant simplistic rationalism. All these evolved to primary importance as these representations intruded into any of the human activities.

UNCERTAINTY AS A BASIS OF OUR UNCERTAINTY

The problem area can be treated in three interrelated topics: uncertainty, logic and language. Uncertainty can be taken as the first one because it mostly refer to the sensory, experimental input data. The development of human thinking on uncertainty is the subject of other occasions, we mention only a few conserns.

It is not by chance that the concepts on uncertainty had a very slow development, from Aristotle who described a possibilistic modality, through Occam who was thinking about natural propensities, to the great philosophers, natural scientists of the Enlightenment and to Kolmogorov who gave a classical framework for probability but felt also the need of further refinements in his last papers. Statistics, probability, all direct to infinity in the number of events and based on their original models first only to a definite number of possibilities in an eternal unchanged world. As this is not the real case, abstractions are mostly weak models of the events, the role of human beliefs, subjective classifications, hypotheses should be included into these theoretically pure concepts. Even the validity of any certain implication of past experience is questioned for the future, this was also a topic from ancient Greece to the paradox of Nelson Goodman nowadays.

In the light of the criticism regarding interpretations of uncertainty and conclusions drawn from not definitely related but somehow corresponding events, we can understand the unsolved quarrel among tenants of the orthodox disciplines in statistics and probability, the neologs as the Shafer-Dempster theory supporters or the heretics as advocates of the fuzzy concept. It is worth to mention that the first notion of the modern idea of minimax consequences developed by Wald and Neumann with some empathical interpretation can be traced back to Theophrastos, a pupil of Aristotle: "Conclusion follows the weaker part."

Our problem is similar but much more acute: we put unreliable data about vague events in our computers, translate human verbal estimations, beliefs into uncertain qualitative gradings and try to apply consequence algorithms for the unification of these vague events on the basis of these uncertain data to predict future consequences, we face a hardly or totally not perspicuous complex because of all these and finally draw brute force binary decisions over this mess.

CONCEPTUALIZATION, THE FIRST PITFALL IN LOGIC

The first step of any logical procedure is (and has been in history of science) conceptualization. The Aristotelian categories are direct ancestors of our types, classes, objects in different programming languages, the well-defined relational dependences, the topos of type declarations, frame and semantic-conceptual nets. The notion of inheritance within a conceptual frame was also familiar, semantics as hermeneutics, the science of meaning, interpretation. In this respect, not too much was added, only refined. Denotation started also with Greek science, a more coherent methodology had to wait until Leibniz and Frege. Investigating relations among concepts and items, the syllogisms provided an analog tool as our IF...THEN construct, used in every rule-based system. This was the modus ponens or, by the ingenious mnemonic of Middle Ages, the Barbara-algorithm. The other important syllogism, the modus tollens (Celarent) leads to our refutation algorithms, to the resolution principle and nonmonotonic logic counterfactuals. As Zeno of Elea formulated: reduction to impossible. The metaconcept, rules about rules, hyperstructures are also results of this fantastic period of human awakening of consciousness. It was realized very early and included into the works of Aristotle that those concepts and syllogistic relations are context dependent. They analyzed, separated the truth of premises and relations, distinguished the essential features of a certain instantiation (as we say: they refer to it as first substantial) and that of the conceptual nature of the same (second substantial) - some later ideas of limited, conditional inheritance. Creation of hierarchical conceptual structures was a widespread logical game realizing the concept of accidental category, the relativity of all that. Similarly we find these different aspects of conceptualization in Thesaurus-organized dictionaries (Webster, Roget, Longmans, Pictorial-Dudem).

The usually cited examples of pitfalls in conceptual thinking, generalization and instantiation are the oistrich and the penguin, i.e. birds which do not fly, the penguin has no real wings, the whale which looks and swims alike a fish but is a mammal. These are simple tutorial cases but emergency directives of a complicated, dangerous plant can be extremely controversial in different situations when some part of the system is out of service or the antecedents are different or any other reason changes the order of risks. Similar can be a medical procedure, considerations on contraindications, risks due to different situations, investigation strategies etc.

MODALITIES, THE SECOND PUZZLE

Most important was the discovery of modalities, i.e. different reasoning (syllogism) under different contextual relations. Aristotle distinguished three: the necessary (apodictic) which is our regular rule form in simple expert systems; the possibilistic; and the contingent one which permits coincidences, somehow directing to our patterns. This contained the fine distinction between something which is possible or probable. In our systems these are transferred as probabilistic, possibilistic logic, beliefs, certainties, contingencies. A significant part of development during the whole history until now has been devoted to the extensions of the modality concepts, their applications to special fields of reasoning (e.g. legal, where relations are transformed to compulsory, permitted etc.) and handling those differences which arise from problems of conceptualization mentioned above. This will be explained further in more detail, here we turn back for a moment to our task: computer representation.

Anybody who is familiar with data structures and operations on them can be easily imagine the implementations. We gave hints at every historical concept to the renamings, computer-science vernaculars, buzz-words. This is all very simple for any short illustration but opens bifurcations, ramifications at each open-ended uncertainty either in data or in conclusion-paths. The result is just as told before: either a short-cut which can lead to totally misleading conclusions or a wide choice of outcomes with a puzzling confusion of their individual values or both.

DIFFERENT WORLDS, THE THIRD CONFUSION

An early understanding led to the conclusion that different relations, views can provide either a completely different but in itself consistent picture of a situation or alternative choices, values can create different situations which evolving in parallel can meet and conflict. Leibniz, gathering from the accidental looking human life, argued that God could not play with dice (an idea reformulated by Einstein) but He created an infinite number of worlds where every possible event and their combinations were realized. We are living only in one single variant. The original vision of Leibniz is an impressive hint to the computational absurdity of any totality. The idea of the necessary rationality of the world"s creation can be also followed from the Genesis, through ancient Greeks who supposed a geometrically thinking God (rationalism was that times mostly embodied in geometry) to our latest cosmogonial theories. Our practical point is the problem of limitation of this unlimited. Modelling a certain task-area we have to define these limits and the consequences of the limitation much more definite and reliable than any time before. We must be aware of the fact that the description of limitation is unlimited as well, if we would do it in a perfect way.

The idea of different worlds, resp. different world descriptions, views returned with Wittgenstein and Kripke not long ago as they described the different world structures with the much cited story of London-Londres (a French boy who has a splendid Londres in mind due to his readings and by chance turns up in an ugly suburb of London and continues dreaming about that splendid city.) The old paradigm of Bosporus-Hesperus symbolises the same. Essentially the same has different names for different agents and by that these worlds are really different, a source of contradictions. We all live somehow similar, if somebody listens to a case told by differently oriented people. Distributed systems are in case of some mismatch Kripke-structures, we shall return to this problem, speaking about the difficulties of dialogs, the effects of low quality information. We can follow the responses of computer and information science and technology in development of interfaces, protocols and the difficulties of these even in relatively simple cases, the alternative views on them. The possible trade-offs, compromises indicate the immense complexity behind all more or less realistic applications.

A relevant additional uncertainty is given by the Halpern-Moses theorem. They have proved that no perfect information can be exchanged between two partners, if the transfer of information requires considerable time. The paradigm is the coordinated attack of two detached generals. This looks of course trivial if during the time required some unpredictable changes can occur, for us it opens additional complexity troubles.

PARADOXES - CONTRADICTIONS

Eubulides, disciple of Euclid defined four basic logical paradoxes which have relevant roles in our systems as well: 1) the Liar (if he claims that his breed is lying, then the truth is uncertain); 2) the Hooded (if he claims that he knows something and it is apparently hidden, the truth is uncertain in this case too); 3) the Bald, the Heap (having one single hair or taking one grain of a heap he/it is still the same, this is true with two, three etc. steps - where is the limit of the true statement?); the Horned (if somebody did not loose his horn, should be horned?). The Liar-paradox is the most significant one. It is the same what we call self-reference and it has a benevolent and a malevolent version. The first is the case when a system provides a reference ("it is to my left") which without further external reference remains ambiguous. Malevolent is a system which claims to tell something about its truth and this remains ambiguous. The case is realistic in interconnected systems where each individual system should rely on the correct operation of the others, the data issued by them. Considering the difficulties of program and hardware verification we can assess this pitfall. The voting problems of the Space-Shuttle computers, the future of SDI software, reliable services of networks are typical cases for that. The self-reference problem is one of the deepest of logic illustrated by several ingenious paradoxes and has different aspects, some closed systems permit to use it without harmful consequence, others lead to apparent contradictions. From our point of view this is an important practical warning and avoidance leads to an increased complexity.

The Hooded paradox returns in the self-explication features of expert systems, the Bald (Heap) in the definition of possibilistic-fuzzy limits and reasoning procedures. The Horned draws attention to sensitivity due to presuppositions, a typical problem of expert system design in complex cases.

Contemporary conclusions are very disappointing: Hintikka speaks about the need of a logical omniscience, Konolige about the requirement of a consequentional closure. Both define a limitation to an unrealistic situation which, pursuing the sequence of paradoxes, is unfortunately the only realistic approach. We meet the unavoidable compromises in any computer-based system.

LOW QUALITY INFORMATION - HIGH QUALITY ELECTRONIC ENCYCLOPAEDIA

An obvious corollary of all uncertainties, limitations detailed above is the unavoidable fact of incomplete information in any complex case. The situation is mostly worse: the human agent as information supplier is usually a low quality resource. If the system impresses its logic on the human partner, avoids any new approaches, then the system and its activity is not really intelligent. In a free discourse, the system not only interprets the information given by the agent but it has to restore a possibly complete and consistent information on the world concerned. Natural language understanding is only one, but still not sufficiently solved chapter of the task. One approach creates (and backs up) two distinct models: that of the agent and the other of the subject. The combination of the two can provide a more reliable picture of the case.

This is an extremely important practical issue. A dialog between a doc and the patient, any communication between an expert and a layman is an instance of the general problem and this is one of the basic paradigms of future man-machine systems world.

The addition of nonverbal information is another side of the same problem. We have no general method for one to one transformations of verbal or pictorial morphological descriptions as for plants, objects of cytology, faces, figures etc. The representation of other sensory data was also an original Aristotelian problem and later of many great philosophers, we hint only to Descartes' Trait de I"homme.

We have been speaking for a time about the third component of complexity - uncertainty: about language. Relations of logic and language are primeval, here we put emphasis on logic; language more detailed is another occasion.

On the other hand, a revolutionary high quality information source is evolving: the hypertexts over an electronic encyclopaedia. High-density storage equipment makes available a complete library of a very broad knowledge base for immediate information retrieval. Handling this, creating associations of remote items is a much more demanding process than any of our recent intelligent, relational data base managements. If this is done to a certain extent in real time operation, the low quality human information and high quality knowledge base can create a new, very efficient symbiosis. As we see later this is only a step further in approaching something of the human intuition but it is a helpful support.

NEW KNOWLEDGE - DISCOVERY

We reached a transition to an old question of possible machine intelligence: can they imitate the human intellect or even do some more? Much before the well-known idea of Turing about a man-machine test procedure, Descartes described a similar one contemplating about the same. From our point of view it is not so important to speculate about computers having emotions or views, feelings about themselves, the self-reference problem was treated here also in a more practical way. Basic for future and contemporary assessment of expert systems would be the ability to create new knowledge. First it is to be defined what is new knowledge. A calculation of a number of special conditions (e.g. a key in cryptography) is also something new but this cannot be included into AI, if it is not a result of a new, computer-generated algorithm. This notion is nearer to what we understand as discovery although several authors misused this term as well, claiming too much for their more or less heuristic search programs within a well-organized discipline.

A rough classification can help us to get a clearer insight. There are discoveries which preserve an earlier scheme of the problem. Finding some inconsistences, new facts which do not fit into the scheme, trace by backtrack to a node or a subgraph composed of a few nodes of the scheme, which can be a culprit for contradiction and then replace this by new data, rules, algorithmic parts. This is the way how nonmonotonic logic works in our expert systems. Any extension of a theory (e.g. the van der Waals equation from the Boyle-Marlotte gas-dynamics law, or slightly the specific theory of relativity) preserves most of the other parts of the scheme. The other even more exciting procedure is the change of the general scheme of a theory, something what was coined by Kuhn as new paradigm. The general theory of relativity, the mechanism of genetic material, the theory of mathematics are typical examples. This is mostly based on some sililarity intuition, usually postulating very far fetched, unexpected analogies, as a helix of DNA and of a staircase, i.e. similarities of structures, external forms, data, situations, synonyms and antonyms. As far as our machine systems have developed until now, this would be a practically eternal search for a machine on an unlimited field of combinations but as we could indicate with the intelligent electronic encyclopeadia it can be a very promising man-machine cooperation. The same conclusion is reached in this way as before: any limited machine procedure reaches the limits of complexity and does not really replace or imitate the human mind but it can amplify our capabilities by a wise partnership concerning limitations on both sides.

LESSONS OF COGNITIVE PSYCHOLOGY

Logic endeavoured from the very start to formulate human cognition. In its struggle for a superior reason - weapon against the irrational evils of human life, it long neglected any other methods of cognition or have let them to be the territory of the other, nonprogressive part or at least excluded from science, separating it from the arts. It was realized by psychology that, due to our aeons long phylogenetic evolution patterns, Gestalts, schemes, i.e. somehow coherent data in our memory have a much more important role in our mental actions as logic. Even those creative activities which are considered to be the most logical ones are not based on logic but just on those patterns as it was proved in chessplaying or discoveries in math. This is just a separating indicator of a medium level professional from the performance of a real talent, the first uses about 2000 of those schemes in his/her daily professional activity, the latter 10-100 000! How are they organized, what kind of hyperstructure do they have, how are the main features extracted and stored, filtering the less important? - this is rather mysterious till now but they are in the focus of recent research.
We are sure that finding some features of this fantastic heuristic would be very rewarding for our machine-based systems as well. We have two projects in different fields, in a special

early brain development medical topic and a legal decision support, based on precedents. The intuitive reasoning methods of the experts on the field are our challenging objectives. We try to find approximate metrics of those very complex spaces of events, measures of similarities, conceptual clustering.

HOW FAR?

We started with reference on overstatements and underestimation. Let us conclude with impressions of the recent situation with no predictive extrapolations - the future is always a surprise.

In spite of more than two decades of ingenious and hard, extensive work, AI and expert systems did not reach a level as it was foreseen in the beginning and is claimed by some aggressive advertisements till now. They are either in a state of demonstration and experimentation or used as tutorials, guidance for non-experts, putting limited systems in more perspicuous ways. They are more helpful in problems which are basically combinatorial and relatively easy to estimate. Here, that range of possibilities is used which lies between the limits of a human sight and the combinatorial explosion, e.g. the cases of computer configuration or molecular structures. This range and the whole progress is very valuable, cannot be neglected in any future development, but we have seen limits which indicate frontiers set by complexity where the number of calculations is practically unlimited, these are the NP complete problems but the polynomial ones also, if it goes to high numbers. Amdahl's Law reminds us to the fact that high parallelism or accelerations by one or two orders of magnitude cannot be helpful in those very complex cases. Our endeavour was just to hint to the deeper reasons of that.

This should not be demobilizing. Efforts are going on just in those two directions what we mentioned at the discovery topic. A steady-state effort broadens the potentials of algorithmic procedures defeating complexity, turning NP problems to P and P to NlogN, approximations with better estimations of errors are invented. New computational and reasoning paradigms are launched which attempt to bypass our present limitations. Being aware of these limitations is an important step itself and a move towards more efficient man-machine systems.

*- •

V.

V , \

W'^r
V

i / x

./*fi

/

;

. \ ' '

'i i^ife^

: á iS : V íílillS A -K ^V ., J Í - ^ , _• -Ju' ^'. í'''.'"^» * '»

. 'ijí-i'S il

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 189-196.

A PARADIGM OF LOGIC PROGRAMMING AND ITS IMPACT ON LOGIC PROGRAMMING LANGUAGES

Miklós Szőts

SZAMALK Applied Logic Laboratory
H-1502 Budapest 112 P.o.b. 146

Abstract. This paper presents a general paradigm of logic programming. It is shown that it covers all the used and proposed logic programming languages, moreover in several respect it can control the developing of further languages.

Keywords. Logic programming, semantics, inductive, definition, search space.

1. INTRODUCTION

Theorem proving systems, including logic programming languages, at an abstract level can be described as ***proof procedures*** that is triples of

- the set of formulae to be processed (syntax),
- the rules of inference, which can be applied (calculus),
- the strategy, which controls the applications of the inference rules.

In the case of logic programming languages syntax is the set of the logic programs.

Here we present a paradigm of logic programming characterising the proof procedures which can be considered as programming languages. This will be done in section 2. We omit deep mathematical investigations. After setting down the mathematical model of the constituents of logic programming languages, we try to point out the practical consequences of our paradigm.

Before separating logic programming from theorem proving in general, let us face another question. Since the definition of every programming language can be forced into the schema of proof procedure, we also have to separate logic programming languages from programming languages in general [Kowalski 1984]. This separation is based not on the three constituents of the proof procedure, but on a fourth factor: on the existence of the declarative semantics. If meaning is attached to programs in a precise, mathematical way, independently the executions of the programs, the programming language can be considered as a logic one.

2. THE PARADIGM

2.1 Logic programs

Formulae considered as logic programs are interpreted in one, fixed, constructive relational structure. The similarity type of this structure is said to be the ***basic similarity type.*** Being constructive precisely means that the model meets the following conditions:

(i) The carrier of the model is generated by the constants that is, each element of it can be named by some variable-free terms. There is a cannonical name among the names of each element.
(ii) All the relations (including the equality) of the basic similarity type are computable that is, the truth-value of every atomic formula is decidable.
(iii) All the functions of the basic similarity type are computable that is, one can find the canonical name of an element named by a variable-free term.

Such models are said to be H-models.

A program in an H-model is a set of definitions of certain relations. Being able to refer to these relations we need new relation stymbols $r_1,\dots,r_m$ (with arities $n_1,\dots,n_m$ respectively) which are not included in the basic similarity type. Their definitions will be of the form $r_i(x)\equiv\sigma_i(x)$. From now on symbols r_i will be called as ***defined symbols*** and formulae σ_i as ***defining formulae.*** A set of definitions will also be called as definition.

Let A be an arbitrary but fixed H-model and let us consider a definition $\Gamma=\{r_i(x)\equiv\sigma_i(x) \ : \ i=1,\dots m\}$. We say that relations $R_1,\dots,R_m$ are defined by the definition Γ or the tupple of these relations is a fixpoint of definirion Γ if model $\langle A,R_1,\dots,R_m\rangle$ satisfies definition Γ, where each predicate r_i is evaluated to relation R_i. A definition can define no relation if for an index i the formula $r_i(x)\equiv\sigma_i(x)$ is unsatisfiable. In general there may be several m-tuples of relations satisfying a definition. Regarding a definition as a proggram, we want to attach an unambiguous meaning the definition. In other words, the existence of a least fixpoint is a basic semantic demand. The class PE of the following definition meets this condition.

The class ***PE*** of ***positive existential formulae*** is the least class of formulae satisfying the following conditions:

(i) quantifier-free formulae of the basic similarity type belong to PE;
(ii) atomic formulae of the form $r_i(t_1,\dots,t_{ni})$, where $t_1, \dots t_{ni}$ are terms of the basic **similarity** type, belong to PE;
(iii) PE is closed under connectives <u>and</u>, <u>or</u> and the existential quantifier

In other words, a formula belongs to PE if it does not include any universal quantifier, and negation can occur only in quantifier-free formulae of the basic similarity type. A definition is said to be ***PE definition*** if all its defining formulae are PE formulae. The choice of PE definitions for logic programs is based on the following theorems (see [Szöts 1986]):

* PE definitions have least defined relations (least fixpoints) in every model of the basic similarity type.

* The least fixpoint can be obtained as the limits of the following series:

  ${}^{0}R_i = \emptyset$,

  ${}^{n}R_i = \{a \in {}^{n_i}A \; : \; \langle A, {}^{n-1}R_1, \ldots {}^{n-1}R_m \rangle$ satisfies $\sigma_i(a)\}$

* Let A be an H-model with infinite universe. In this case
  (i) Least fixpoints of PE definitions are recursively enumerable relations in A.
  (ii) All the recursively enumerable relations in A can be defined by PE definitions.

It is claimed that logic programs are PE definitions. Really this is true only up to equivalence. However, equivalence here does not mean semantic equivalence, but the equality of least fixpoint and of the construction of the least fixpoint. As an example: equivalence symbol in the definitions can be changed to implication, as it is done in programming with definite clauses.

2.2 Calculus

An execution of a logic program Γ (a set of PE definitions) is a proof of a PE formula δ from Γ. The expression "proof of δ from Γ" means that the validity of δ in the model $\langle A, R_1, \ldots R_m \rangle$ is proved, where A is the fixed model, and $R_1, \ldots R_m$ is the least fixpoint of Γ. The validity in one fixed model can be transformed into the semantic consequence relation. Let Ax denote the set of quantifier-free ground formulae holding in A. The following theorem holds:

$\langle a_1, \ldots a_{n_i} \rangle \in R_i$ iff

$r_i(a_1, \ldots a_{n_i})$ is a semantic consequence of Ax ∪ Γ

In the proofs the elements of Ax are not used as axioms, the machinery representing the H-model computes directly the truth-values of the quantifier-free formulae of the basic similarity type.

The main inference rule of every calculus belonging to a logic programming language is the ***unfolding rule*** that is substituting the defining formulae in the place of the defined relation symbol. It is a special form of modus ponens, only atomic formulae can be derived by it. This rule syntactically simulates the construction of the series providing the least fixpoints.

Studying the question from a proof theoretic point of view, we found a restriction of the natural deduction system (see [Prawitz 1965]), called ***PE nuturnl deduction***, which is suitable for deductions of PE formulae from PE definitions. It consists of the introduction rules for connectives <u>and</u>, <u>or</u> and the existential quantifier, and of unfolding rule as the only elimination rule. PE natural deduction has the following properties:

* It is complete for PE definitions as axioms and PE formulae as theorem.
* The PE natural deduction tree, the coresponding Gentzen sequent tree and the corresponding analitical tableau are isomorphic to each other.

PE natural deduction can be regarded as a schema for calculi of logic programing languages. However it does not mean that there cannot be significant differences between variants of PE natural deduction. The most important one lies in the treatment of variables bound by existential quantifiers. There is two distinct ways:

(a) substituting variable-free terms for the variables, or
(b) handling bound variables as meta-variables

In case (b) the PE natural deduction provides systems of constraints, - open formulae of the basic similarity type, - which have to be solved to obtain the results. The method of solution depends on the fixed model. In case (a) the leafes of the proof trees are ground formulae, - these have only to be checked in the fixed model.

We can also give a general characterisation of the search spaces of calculi:

(1) Within our paradigm the search space of a calculus degenerates to an and/or tree.
(2) This and/or tree is deterministic in the following sense: the immediate successors of a node depend on the node only.
(3) The search tree is closely related to the parsing tree of the formula to be proved. In case (b) it can be built up directly from the parsing trees of the goal formula and the defining formulae.

The above features explain, why it is significantly more effective to prove in the variants of PE natural calculus, than in the general case. Point (3) seems to be the most important, not only because it automatically implies the others, but also because it makes the proof procedure "transparent". By transparency **we** mean that one can (partially or totally) visualize the whole process of proving formula knowing only the formula itself. That is why logic programming can be considered as programming: when one writes a formula (a logic program) he can also regard the possible ways of proofs (executions) of it, not only its descriptive meaning.

2.3 Some expansions of the paradigm

Here we point out some way, how the outlined paradigm can be expanded

(1) The fixed model can be expanded by some superstructure (e.g. hereditary finite sets [Gergely, Ury 1988]). It means that not only the elements of the universe can be refered to, but also their finite sets, and finite sets of finite sets. This has not only theoretical advanteges, but helps discussing data structures and abstract data types [Szöts 1987].

(2) The paradigm offer tools for treating bounded universal quantifiers, they can be included into PE formulas. Furthermore a restricted usage of negation presents itself. New PE definitions for the negative literals can be generated from the original definitions, and this way negative literals can be deduced precisely the same way as positive ones. By this method negative literals with variables can be handled. The adequate semantics can be given in three valued logic (in partial models) introduced in [Fitting 1985]. The same fix point construction can be used as we use in **2.1**.

(3) The implication symbol can also be added to the connectives forming a defining formula. It means that the introduction rule for implication has to be added to PE natural deduction. The new calculus saves features (1), (2) of section 2.2, however feature (3) is lost in some sense. Such systems are discussed in [Gabbay, Reyle 1984] and [Miller, Nadathur, Scedrov 1987]. In this case the familiar fixpoint construction also provides meaning to the logic programs, however they have to be interpreted in Kripke models

3. LOGIC PROGRAMMING LANGUAGES

3.1 Existing languages as instances

First we show that the widely acepted PROLOG language fits into the presented paradigm.

In the pure PROLOG the basic similarity type consists of the function symbols and the fixed model is the Herbrand universe with the functions interpreted in the natural way. In practical PROLOG versions the arithmetical built-in predicates also act as relations of the basic similarity type. Note that our paradigm shows that these built-in predicates are not "impure" features of PROLOG. The predicates occuring in the heads of clauses are the defined symbols.

The PROLOG programs are but PE definitions - in normal form. The normal form is forced by the resolution principle. Similarly, if normal form of the programs is insisted on, the PE natural deduction turns to SLD resolution. This is a good example of the interaction between the precise formulation of the syntax and the calculus.

The foundation of the programming with definite clauses [Apt, Emden 1982] is a special case of our general fixpoint theory. However their treatment is of syntactical nature, and so it is connected to SLD resolution.

The SLD resolution is a simplified version of the PE natural deduction. Because of the normal form, the introduction rules are not explicite. The origine of SLD, the SL resolution uses lifting by meta-variables, and so does SLD. Because of the Herbrand universe, handling of constraints turns to be simple: they can be solved by unification independently of one another. The unfolding rule combined with unification provides the resolution rule for definite clauses.

SLD resolution is the only variant of resolution which could be the calculus of a logic programming language. The reason is that this is the only one which satisfies the conditions for the search space stated in 2.2. The only resolution proof procedures satisfying (1) are the linear resolutions, but even the more refined variant of them, the SL resolution is not effective enough, - as practice showed it. Even point (2) is weakly satisfied by SL resolution, since the ancestor resolution step is deterministic. However, in the input resolution step every literal of an input clause can be resolved upon. This uncertainty makes point (3) fall totally.

Let us see the different extensions of PROLOG. Since all of them keep the clause form, they similarly keep also the basic structure of SLD resolution.

[Chandra, Harel 1985] presents a theory of Horn clause queries. There the basic similarity type consists of relational symbols only. The fixed H-models are represented by relational data bases.

In the language EQULOG (see [Goguen, Messeguer 1984]) the basic similarity type also consists of function symbols only. However, for certain data type the carrier of the fixed model is not the Herbrand universe, different terms may denote the same elements. Therefore constraints cannot be solved by unification. It is supposed, that the most general solutions of equations can be expressed by terms. Again this presupposition helps to solve constraints independently of one another.

The paradigm of constraint logic programming (see [Lassez 1987]) realizes the importance of handling constraints togather. There unfolding rule is separated from the solution of some constraints. PROLOG III (see [Colmerauer 1987]) can be cosidered as an implementation of constraint logic programming.

We in the Applied Logic Laboratory started to realize a language called LOBO (LOgic of BOunded quantifiers), which gives up normal form of the definitions. It realizes the introduction rule for existential quantifier without using meta-variables (case (a) in 2.3). The substitutions of terms is guided by the bounding formulae of the quantifier. Bounded universal quantifier can be also used. About LOBO see [Szőts 1984], [Gergely, Szőts 1985].

3.2 The moral of the paradigm

We think that the general paradigm, we present here, helps not only to classify and analyse existing or proposed logic programming languages, but it can also control the development of new ones. Having a clear theoretical basis can help to avoid ad hoc solutions and to find the adequate one. Here we want to show two important features which can be derived from our paradigm.

The first is the principle of "programming in one model". The computing mechanism of a logic programming language can be separated into two constituents:

a version of the PE natural deduction, and
computations in the fixed model.

These can be considered as a higher and a lower level respectively.

The version of the PE natural deduction is a universal calculus, while computing in the fixed model can be implemented independently of any logic system. If meta-variables are not used, it only computes values of functions, and tests relations, however in the case of meta-variables the method for the solution of systems of constraints belongs here. Since the first constituent is universal, the same implementation can be coupled to different computational methods corresponding to different fixed models. This way ***open logic programming languages*** can be developed. This construction seems to be practical in the case of domain dependent languages. Then the functions and relations of the fixed model can be computed by complicated programs, and the logic component organises and coordinates the execution of these programs. The same logic component can serve several languages.

The second feature is connected to the form of the programs. Our arguments in section 2. clearly show that the clause form used in PROLOG and in its relatives is totally incidental. The normal form has its advantages: the search space is regular, but also has its drawbacks: some subformulae have to be repeated. If one defining formula is used rather than several implications, a greater formula can be handled by the calculus than a conjunction of atoms.

Because of the lack of space we could not expound the extensions of the paradigm, therefore their impacts cannot be discussed. However we want to note that the super structure (see point (1) in 2.3) can help not only the adequate treatment of data structures but also of several other features of logic programming, like the "set of" predicate, perceptual processes and so on.

4. FINAL REMARK

Here we have reported some results of a project which aims to provide an adequate theoretical characterisation of logic programming languages. The project is far from being finished. Beside some theoretical questions (like using higher orde logic) search strategy has not been even touched. The treatment of this component is one of the greatest problem in logic programming. Please accept our paper with this apparent deficiency.

References

Apt, K.R., Emden, M.H.van [1982]
"Contributions to the theory of logic programming"
Journal of ACM VOL 29 No 3 pp841-862

Chandra, A.K., Harel, D. [1985]
"Horn clause queries and generalisations"
The Journal of Logic Programming Vol 4 No 1 pp1-15

Colmerauer, A. [19087]
"Opening the PROLOG III universe"
Byte August 1987 pp177-182

Fitting, M. [1985]
"A Kripke-Kleene semantics for logic programming"
The Journal of Logic Programming Vol 4 No 2 pp295-312

Gabbay, D.M., Reyle, U. [1984]
"N-PROLOG: An Extension og PROLOG with Hypothetical Implications I"
The Journal of Logic Programming Vol 1 No 3 pp319-355

Gergely, T., Szöts, M. [1985]
"Some features of a new logic programming language"
Proceedings of the Workshop and Conference on Applied AI and Knowledge-based Expert Systems ed. P. Revay, Stockholm, Norstedts Tryckeri AB

Gergely, T., Ury, L. [1988]
"Constructive base for programming and specification theory"
technical report of ALL, Budapest

Goguen, J., Meseguer, J. [1984]
"Equality, types, modules, and (why not?) generics for logic programming"
The Journal of Logic Programming Vol 1 No 2 pp179-210

Kurucz, A., Szöts, M. [1987]
"A proof theoretical foundation of logic programming"
technical report of ALL, Budapest

Lassez, C. 1987
"Constraint logic programming" Byte August 1987 pp171-176

Miller, D., Nadathur, G., Scedrov, A. [1987]
"Hereditary Harrop formulas and uniform proof systems"
Symposium on Logic in Computer Science pp98-105
IEEE Computer Society Press

Prawitz, D. [1965]
Natural Deduction Almquist & Wiksell

Szöts, M [1984]
"Comparison of two logic programming languages"
Second International Logic Programming Conference
ed St. Tarlund, University of Uppsala

Szöts, M. [1986]
"A general first order theory of logic programming"
thesis, technical report of ALL, Budapest
Szöts, M. [1987]
"Abstract data types in logic programming"
technical report of ALL, Budapest

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 197-206.

COMPUTER-AIDED GENERATION, PRESENTATION AND INTERPRETATION IN THE HISTORY OF SCIENCE AND TECHNOLOGY

Zs. Ruttkay, A. Márkus
Computer and Automation Institute
Hungarian Academy of Sciences, H-1502 Budapest, P.O.B. 63

Abstract: Although previous efforts have proved that computers can be used to store and retrieve the factual data about the history of a technical product or process, little attention has been paid to the problems of handling free-form data and to the means for filtering out the relevant information, for arranging it into a sequential report, for browsing it in order to answer questions or gain hints for further data collection. Both the intellectual and manual aspects of compiling the factual data into reports were entirely left to the human expert. The paper outlines the functional requirements a computer system should fulfill in order to meet the above-mentioned needs of historians. This well-defined application domain is investigated as a foil against hypertext systems.

Keywords. Hypertext systems, Knowledge Acquisition, Knowledge Representation.

1. INTRODUCTION

Previous efforts of a Japanese-Hungarian team of historians and engineers in computer-aided history of technology have proved that computers can be used to store and retrieve the manifold and comprehensive factual data about the history of a technical product or process. On the other hand, little attention has been paid to the problems of handling free-form data and to the means for filtering out the relevant data from the huge amount of information, for arranging it into a sequential report, or for browsing it in order to answer questions or gain hints for further data collection [7,8,13]. Both the intellectual and manual aspects of compiling the factual data into reports were entirely left to the human expert.

Our aim is to work out a computer system to support the actual needs of historians [4]. We have considered this well-defined application domain as a foil against first-generation hypertext systems [1,11]: we point out those functionalities which cannot be provided by current hypertext systems.

In Chapter 2 a short account on the analysis of historians' methodologies is given, in Chapter 3 the functional requirements for an appropriate computer tool are listed and contrasted to the capabilities of present hypertext systems.

2. METHODOLOGIES OF THE HISTORIANS

Relying upon a wide scope of authentic sources and having interviewed historians of good reputation, first we have examined the specific features of historians" methodologies and attitudes: how do they collect and interpret the information, how and for what purpose do they make inferences on it. Nothing proves the difficulty of these questions better than the fact that there is no commonly accepted handbook on how to write history.

The interpretation of the available historical information is the real challenge for a historian, it requires intelligence, creativity, some guiding prejudice but no pride; systhematic tests as well as luck; simple common sense and sophisticated experience; wide background knowledge on the socio-political circumstances and a reasonably deep understanding of the specific scientific field, etc. The historian has to be sensitive to the sources, he has to be open-minded and flexible. He should not by-pass facts if they cannot be explained, or if they do not fit into his scheme. This also implies that history should form the historian; by investigating more and more details, his methods and preconceptions might be continuosly changing.

On the other hand, historiography should meet overall criteria of scientific cognition, e.g.: a theory should not contradict true factual data, it should keep an eye on completeness and consistency. Research should be carried out by means of contrasting hypotheses, operating with clearly defined concepts, relying upon logic in reasoning [9, 12].

2.1. Setting the aims

Usually, the historian has his aims set before he starts intensive research. First of all, he narrows down the topic he is interested in. He might define his interest not only by referring to specific periods or areas, but to concepts, relations or phenomena (e.g. the impact of scientific theories on technology, or the role of ingenious inventors in the history of science and technology).

An important aspect of his aims is the initiative for the research: whether he has a hypothesis to be checked, or intends to answer a specific question or to investigate a topic. He has to define the depth of investigation, based either on some preconceptions of the researcher, or on the resources available, or on the expectations concerning the research (to produce a basic monograph that should be referred to for any detail, or to give an interpretation of some period/phenomema).

The setting of the aims of the research depends on the intended consumers of the results, too: whether they are familiar with the topic, whether they have special preconceptions or interests (e.g. resulting from nationality, education), whether they want to be amused or informed. Their background knowledge and mental capabilities affect the scope of context and the methodologies of interpretation.

2.2. Relevance of data

The historian should have an initial idea about what data should be looked for, and what might be the sources of the information. But he should not insist on his a priori decision on the relevance of the data: while looking through the preselected sources, he might have the impression that he had an unrealistic expectation concerning the content of the sources (e.g., they do not refer to the type of information he needs). Sometimes, the available forms of data do not match the scheme made up by the researcher. Some information, initially beyond the scope of data collection may change the scope of search entirely: further on, the histroian will be interested mainly in this additional data; he may reiterate through those parts of the sources which were skipped previously.

2.3. Definition of concepts

The definition of concepts is not a straightforward procedure either: the task is to distillate concepts from the comprehensive data rather than to arrange data according to a given conceptual taxonomy. A related problem is to trace the semantic changes of the same concepts used by different authors at different times: the sources should be interpreted carefully: e.g. in cases of confusion caused by the coincidence or misuse of names, objects should be identified by their attributes.

A clear and fixed reference world should be given: all the terms occurring in sources should be expressed in terms of this reference world. This also makes it clear that the meaning of historical texts cannot be clarified by verbatim analysis.

2.4. Partial information, intencionál description

Historians have to cope with ill-defined entities as well. Usually, only some partial information is available, which is not sufficient to identify the entity. In most cases, the entities can not be defined extensionally, i.e. by listing their relevant features, but intensionally, by relating them to other entities.

Historians are keen on differentiating between not knowing the value of some data and not knowing about the existence of some data. They take into account the knowledge about the completeness of the processed sources and about the unevennesses of the collected data.

2.5. Contradictions

In the stage of data collection, historians often have to face contradictions: multiple values for the same data, facts contradicting to common sense knowledge or violating the consistency of the body of data. Human historians do not destroy contradictions with fire and sword, rather, they keep them as hypatheses until further, clinching arguments have been advanced. Such competing hypotheses could guide further investigations in order that, finally, one of them be accepted to be true.

The extreme or unreasonable data arouse the interest of a historian: they might urge him to collect additional information to give a reasonable explanation of the deviation. Even an evidently false data should not be rejected without considerations: it might reflect e.g. the misbelief of the author of the source, which might be of crucial importance in explaining his activities.

2.6. Principles for presenting data

A simple organizing principle is to choose specific attributes of the data and arrange the pieces according to the value of the chosen attributes (e.g. in a chronological report). In this case, details on the listed objects/events are encountered in a more or less strict lexicographical order according to the values of the organizing attributes (e.g. chronology -- country -- inventors or inventors -- chronology).

Moreover, one can arrange the data into chunks around kernels. These kernels can be specified in terms of the type of events/objects (e.g. inventions), or special aspects of an event (e.g. scientific background, technical details of an invention, further impacts of the invention).

Another approach is when the stress is not on the details of the distinct objects/events, but their impact on each other. History can be presented along specific relations, resulting in a story with more inherent logic than ordering of the facts by the values of certain attributes. In this style, restrictions and preferences can be given on the class of statements (e.g. causalities), on the qualification of statements (e.g. pushing and pulling factors), on the semantics of statements in terms of the type of statement and type or value of some of its attributes (e.g. the role of manufacturing demands).

2.7. Filtering criteria for the data

From time to time historians restrict the scope of their interest; in order to deal with special aspects filtering criteria are to be formulated in terms of attribute types and/or types of statements.

The depth of the investigation can be constrained by discarding either some types of data (details of inventions) or statements (further impacts, parallel attempts). Often, instead of listing specific objects/statements, an aggregation of these statements is given (e.g. referring to sets of objects, overall statements). The depth of reasoning (exploring causalities) can be limited, and also secondary information (on sources, on qualification of statements) can be disregarded.

When presenting the information one may restrict himself to one alternative of the values/explanations/hypotheses; in other cases, putting emphasis on the competing alternatives may be a main point of interest.

2.8. Investigating the structure of the body of data

Similarities in the structure of statements can be investigated, either to enlighten one of them by the other, or to filter out typical historical patterns. The same statements with different attribute values can be related to point out similarities; groups of statements with different attribute types may suggest surprising associations for analogies, while ones with attributes of the same value or the same type but with different qualification (e.g. failure -- success) can be contrasted.

The structure of the body of data can be the topic of investigation in itself, too. In order to point out separate chunks of data, one can investigate whether two satements or objects can be related (via statements sharing some attributes), where the hot points in the network of collected information are (e.g. too many unknown attributes for statements, very few statements referring to an object).

2.9. Reasoning over the facts

Reasoning plays an essential role both in data-collection and interpretation: historians do it consciously, using a comprehensive set of - not necessarily deterministic - rules. On the one hand, a body of data conveys much more information to the historian then simply a set of facts. He, relying upon common-sense and domain-specific knowledge (rules, procedures), endows the facts: whenever needed, through deriving conclusions he can add new facts to the initial ones.

When investigating and interpreting the data, the historian - using all his expertise as a historian - makes guesses for further statements to be tested, or for possible explanations to be checked.

2.10. Data collection, presentation and interpretation interwoven

It seems to be an important characteristic of historians' work that data-driven and goal-driven styles are intermixed in their work. Not much advice can be given how to separate and guide the stages of data collection, representation and interpretation. When collecting data, it is necessary to investigate the known data in order to judge about the value of some new information: a new piece of data may initiate the need of reconsideration of some data accepted previously, the investigation of the body of data may result in hints for reinterpretations; on the other hand, a hypothetical interpretation can be used to guide further data collections.

One should not forget that both the historians' methodology and their domain-specific knowledge do change in course of their activity, historians do learn from their experiences.

3. WHAT IS REQUIRED OF A COMPUTER AID FOR HISTORIANS

In past decades, several software tools have been designed "to support the task of transforming a chaotic collection of unrelated thoughts into an integrated, orderly interpretation of ideas and their interconnections" [3]. These so-called hypertext systems have gaining interest both in the users' and software engineers' communities. While they have been proved to be handy and adequate tools in several, usually rather small-scaled applications, they turned to be too restrictive and difficult to use when modelling complex, large-scale domains [2].

Central aims of hypertext systems fit to the needs of historians quite well, so we can declare, in general, that what is needed is a hypertext system. Surprisingly enough, following the above analysis of historians' work, we have found that none of the existing hypertext systems would be entirely satisfactory for their purposes. Even a well-isolated and not too large application domain [5] has raised issues which have significance beyond the existing hypertext systems.

The reader can get a view of hypertext systems e.g. from [1,2]. We will use the generally accepted hypertext notions (network, node, link, type, instance, authoring versus browsing, query e.t.c.) in their usual meaning.

3.1. Representation of objects and statements

The usual node-link scheme cannot be used with the cast featuring objects plus binary relations between them: it is necessary to deal with relations having more than two attributes.

The suggested uniform representation of objects and relations is in form of nodes with predefined slots, and links in form of pointers from a certain slot of a node to another node as a whole. This representation is appropriate to deal with relations wich may have both objects and relations among their attributes. The unprocessed, purely textual information within a node can be stored in character-type slots of the node.

3.2. Instantiated versus referred nodes

One should distinguish between the events of (1) creating a new node and (2) creating a link to an already existing node. In the case of an object identified by the value of one of its slots (e.g. a name) it is easy to find out whether the node has been already created, but the incidental dismatching of the values of a certain slot (e.g. the date of birth of somebody) should be registered. A systhematic check of all the candidates should be performed in case of nodes wich can be distinguished only by identifying the values of their slots. The system should be prepared to instantiate nodes with missing or temporarly given slot values.

3.3. Permissiveness in assigning values to the slots

In order to suit to the incremental style of data collection and concept distillation, both the defaults and the constraints should be permissive: extreme values, violated type or domain restrictions should be allowed. In addition, multiple candidates for a value of a slot should be allowed.

Besides single-valued slots, slots with more than one values could be defined. Also distinction should be made if a slot is used to identify a node or not.

3.A. Invertable pointers

Since in advance no preferred pathes should be specified within the network, the direction of the access of nodes should be left undefined. On a more technical level: pointers should be bi-directional.

As a result - in contrast to the common practice when a single structure of the physical arrangement of the nodes is fixed once and for ever (implemented as e.g. files or folders) - , the system should offer several methods of the same rank for the access and re-arrangement of the nodes, and should store the nodes in that way, if required.

3.5. Modes of usage: authoring, browsing, presentation

The system should provide dedicated tools for the purposes of authoring, that is of a piecevise mapping from a large amount of external textual information onto a hypertext and continuosly modifying this hypertext whenever necessary.

In addition to the support of the browsing mode driven by the queries from the online users, features should be given for presenting the hypertext in form of traditional reports fulfilling some overall requirements.

As historians essentially do not work with a frozen set of data, and the activities of data collection and interpretation are highly interrelated, it is necessary to allow the different modes of usage to be embedded into each other.

3.6. Tools for browsing the network

In order to avoid the frequently discussed situation of "getting lost in hyperspace", there should be offered more sophisticated browsing techniques than freely wandering over the network.

Virtual links should be generated to ease the navigation over an extensive network or to indicate unexplored parts that might be interesting for further investigation.

In the case of dense and large networks, the graphical visualisation of the network is not sufficient: the content of nodes should be represented in textual form, unfilled parts, alternatives, links should be indicated consequently.

3.7. Sophisticated network-editing

A much discussed shortcoming of present hypertext systems is the problem of "premature organisation", i.e. they do not support sufficiently the modification of the network in course of further progress.

New editing functions should be added to the traditional node/link creation/deletion primitives, moreover the technical issues should be solved in a consequent and transparent way (e.g. problems of alternatives, links to non-existing nodes).

The modification of the content of nodes should be supported. By providing utilities, both the elevation of a textual slot value into a node and the splitting/joining of nodes should be allowed.

3.8. Global filtering

The more comprehensive the network is, the more inevitable is to offer tools for deliberately restricting the scope of investigation. Global filtering criteria should be expressed, first of all, in terms of types and values of nodes and links, or by restricting the qualification of statements to be taken into account; finally, in terms of the structure of the subnetwork at hand.

Filtering criteria can be used in an exclusive way or as preferences during the sessions with the system. By changing the filtering criteria, the system should be tuned to various purposes and specific users, or to different working methods; even preconceptions can be demonstrated in this way.

3.9. Network-query

Query-based access should complement navigational browsing, allowing search patterns referring both to the content of nodes and to the structure of links. A pattern language should be provided to formulate expressions with the usual operations (negation, alternatives, closure).

3.10. Optional enhancement with rule-based features

In contrast to relying upon only the network as it is, the system itself should possess knowledge about how to explore and enhance the network.

On the one hand, the system should be able to generate new nodes/links on its own initiative, and not only by expecting the user to do so explicitely. So the system should be active, by performing utility-like operations.

On the other hand, the system should aid the user on a higher level as well: relying upon domain-specific rules, suggestions should be given for further queries/data collection. These rules could be processed by both forward and backward chaining. In the first case, the inclusion of a new node is suggested, as a probable conclusion, while in the second case a possible interpretation of the presence of a node/link is suggested.

4. PRESENT STATUS OF THE RESEARCH AND PERSPECTIVES FOR FURTHER WORK

In order to collect building material to a prototype version for testing the validity of the above approach, this research has been accompanied with a meticulous investigation of a large number of case studies concerning histories of 2oth century inventions [5]. The texts mounted up to 3ooK written by professional historians for the interested general audience.

In a nutshell, this investigation led to the following results:

Having set the borderline between formatted and unformatted data and discarded all the details specific to products, processes, firms and alike, there remained more than 3oo entities used to describe data related to the historical aspect itself [6]. These entities refer to each other but they could not be squeezed into a usual concept hierarchy; on the other hand, they could be characterised by a profile of several dimensions (restrictive versus constructive role, personal versus social scope, various aspects, intentional character of the entity etc.). Moreover, the organism of several case studies has been investigated in detail: beyond the statement level there has been found a bewildering abundance of presentation variants supporting orientation within the data and rendering the collected data [7].

In the next stage, these pieces of data should be used as actual patterns within a computer aid to be used by students of the history of science and technology: in the first version, this tool should help in writing case studies of a similar structure and scope.

Further on, based on the experience of using the prototype, the scope and flexibility of the system will be increased step by step, along the above-mentioned directions.

Acknowledgments

The authors wish to thank Hideto Nakajima and József Váncza for his critical remarks on the first draft of this paper, and Ágnes Vórkonyi for the inspirating discussions on historiography.

References

1. Conklin, J.: Hypertext: An Introduction and Survey, IEEE Computer, Sept. 1987, pp. 17-41.

2. van Dam, A.: Hypertext'87 Keynote Address, Communications of the ACM, Vol. 31 No. 7, (1988) pp. 887-895.

3. Halasz, F. G.: Reflections on NoteCards: Seven issues for the next generation of hypermedia systems, Communications of the ACM, Vol. 31 No. 7, (1988) pp. 836-852.

4. Hatvany, J.: On using a computer to write a history book, in: Progress Reports of the Joint Japanese-Hungarian Project for Expert Systems on the History of Science and Technology No. 3. Budapest, 1986.

5. Jewkes, J., Sawers, D., Stillerman, R.: The Sources of Invention. MacMillan & Co. Ltd, London, 1962.

6. Márkus A., Ruttkay Zs.: A Programmers' View on the History of Science and Technology, Part II.: A Close View on Texts and a Proposal for a Tool. To appear in the Progress Reports of the Joint Japanese-Hungarian Project for Expert Systems on the History of Science and Technology. Vol. 4. Tokyo, (1987).

7. Murakami, Y.: Traditional historiography challanged, in: Progress Reports of the Joint Japanese-Hungarian Project for Expert Systems on the History of Science and Technology No. 1. Budapest, 1986.

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 207-215.

BRAINDEX - A PC-based medical decision support system

V.H.Haase[1], H.Moik[1], G.Pfurtscheller[2], G.Schwarz[3], B.Willegger[1]

1) Institute for Information Processing
University of Technology Graz
2) Institute for Electro- and Biomedical Technology
University of Technology Graz
3) Clinic of Anaestesiology
State Hospital Graz

ABSTRACT

An interactive and a non-interactive system for supporting decision-making in brain death diagnosis was implemented on a PC. The systems accept medical data from a database. Both systems were tested in clinical practice. The interactive system yielded greater acceptance by the physicians and hence is considered to be developed further.

1. Introduction

In recent years the application of modern medical equipment made it possible to sustain biological functions like respiration, blood circulation and the nutrition cycle by machines. Therefore it happened over and over again that the physical body of a patient was maintained alive by technical facilities whereas the brain was dead already.

The technique of organ transplantation now confronts the physician with two problems; on the one hand he has to be sure that the organ donor is dead in medical and legal respects, on the other hand the taking of organs has to be made as early as possible to provide the recipient a sound organ.

For these and other reasons the physician is exposed to great mental burden and the wish to consult with a collegue who is unaffected by emotional factors is obvious. A computer which acts on expert level seems to fulfill this prerequisite best.

Another reason to develop a computer-based decision support was the great amount of medical data which were collected by clinical tests and stored in a database. These data suit well as a basis for a consultation session.

The decision support system discussed in this paper does not diagnose brain death since this must be left to the responsibility of the physician, but it draws his attention to certain points which are not in accordance with brain death. Hence the task of the system is to check all available facts to detect data inconsistent with brain death diagnosis and inform the user (the physician) about it.

The project started with developing two systems, one interactive, the other having the consultation results listed and printed out. This paper concentrates on the description of the interactive system since it seems to be received better by the physicians in practice.

2. Medical Background

The classical definition of the biological death is the irreversible loss of the function of the central nervous system following the cessation of blood circulation, which yields the death of tissue and cells.

Brain death, however, includes the complete cessation of the functions of cortical, cerebellar and brainstem structures, i.e. the irreversible loss of all brain functions with preserved function of other organs (cardiocirculatory and respiratory systems).

The irreversible cessation of brain functions, which is equivalent to the definite loss of external and internal control functions of consciousness, is considered as the death of the individual.

There are six phases of a comatose patient towards brain death, distincted by the location and extent of the lesion. The functional disorder of cerebral activity is recoverable at any stage, but in most cases patients who reached the last two stages (bulbar syndrom) died within a few hours.

The most important signs of brain death are deep coma (unresponsiveness), no spontanous respiration and loss of cephalic reflexes (e.g. pupillary light reaction).

Isoelectric EEG, evidence of the stop of cerebral perfusion (angiography) and evoked potentials are additional tests required to establish a secured diagnosis of brain death.

The time of the clinical explorations depends on the extent of interpretability of the various symptoms and the progression of the brain lesion.

There has to be some time between the first diagnosis of brain death and the final declaration of death, which can reach from 6 hours up to 3 days and more, dependent on the age of the patient and the nature of the lesion which caused the brain death.

3. Computer-supported brain death diagnosis

The decision support system is called by the user directly after consultation (or update of the patient) database. Therefore the patient and the medical test data are specified already when the consultation starts.

Since all necessary data are gathered from the database, the user need not be questioned about medical data as e.g. in MYCIN. Parameter values which are not present in the database are considered not to have been collected or not to be judgeable.

First some prerequisites are checked which are necessary for a valid diagnosis. Depending on the severeness of missing prerequisites, the choice to continue the consultation is left to the user or the system terminates.

Next the system tries to conclude a first hypothesis of the patients state from the most significant parameters. If the patient is in a phase which can lead to brain death within a few hours (bulbar syndrom), the actual consultation starts.

Every inconsistency with the brain death diagnosis is displayed on the screen, together with explanatory text and bibliographical references. Then the physician can indicate his judgement of a specific inconsistant set of symptoms by selecting one of three possibilities.

If he enters TOLERATE, that means that the displayed data are consistent with brain death. Selecting NOT TOLERATE he informs the system that the inconsistent fact actually is a contradiction to brain death diagnosis. In this case the system immediately terminates the consultation with the conclusion that the patient is not dead. Between those two extremes the user can select NEGLECT to tell the system that he judges the constellation as a contradiction, but wants to continue the consultation.

However, at the end of the session, the system displays all inconsistent facts found during the consultation. The facts are grouped in untypical, but in brain death permissible and in contradictory data. In this manner the user is reminded of all inconsistencies in a short form and can establish his diagnosis.

The non-interactive system does not have the advantage of a communication with the physician. Hence it cannot decide whether a detected inconsistency is a contradiction in all respects or there are reasons to tolerate it. On the other hand the physician need not sit down at the computer since he gets a printed output.

4. Development Issues

4.1 Software Tools

The interactive decision support system was developed with the expert system shell "Personal Consultant Plus" (PCPLUS). This shell is a successor of EMYCIN, hence producing rule-based backward-chaining systems. Further features (frames, forward-chaining, meta-rules) are implemented to increase the power of the shell.

At the development level, a running prototype can be implemented, tested and improved in the manner of 'rapid prototyping*. The user of the developed system gets a special version which is freed from the development overhead and hence runs more rapid.

Since systems developed with PCPLUS are running in a LISP-environment (PCSCHEME), special LISP-functions can be defined and used within the knowledge base. So the knowledge engineer himself can implement functions and procedures he needs for his special problem and which are not supplied by PCPLUS.

The rules and associated parameters, which deal with a special problem area within the knowledge base can be grouped together in so-called frames. This PCPLUS feature helps structuring a large knowledge base and prevents the developer from getting lost in a mass of knowledge base items.

For the definition of rules, an "Abbreviated Rule Language" (ARL) can be used. PCPLUS itself changes the ARL-expressions into their LISP representation, which can be edited as well.

Even though working with PCPLUS is very convenient, the knowledge engineer who uses the system for a while would wish to have a facility for faster knowledge base editing, especially at later stages of rapid prototyping.

The non-interactive system is implemented in INSIGHT-2. This shell is not as powerful as PCPLUS, concerning user communication and development features. Therefore the knowledge base can directly be manipulated in a text-editor. Besides, since it is compiled, it runs faster than the LISP of PCPLUS.

The database which provides the facts for the two decision support systems is implemented in dBase III +.

4.2 Hardware

The whole system (database, PCPLUS decision support, INSIGHT-2 decision support) is working on a IBM PS 2/60 under MSDOS, version 4.0. 1 Mbyte memory, 4 Mbyte Harddisk and an arithmetic coprocessor are provided. In addition to it, an EGA-Monitor and a matrix-printer (STAR ND-15) are employed.

4.3 Implementation

The knowledge base of the interactive system contains 300 rules which are grouped in 8 frames. The data base differentiates 300 items, but the decision support system only checks 70 of them so far.

The system design is influenced by ideas of 'hypothesize-and-test'. After having checked the prerequisites, a frame is called which simulates forward-chaining. The forward-chaining process is driven by the most significant parameters and yields procentual values for each coma-stage, which represent the amount of confirmative data for each stage.

If the strongest confirmation is in the phase towards brain death, the system hypothesizes 'brain death' and tries to test all known facts to prove this hypothesis via backward-chaining.

The facts are differentiated in various ways. 'Confirmative data' are expected facts, which support brain death diagnosis. Since the physician is only interested in discrepancies, these facts are not displayed.

'Untypical data' are data which are not the normal case, but can be tolerated or explained in some way. These facts are not contradictions to brain death, but should be shown to the user.

'Contradictory data' are facts not consistent with brain death, which cannot be explained by the system. If only one contradictory fact is found, the system considers the possibility of an invalid test-result, otherwise it refuses the diagnosis of brain death.

There are seperate frames implemented for groups of parameters (e.g. cephalic reflexes, spinal reactions). The frame TOP-LEVEL is the overall-frame which manages the flow of the consultation. The checking of prerequisites at the beginning and the summary at the end of the consultation are done by special frames as well.

The system does not search the data base for facts, but accepts a file of all current data, which is provided by the database before calling the decision support system.

The explanatory text which appears on the screen during the consultation is not implemented as a part of the knowledge base, but stored in a separate 'textarchive'. So it can be manipulated without changing the knowledge base, which has the advantage of not having to create a new user-version of the system.

The INSIGHT-2 system works in a similar way, although some controlling mechanisms had to be simulated since only pure backward-chaining is supplied by the shell.

5. Experience and Future Aspects

The first prototype of the whole system (database and the two decision support systems) was installed at the Institute of Anesthesiology, University of Graz, and tested with about 30 patients.

The acceptance of the interactive system was higher than the non-interactive. The testing physician suggested improvements of the explanatory text and extensions of checked parameters, but agreed with the logical methods of working.

The second release of the system will also contain new features, which do some teaching tasks: there will be a dictionary connected, so that the user can stop the consultation at any time and look up an expanded general explanation for involved medical items (e.g. how to cause a reflex and how it proceeds). Besides there should be the possibility to consult the system without the database, i.e. without having a real patient. In this case the system has to elicit the required data from the physician.

Future research will be done to extend the system from brain death diagnosis to coma-diagnosis, to implement temporal reasoning and to provide some prognostic statements about the patients outcome.

References:

[Al] P.L. Alvey et al.
An Analysis Of The Problems Of Augmenting A Small Expert System. Aus: Research and Development in Expert Systems, Ed. M.A.Bramer, pp. 61-72
British Computer Society Workshop Series 1985

[Bo] Michael Borecky
Kriterien für die Bestimmimg des Hirntodes
Diplomarbeit TU Graz, BMT, Graz 1986

[Ga] John G. Gammack et al.
Psychological Techniques For Elicting Expert Knowledge. Aus: Research and Development in Expert Systems, Ed. M.A.Bramer, pp. 105-112
British Computer Society Workshop Series 1985

[Ins] INSIGHT2 - Reference Manual
Level Five Research Inc., 1985

[Kr] J. Kriz, H.S. Sugaya
Logic Programming. Aus: Computer Systems for Process Control, Ed. Gueih
Plenum Press New York 1986

[Li] Gerhard Litscher
Multimodal evozierte Potentiate
Dissertation TU Graz, BMT, 1987

[Lo] John M. Long et al.
Use of expert systems in medical research data analysis: The POSCH AI project. Aus: National Computer Conference 1987, pp. 769-776

[Mo] Heribert Moik, Bernhard Willegger
Erstellung eines Systems zur Unterstützimg der Hirntod -Diagnose
Diplomarbeit TU Graz, 1988

[Nay] Chris Naylor
How to build an inference engine. Aus: Expert Systems - Principles and cas studies,
Ed. Richard Forsyth,
pp. 63-88
Chapman and Hall 1984

[PC] Personal Consultant Plus - Reference Guide
Texas Instruments Inc., 1987

[Pe] Gerhard Pendl
Der Hirntod
Springer Verlag 1986

[Pfu] Gert Pfurtscheller
Messung der Komatiefe. Aus: Notwendiges und nützliches Messen in Anästhesie und Intensivmedizin, pp. 51 - 61
Springer Verlag 1984

[Pu1] Frank Puppe
Diagnostik-Expertensysteme
Aus: Informatik-Spektrum 10/1987, pp. 293-308

[Pu2] Frank Puppe
Diagnostisches Problemlösen mit Expertensystemen
Springer-Verlag 1987

[Ro] H. Rotter
Moraltheoretische Fragen zum Hirntod.
Aus: Intensivbehandlung, Jahrgang 11, Nr. 1/1986
pp. 22-25

[Sch] Horst Josef Schubert
Datenbank und Diagnoseunterstützung für Patienten mit Verdacht auf Hirntod
Diplomarbeit TU Graz, 1988

[Wal] A. Earl Walker
Cerebral Death
Urban & Schwarzenberg, 3rd Edition, 1985

[Whi] A.P. White
Inference Deficiencies In Rule-Based Expert Systems. Aus: Research and Development in Expert Systems, Ed. M.A. Bramer, pp. 39 - 50
British Computer Society Workshop Series 1985

B R A I N D E X

```
Klinische EEG-Bewertung     RESTAKTIVITÄT
Hirnläsion      SEKUNDÄR

Konstellation: sekundärer Hirnschaden, Hirnstammareflexie und Apnoe bei
NICHT isoelektrischem EEG

===> suspekter isolierter Hirnstammtod

Diagnose Hirntod ist erst nach isoelektrischem EEG zu stellen!

Schwebezeit: mindestens 72 Stunden

Konsequenzen:  EEG-Kontrollen
               AEHP
               VER

Lit. [31]

** Frage zu dieser Konstellation - bitte RETURN drücken
```

Fig. 1: Result of a consultation

B R A I N D E X

```
Klinische EEG-Bewertung     RESTAKTIVITÄT
Hirnläsion      SEKUNDÄR

Konstellation: sekundärer Hirnschaden, Hirnstammareflexie und Apnoe bei
NICHT isoelektrischem EEG

===> suspekter isolierter Hirnstammtod

Diagnose Hirntod ist erst nach isoelektrischem EEG zu stellen!

Schwebezeit: mindestens 72 Stunden

Konsequ┌Ihre Beurteilung─────────────────────────────────────┐
       │                                                      │
       │  Sie können obige Werte:      tolerieren             │
       │                               vernachlässigen        │
Lit. [3│                               nicht tolerieren       │
       │                                                      │
       └──────────────────────────────────────────────────────┘

** Frage zu dieser Konstellation - bitte RETURN drücken
```

Fig. 2: Asking for physicians's judgement

```
RULE247
=======
   SUBJECT :: HT_UNTERSUCHUNG-RULES
   DESCRIPTION :: (EEG iso und sek. Hirnläsion, kein Widerspruch bisher)
If    1)  1) Hirnläsion is BEIDES, or
          2) Hirnläsion is SEKUNDAR, and
      2) Klinische EEG-Bewertung is ISOELEKTRISCH, and
      3) (VALUE-OF WIDERLIST) is equal to ,
Then  1) it is definite (100%) that Andere Gründe für isoelektrisches EEG is
         SEK_AREFLEX, and
      2) evaluate (D (QUOTE (EEG_KLINISCH HIRNLASION)) (QUOTE (EEG_VORBICHT))),
         and
      3) it is definite (100%) that Klinische Beurteilung des EEG fur Hirntod
         erklärt is ERKLARBAR, and
      4) evaluate (WAIT), and
      5) setze erforderliche Schwebezeit neu, falls notwendig.

   IF    :: ((HIRNLASION = BEIDES OR HIRNLASION = SEKUNDAR) AND
            EEG_KLINISCH = ISOELEKTRISCH AND (VALUE-OF WIDERLISD EQUAL
            QUOTE ))
   THEN :: (EEG_VORSICHT = SEK_AREFLEX AND (E (D (QUOTE (EEG_KLINISCH
            HIRNLASION ) ) (QUOTE (EEG_VORSICHT)) ) ) AND
            HT_EEG_KLINISCH_ERKL = ERKLARBAR AND (E (WAIT)) AND (SET 72) )
   PREMISE :: ($AND
               ($OR
                 (SAME FRAME HIRNLASION BEIDES)
                 (SAME FRAME HIRNLASION SEKUNDAR))
               (SAME FRAME EEG_KLINISCH ISOELEKTRISCH)
               (EQUAL*
                 (VALUE-OF WIDERLIST)
                 (')))
   ACTION :: (DO-ALL
              (CONCLUDE FRAME EEG_VORSICHT SEK_AREFLEX TALLY 100)
              (E
                (D
                  ('
                    (EEG_KLINISCH HIRNLASION))
                  ('
                    (EEG_VORSICHT))))
              (CONCLUDE FRAME HT_EEG_KLINISCH_ERKL ERKLARBAR TALLY 100)
              (E
                (WAIT))
              (SET 72))
```

Fig. 3: Corresponding rules in the knowledge base

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics. Conf. p. 217-225.

A FRAME-BASED APPROACH TO PROTOCOL ENGINEERING

Péter Ecsedi-Tóth
SzKI, Computer Research and Innovation Center
H-1015, Budapest, Donáti u. 35-45. Hungary

Abstract We shall present a frame-based language specially tailored according to the needs of communicating protocols. It will be shown how this language can be used in a natural manner for specification of PDU's and the communication itself. The resulting specifications fulfil two fundamental requirements;

- first, they are comprehensible even by the nonexpert user;
- second, they are executable directly by machines.

Keywords; knowledge representation, frame-based language, protocol specification, protocol engineering.

1. INTRODUCTION

Recently, it has been realised by computer scientists that using the methodology of Artificial Intelligence may help considerably in reducing the complexity of intricate problems. One examole for such a problem is the proper treatment of communicating protocols. Indeed, protocols may exhibit very intricate behaviour for being paralelly running, nondeterministic and communicative processes. Investigation of applicability of this new technology for the specification, verification and testing of protocols has begun and now is on its way.

Our approach follows this line of thinking, i.e. we wish to contribute to the investigation of applying AI methodology for protocol engineering problems. Actually we are going to present a frame-based language specially tailored according to the needs of communicating protocols. It will be shown how this language can be used in a natural manner for specification of PDU's and the communication itself.

Albeit protocols on the lower layers of the OSI hierarchy are fixed (and a good deal of them is implemented by software or hardware tools), a large number of different new protocols will appear again and again on the application layer. The frame-based language suggested here can be used with ease for describing these new applicational protocols. The resulting specifications fulfil two fundamental requirements:

- first, they are comprehensible even by the nonexpert user;
- second, they are executable by machines.

Problems connected to protocol engineering can be devided into the following four groups:

- formal specification of protocols and services
- (automated) implementation of the specification
- validation (verification and testing)
- conversion of different specifications.

The specification problems play an outstanding role in all areas of protocol engineering, and so we are considering primarily these problems.

2. FRAMES, AN INFORMAL DESCRIPTION

2.1. The philosophy

Frame-based systems usually treat information on three different levels: the level of frames, that of the frame-structures and finally, the level of worlds.

Generally speaking, one can say that a frame is a structured symbolic model of a concept. Actually, it can be considered as a (usually ordered) set of arguments, called slots, each of which is used to represent a property of the concept to be modelled. Now, every slot may have a value, which is another frame or is a kind of expressions in some formal language. On the other hand, any part of a frame can be omitted if convenient; the omitted parts are considered hidden by the system.

2.2. Meta-information

In order to cope with the complexity of real world concepts, one may associate information ("meta-information") to any part of a frame, that is, to the name of the frame, to the slots and to their values. The meta-information is also formalised in frames which are called meta-frames. The pieces of meta-informátion are stored in the slots of the meta-frame.

In this paper we shall use only some meta-frames composed of the following slots:

- default
- range
- cardinality
- demon.

The intuitive meaning of these slots are quite familiar; the interested reader can find more details in [1].

2.3. Frame-structures

The second level of handling information in frame-based systems is the level of frame-structures. These structures are used to represent the relations among the concepts modelled by the frames. The idea behind this formalization is that if certain slot (and its value, if any) are not found in a particular frame, then a search is made by the system along the relations to see whether the slot and its value are contained in another frame accessible from the present one along such relations. If so, then the slot and its value will be "inherited" by the original frame.

Here we shall use only the familiar relation called "is_a" ("a_kind_of", or "instance_of"). If two frames are in an is_a relation, then we say that the first frame is an "instance" of the other, and this means that the instance will inherit all lacking information from the other one.

2.4. Worlds

The third level of handling information is provided by the possibility of forming groups from frames, in order to describe e.g. the so called "possible (or alternate) states" of affairs.

A world can represent hypothetical alternatives, or can be used as a tool of representing states of a process that change with time. Worlds can be manipulated in several different ways.

3. CONCEPTS IN PROTOCOL SPECIFICATION

Solving the problems arising in protocol engineering, we must consider three different aspects:
- architectural considerations
- functional considerations, and
- description of messages.

In this section we shall discuss the three aspects in details.

3.1. Architectural descriptions

In order to decrease the inherent complexity of protocols, they usually are divided into different components. The components obtained by such decompositions are connected to each other through some interaction points called gates. Then, the specification of the decomposition of a protocol must describe the following three parts: the components themselves, the gates and the way of composition. Below we present four frames which can be used to describe the general architectural aspects of decomppositions. (We shall use the following notations. Frames will be surrounded by the keywords frame and end. The slots and their values are separated by colons, and we shall use semicolons after the name of the frame and also after the values of the slots. Meta-information will be denoted also by keywords; here we shall use only the metaslot...end pair.)

3.1.1. The component frame

At this point we wish to specify the component from the outside, i.e. as if it was a black box; the inner activity of the component will be described later. The component, however, can communicate with its environment through the gates. We shall distinguish among input gates, output gates (which are unidirectional gates) and bidirectional gates given in the three slots of the following frame. To each slot a meta-frame is associated which defines defaults for these slots. This means, that, e.g. if we give the definition of a particular component without specifying value for one or more slots, then the value none will be inherited from this frame. Thus, using this default/inheritance machinery we may reduce the length of descriptions.

```
frame component;
   input_gates:;
      metaslot default:none;
      end
   output_gates:;
      metaslot default:none;
      end
   bidirectional_gates:;
      metaslot default:none;
      end
end
```

As an example, one may define the OSI transport_protocol_entity (tpe) in the following way:

```
frame tpe;
   is_a:component;
   bidirectional_gates:upper, lower;
end
```

3.1.2. The gate frame

The gate frame simply describes a gate. In this frame we may indicate the components among which the gate is located (in the "between" slot), the direction of it, and we can list all messages which may pass through the gate. Finally, for a finer study, we may specify its "capacity" (size) and "speed", (the user may augment these two slots by meta-information, if necessary).

```
frame gate;
   between:;
   direction:;
      metaslot default:bidirectional;
               range:bidirectional,from_1_to_2,from_2_to_1;
      end
   messages:;
   capacity:;
      metaslot default:infinite;
      end
   speed:;
      metaslot default:infinite;
      end
end
```

For illustration, let us define the gate named "upper" of the tpe above.

```
frame upper;
   is_a:gate;
   between:user,tpe;
   direction:bidirectional;
   messages:T_CONreq,T_CONind,T_CONresp,T_CONconf,T_DISreq,T_DISind,T_DATAreq,T_DATAind
end
```

3.1.3. The composition frame

The connections among the different components are specified in two steps. First we define the composition of two components only in the "binary_composition" frame, and then we generalise to more complex situations. In the binary_composition frame, we shall give the two components which are connected to each other and the gate(s) which are used to realise the connection. Furthermore, we specify the messages between these components.

```
frame binary_composition;
   components:;
      metaslot cardinality:2;
      end
   messages_from_1_to_2:;
   messages_from_2_to_1:;
end
```

For illustration we define the binary composition of two tpe's: tpe1 and tpe2 (which are assumed to be defined previously working in class 0).

```
frame tpe1/tpe2;
   is_a:binary_composition;
   components:tpe1,tpe2;
   messages_from_1_to_2:CR,CC,DR,DT,ER;
   messages_from_2_to_1:CR,CC,DR,DT,ER;
end
```

More general compositions can be specified by using the "composition" frame, described below. In this frame we shall give what binary compositions are used among what components. Moreover, in order to be able to build a "larger" black box consisting of the composed components, we may define the "visible" and "invisible" gates of the composite. The visibility is meant from the viewpoint of the observer being outside the larger box.

```
frame composition;
   components:;
   binary_compositions:;
   visible_gates:;
      metaslot default:all;
      end;
   invisible_gates:;
      metaslot default:none;
      end
end
```

For the sake of illustration let us suppose that the components tpe1, tpe2, and sp (service_provider; the union of the network and the lower layers) and their binary compositions tpe1/sp and tpe2/sp are defined. Then, the whole protocol can be defined as follows:

```
frame transport_protocol;
   is_a:composition;
   components:tpe1,tpe2,sp;
   binary_compositions:tpe1/sp,tpe2/sp;
   visible_gates:upper1,upper2;
   invisible_gates:lower1,lower2;
end
```

3.2. Functionality of components

Having defined components (as black boxes) and their compositions, we may turn our attention to the description of the inner structure and functionality of the components.

The inner activity of a component is most appropriately described by adopting the extended finite state machine approach. Accordingly, we may describe the activity by making use of "snapshots" representing the state of affairs at relevant moments, and then specifying the dynamic "transitions" among these snapshots. In a certain sense, we will be in analogy to the description of compositions: first we give the "transitions" between two states (the "binary" step) and then we give the "declarative" part (the general step) which describes the possible states and transitions among them as a whole.

3.2.1. The transition frame

The skeletal information on a transition can be given in four items: the (present) state, the input event, the output event, and the next state. These four items will be given in the four slots (state, input, output, and next_state) in the frame below.

In real applications, however, transitions may contain (or depend on) more information. For example, in the popular "extended finite state machine" approach [2], one may provide also information about the conditions when the transition is enabled and what additional actions should be done during the transition; this information will be given in the slots "enabling_predicate" and "action".

Another important property of transitions is how they handle the "passing" information (the parameters). Parameters may originate from three different sources:

- the ones of the first group are "left" in the (actual) state by the "previous" transition (for example, a procedure attached to the previously executed transition computes some value which is stored in a variable of the description of the actual state)
- the members of the second group arrive by inputing some messages
- finally, the parameters of the last group are computed by some procedure attached to the present transition; i.e. they are "born" in the actual state.

The parameters originated from any of these sources may be manipulated by the procedures of the present transition. Then, in analogy to their sources, three things may happen with them:

- they may be left in the state for the next transition
- they may be output in a message, and finally
- they may die (i.e. they are neither stored nor output).

This groupping of parameters may be specified in the following slots: parameters_from_predecessor, parameters_to_successor, parameters_from_input, parameters_to_output; all other parameters are handled as of the third type.

Furthermore, we may give some other information on transition, too. For example, we can specify how and where the transition may be interrupted, what is its time duration e.t.c.

The resulting frame is given below:

```
frame transition;
   state:;
   input:;
   output:;
   next_state:;
   enabling_predicate:;
   action:;
   parameters_from_predecessor:;
   parameters_from_input:;
   parameters_to_successor:;
   parameters_to_output:;
   interruptability:;
   interruption_points:;
      metaslot range:(before_input, after_input, before_action, after_action, before_output, after_output)
      end
   can_interrupt:;
   time_duration:;
      metaslot default:0;
      end
end
```

Let us see an example. A typical transition of the transport protocol is described as follows;

```
frame T_CONreq/N_CONreq;
    is_a:transition;
    state:closed;
    input:T_CONreq;
    output:N_CONreq;
    next_state:wait_for_network_connection;
    enabling_predicate:network_connection_is_not_available;
    parameters_from_input:called_address,calling_address,...;
    parameters_to_output:called_address,calling_address,...;
    interruptable_by:T_DISreq/DR,DR/T_DISreq;
end
```

3.2.2. The declaration frame

In this frame we may specify the set of all states and transitions of a particular component. Of course, these data are available in the descriptions of transitions, nevertheless such redundant information may increase readibility, and thus it can be usefull.

```
frame declaration;
    component:;
    states:;
    transitions:;
    input_messages:;
    output_messages:;
end
```

As an example, we define the declaration of the whole connection establishment phase of the transport protocol entity working in class 0.

```
frame declaration_of_connection_establishment;
    is_a:declaration;
    component:tpe;
    states:closed,wait_for_network_connection,wait_for_connection_confirmation,open,wait_for_connection_resp);
    transitions:T_CONreq/N_CONreq,N_CONconf/CR,CC/T_CONconf,T_CONresp/CC,CR/T_CONind,N_DISind/T_DISind,"
            T_DISind/N_DISind,T_DATAreq/DT,DT/DT,DT/T_DATAind;
end
```

3.3. Description of messages

The messages needed in the problems related to protocols are of several different types. As examples, we can mention the Service Primitives, or the Protocol Data Units (PDU's) at different OSI layers. The description of these different types of messages depends on the particular place where they are used. Hence, we do not suggest a general frame for describing all types of messages. Instead, we provide some frames for describing the PDU's of the transport layer (TPDU's). Though the given frames are particular examples with restricted usage only, the ideas behind the definitions generalise to other circumstances easily.

Following very closely the definition of the standard, we may use the following frames for describing TPDU's.

```
frame tpdu;
   length:;
      metaslot format:binary;
               range:0,1,...,254;
      end
   fixed_part:;
   variable_part:;
   data_field:;
end

frame variable_part;
   parameters:;
      metaslot range:sequence_of_"parameter"_instances;
               cardinality:1255;
      end
   checksum:;
      metaslot constraint: if class is not 4 then empty;
      end
end
```

```
frame fixed_part;
   tpdu_mnemonic:;
   tpdu_code:;
   credit:;
   destination_ref:;
   source_ref:;
   other:;
end

frame parameter;
   parameter_code:;
      metaslot format:binary;
               range:0,1,...,255;
      end
   parameter_length:;
      metaslot format:binary;
               range:0,1,...,248;
      end
   parameter_value:;
end
```

The following frame specifies the CR transport data unit:

```
frame connection_request;
   is_a:tpdu;
   fixed_part:frame tpdu_mnemonic:CR;
                    tpdu_code:1110;
                    credit:0;
                    destination_ref:0;
                    source_ref:;
                    other:fr_other;
              end
   variable_part:frame parameters:tsap_id,tpdu_size,version_number,security_parameters,...;
                 end
end

frame fr_other;
   class:;
      metaslot range:0,1,2,3,4;
      end
   option:frame format:;
                  metaslot range:normal,extended;
                           default:normal;
                  end
               explicite_flow_control:;
                  metaslot range:yes,no;
                           default:yes;
                  end
          end
end
```

A parameter, e.g. the tpdu_size, can be specified as follows:

```
frame tpdu_size;
   is_a:parameter;
   parameter_code:1100 0000;
   parameter_length:1;
   parameter_value:;
      facet range:8192,4096,2048,1024,512,256,128;
            constraint:if class=0 then (range:2048,1024,
                                         512,258,128);
            default:128;
      end
end
```

4. Using the frame-based specification as a knowledge-base

The specification obtained by using the frames described above defines the protocol completely. If, however, we would like to put the whole thing into work (automatic implementation), then we must specify the "active" part of the system, too. This active part works as an "inference engine" over the knowledge base consisting of the specification. We have two different possibilities to formalise this "inference engine". Traditionally, we may give it as a separate procedure which uses the frames in the knowledge base as abstract data types. In most frame based systems such procedures can be formalised easily as frames. The other way, fitting better to the philosophy of the frame-based systems, is to divide the activities into little pieces and associate these pieces to different frames as demons. In this way the "inference engine" is distributed over the specification and thus, activities are localised to the appropriate places where they should be used. Because of the lack of space, we do not treat here definitions for the "inference engine"; they will be presented elsewhere. Doing so, the given specification can be considered as the main body of the (experimental) implementation of the protocol in question. Similar is the situation with the validation and conversion of the protocols; in fact, one can augment the specification with demons or separate frames defining procedures to cope with the problems [3].

References

[1] Barr,A., Feigenbaum,E.A., The Handbook of Artificial Intelligence I-III. (second ed.) HeurisTECH Press, Stanford, 1986.

[2] Chung,R., A methodology for protocol design and specification based on an extended state transition model. Proc.ACM Sigcomm'84, Symp. Comm. Arch. & Protocols, Montreal (Canada) 1984, pp.34-41.

[3] Choquet,N., Fribourgh,L., Mauboussin,A., Runable Protocol Specifications Using the Logic Interpreter SLOG, in Protocol Specification, Testing and Verification V. (ed. M.Diaz), Elsevier Science Publishers, 1986.

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 227-231.

KNOWLEDGE BASE MANAGEMENT ON THE PC.

P. Koch
Computing Applications and Services Company
H-1502 Budapest 112. P.O.B. 146.

Abstract. In this paper we briefly discuss some technical aspects of building knowledge bases (KB) within the limits of the PC. A simple method called Knowledge Base Partitioning and Chaining (KBPC) is presented which can manage KBs breaking these limits.

Keywords. Expert system, knowledge base partitioning and chaining, expert system shells, taxonomy network.

1. INTRODUCTION

In the last few years expert systems - probably the most pragmatic applications of artificial intelligence (AI) - has entered the way of commercialization. These means that large number of tools and applications are offered for sale for users working in business, technology, medicine, geology etc. The hardware for these products ranges from the PCs up to the specific LISP machines and the power of the tools increases accordingly. Due to the wide variety of tools (shells) for different type of applications the knowledge engineers (KE) tend to prototype and refine expert systems using these tools instead of direct AI programming in LISP or PROLOG. The polarization within ES development is clearly reflected by the capabilities of the tools on the market [1]. Large, strategic applications require high-power, high-cost software (KEE, ART, KNOWLEDGE CRAFT, LOOPS etc.) and costly LISP machines to run them. On the other end of the range small-scale expert systems may need only an IBM XT or AT and a developmental tool of a few hundred USD (INSIGHT 2, EXSYS, M.1, PC PLUS etc). This latter group of tools and applications is what we call "low-tech" AI. In the following we discuss the problem of knowledge base (KB) management in low-tech expert systems.

2. THE KNOWLEDGE BASE AND THE HARDWARE

It is well known that the crucial issue for the performance of an expert system is its knowledge base. The three most important factors influencing the usability of a KB are:

- the quality
- the complexity   of the knowledge in the KB,
- the amount

Here we do not discuss the first two issues roughly saying that the larger the knowledge base the better the ES works. Different types of problems may require different knowledge representation (KR) formalisms to apply [2]. The most commonly used KR techniques are

- production rules
- frames
- semantic networks

or the combinations of these. While the production rules are relatively easy to implement on the PC, the other two techniques are rather memory consuming. Due to this, practically ftil PC range ES shells and applications use rules to formalise domain expertise and meta-knowledge applied. This obviously restricts the complexity of the knowledge we can use which, in turn, decreases the "expertise" of PC-range expert systems. As its name suggests, "low-tech" AI has the primary goal to implement the simplest AI (at present mostly ES) techniques on the low-end computers, that is, on the PCs (by PC here we mean the IBM PC and compatibles, although the McIntosh is coming up). The idea behind this strategy is likely to make the elements oi this relatively new and promising technique available for a large number of end-users at reasonable investment (sw, hw, mental effort). For low-tech tool vendors the above say that their shells should run on "standard" configuration IBM XT/AT machines under MSDOS (it should be noted that the spread of 38b-based machines, PS2/OS2 and workstations may change the market). So the task is the development of expert systems with KBs (rule bases) large enough to produce acceptable advices as "domain experts" and small enough to work under MSDOS.

3. ALTERNATIVES FOR THE PC

The task formulated in the previous section is somewhat contradictory. ES and shell developers usually adopt one of the following methods to cope with this problem

- the use of plug-in extension cards
- the compilation of the KB
- the use of traditional (non-AI) programming languages to implement the tools.

Plug-in cards like EMS or Gold Hill"s HummingBoard are good solutions if do not insist on the use of "standard" configuration. The other two techniques need no extensions. Instead, they try to shrink the knowledge base (without the loss in the amount of knowledge) and the code that makes the knowledge work. The KB compilation may effectively free memory space but the elements of the compiled KB cannot be modified without recompilation. The use of traditional languages (mainly C and Pascal) is a strong tendency amongst the tool developers. Even if good results can be achieved neither of the latter two methods offer a final solution for the problem. During the phase of ES refinement and extension the amount of knowledge (the number of rules, objects etc) rapidly grows exceeding the capabilities of the tool. In the following section we briefly describe a method called KB partitioning and chaining (KBPC) which proved to be useful in solving the task above.

4. KNOWLEDGE BASE PARTITIONING AND CHAINING (KBPC)

The idea of KBPC comes from the simple observation that in a number of cases a larger KB can be subdivided into essentially independent partitions. For instance the KB partition (subset of rules) used by a medical expert system to interpret clinical lab data for a patient normally independent from the rules used to reach a final diagnosis or presenting treatment recommendations. These partitions are usually not perfectly separated: they communicate through a set of common KB elements (objects and hypotheses). These common elements represent the intermeaiate results archieved during the evaluation of a partition and passed to one or more other partition(s). In technical Terms each partition can be represented by a triple, <Ri,Hi,Oi>, where Ri, Hi, Oi are the ith subsets of the set of rules (R), hypotheses (H) and objects(O), respectively. Each triple is a "self-contained chunk" of knowledge, that is, a set of hypotheses to be verified, the associated rules to infer them and the objects used by the rules. Discovering the real partitions in the knowledge of the domain expert is not an easy task since in most of the cases the expert is unable to articulate his/her expertise [3]. It depends on the skill of the knowledge engineer to find meaningful partitions during the phase of knowledge acquisition. There are many ways to implement the KBPC method. Here we briefly outline a technique we adopted for our ES shell, GENESYS.
For backward systems we use a method called 'hypotesis ordering'. This can be performed in the following ways:

- direct ordering
- using problem taxonomy network.

In case of direct ordering the knowledge engineer defines the partitions over H and the system automatically constructs the associated subsets of R and O. The result of this processis a set of (usually related) triples.
When using the problem taxonomy network the knowledge engineer is allowed to formulate the explicit description of the taxonomic relations among the elements of the problem space (i.e. the hypotheses to be verified) using the hypothesis net(s) (HN).

For instance, in a medical application the knowledge engineer can construct the taxonomy of a disease family starting with the generic disease (as the root node) and completing the network with the specific diseases to be diagnosed (as terminal nodes). In this case the HN determines the "chains of reasoning" leading to the terminal nodes (final diagnoses). The set nodes (hypotheses) along these chains are the different partitions of H. Again, the system collects those and only those rules and objects for a certain partition of H which are necessary for the chain of reasoning represented by that partition.
For forward cases we use the question nets (QN). The technique by which QNs support KB partitioning is similar to that of HNs, so we do not discuss it here. Note, that apart from KB partitioning, QNs are very useful for explicitly and dynamically control the way of data collection (questioning) during the consultation.
We mentioned that the KB partitions are not completely disjunct. For instance, rules in one partition may infer value(s) for an object in another partition which uses this object to reach its own conclusions. KB chaining is the way to organize the activisation of KB partitions and the information flow between them. Chaining means swapping, too. When the system evaluated a partition and determined which partition is to call next it saves the intermediate data and the system status and loads the selected KB partition from disk. After that, using the intermediate data the system continues problem solving by evaluating the loaded KB partition. Selecting the partition to be loaded performed dynamically taking into account the current state of the consultation. By the use of KB chaining loops can also be organized: this means that the same partition can be reevaluated later if necessery.
KB chaining allows the knowledge engineer to navigate over a larger KB keeping only that partition in the memory which is necessary to work with. The KBPC method proved to be useful for medical applications in our practice and in others using GENESYS.

5. ARGUMENTS FOR AND AGAINST

In this section we list the advantages and shortcomings of KBPC in comparison with the techniques mentioned in section 2.

5.1. Advantages of KBPC

The KBPC method has the advantage over the others in section 2., that it "breaks" the memory limit of MSDOS and the total amount of knowledge used by an ES do not depend on physical memory available. The knowledge engineer has to keep in mind that the size of only a single partition is restricted. Using KBPC he can take the advantages of a much larger KB than a single partition fitting in the memory. Another advantage is the high level modularity of the KB. This makes the KB maintanance much more convenient and safe because individual partitions can be modified independently and the effect of the modifications are usually do not propagate over the other partitions.

5.2. Shortcomings of KBPC

Unfortunately, there are some shortcomings of the method as well. First of all, partitioning not always possible, and even if it is, the construction of meaningful partitions strongly depends on the skill and experience of the knowledge engineer. From technical point of view the administration of KB swapping and chaining during the course of consultation increases the response time which may cause inconveniencies using the system. Furthermore, the "swapping time" depends also on the organisation and the maintenance practice of the (hard) disk a factor changing at different users. Another disadvantage of KBPC is the more complex control structure which results in larger size of the run-time code which, is turn, decreases the maximal size of a single partition.

6. SUMMARY

In this short paper we discussed some problems of KB management on the PCs. We outlined a technique called KBPC to "break" the memory limit of standard configuration PCs. In comparison with other techniques (plug-in cards, KB compilation etc) we cah srape that KBPC has the advantage that the size of only a single KB partition is limited by the physical memory and not the total amount of knowledge. The main problem with KBPC is that there are cases when partitioning is imposssible and the whole nex:noa strongly depends on the skill of the knowledge engineer.

REFERENCES

1. Expert System Strategies, Vol. 4, No. 1, 1988.

2. Klahr, P., Waterman, D.A. Expert Systems. Addison Wesley Publishing Co., 1986.

3. Kohout, L.T., Bandler, W. Knowledge Representation in Medicine and Clinical Behavioural Science. Abacus Press, 1986.

X'

I

BEYOND NUMBER CRUNCHING Third Austrian-Hungarian Informatics Conf. p. 233-243.

TOWARDS AN INTEGRATED VIEW OF DATA PROCESSING, ARTIFICIAL INTELLIGENCE AND SOFTWARE ENGINEERING

P. Krauth
Software Department
Research Institute for Measurement and Computing
Central Research Institute for Physics
Hungarian Academy of Sciences
H-1525, P.O.B. 49, Budapest, Hungary

Abstract. The three area of computing (DP, AI, SE) have traditionally different user and developer communities, and incompatible product sets. The article tries to identify a viewpoint from which these areas look similar. Having surveyed the currently available products, it is recognized that a knowledge-based dictionary is or can be in the core of any DP and AI-application and SE-environment. This fact together with a current standardization process can substantially inluence the way of how software will be manufactured in the future.

Keyword. Artificial Intelligence, Data Processing, Software Engineering

1. INTRODUCTION

In the eighties there have been rapid and successful developments in the research and application of data processing (DP), artificial intelligence (AI) and software engineering (SE) technics, as exemplified by the current proliferation of products in all of these areas. The fact, however, that the primary motives behind these areas are fairly different, resulted in markedly separated user/research communities and created substantial mutual misunderstandings.

In particular, the main goal in data base research is to achieve maximum performance in terms of extremely large data, number of concurrent users and response times. On the other hand, artificial intelligence research strives for increasing the functionality of a system, incorporating complex knowledge rather than bare data only, and simulating human-like behaviours such as problem solving, planning etc. Finally, the software engineering activities concentrate on the overall quality of a system balancing system requirements against maintainability, ensuring user satisfaction and applying consistent technics during the entire life of the system.

However, despite all of their differences, the application of DP, AI and SE technics is identical in one respect: each of them improves our abilities to produce more complex software systems and to utilize technological progresses in hardware. At a first glance, this seems to be fairly obvious statement, but it represents an important shift in emphasis and establishes a good basis for developing an integrated view of DP, AI and SE. Looking at these areas of computer science from this very pragmatic point of view, has a lot of advantages. In this way, for example, it becomes possible to get rid off all of the misjudgements attached to DP, AI and SE. Just to name a few of them:

- data processing is nothing but a rather complicated way for handling simple data files,
- with the available technics of AI (even on a PC) one can get close to the understanding of human thinking,
- with the advent of software methodologies, the problem of producing quality software has been solved once and for all.

One can also recognize an on-going cross fertilization of ideas and technics emerging from the aforementioned areas. Last but not least, this viewpoint could also be useful in trying to predict future developments.

In Section 2, 3 and 4, first we give a summary of the achievements and the apparent shortcomings of the different technics in the areas of DP, AI and SE, respectively. Then, in Section 5, 6 and 7, we report products currently on the market making good use from the interaction of the various technics. In Section 8, as conclusion, we try to outline the shape of possible future systems based on the integration of DP, AI and SE.

2. DATA PROCESSING

The overwhelming majority of today's applications is data processing software, and, because od this, based upon some kind of database. This is why the practical importance of database management systems (DBMS) cannot be overemphasized.

Traditionally, the main technical problems which a DBMS has to cope with, are:

- concurrency control in a multi-user application environment
- effective management and storage of extremely large amount of data (from hundreds of megabytes to gigabytes)
- recovery capabilities to protect data against system malfunctions

The first problem was solved by the introduction of the concept of transactions and the related transaction management algorithms. The second challenge enforced the page segmentation of the databases and the invention of powerful data access mechanisms (balanced tree, hash coding, logical pointers and so on). Finally, the recovery problem has been eliminated by backup/restore functions and continuous journaling.

Another technical problem which has been gaining increasing attention since local area networks had appeared, is the physical distribution of data throughout a network. Although distributed database management systems are already on the market (e.g. INGRES, ORACLE, SYBASE, EMPRESS etc.), these products have not yet matured enough and much more improvements can be expected in the near future. Nevertheless, their appearence means that the fundamental logical and algorithmical difficulties have been already overcome, and heralds a new era in DP-applications.

Turning to more conceptual problems, it must be noticed that the database research and development community have long been confronted with the problem of data modelling. From the very early years, diagramming technics (e.g. Bachmann-diagram, Chen's entity-relationship diagram), data description languages (e.g. CODASYL DDL) and data dictionaries (e.g. IDMS's IDD) have been developed and used to aid the modelling activity. However, the usefulness of such tools and technics has been severely restricted, since the main contradiction between the physical level effectiveness of a particular DBMS and the logical expressive power of its data description method, could not be easily resolved. The former requirement is essential at run-time and for the end-users. The latter is crucial for the development team during the entire life of the application, and consequently, regarding the delivery time of the amendments or enhancements, for the end-users, too. This problem is rooted in the fact that, traditionally, a DBMS is evaluated primarily against run-time effectiveness, and, to meet the performance criterias, the modelling capabilities were always compromised.

However, as a result of substantial progresses in hardware/software technologies (16- and 32-bit microcomputers, database machines, matured relational database technology etc.), the growing popularity of database softwares caused recently that inefficiency in database design has become a major obstacle in producing database applications.

3. ARTIFICIAL INTELLIGENCE

The desire or, better to say, the challenge to build a machine similar to a man, can be followed back to the medieval times. Therefore, it is no wonder that a research direction within computer science (called AI) was born to explore the feasibility of using computer technics to simulate certain human-like phenomena like speech, vision, thinking.

However, despite the great number of smart programs developed through years of time, AI is still in its infancy and far from the original goal: to understand how the mind works. Moreover, it turned out in between that those smart programs are not intelligent enough and there are opinions that they will never be.

What is then the use of AI? It stimulated brand new ideas and, as a by-product to the main stream research, we have now a collection of powerful technics whose usefulness has already been demonstrated by the proliferation of expert systems (ES).

The main characteristics of these technics are their flexibility. This was required by the human brain's apparent high-level adaptation ability, and has resulted in software construction methods based upon concepts that unify data and process descriptions and neglec the usual strict distinctions between them. Such construct is, for example, a LISP-expression which can be used for defining a data structure as well as for specifying a flow of control. Similarly, a Horn-clause in PROLOG can represent a data item (fact) and also a small process fragment (rule). The concept of frames (or objects or schemas or units or etc.) in the so-called frame-based (or object-oriented) systems, is an even more powerful way to combine data and processes where the values of the slots (a frame is divided into slots) can be either data or process description. Frames can also be organized into semantic network which, again, could behave either as a static or as a dynamic structure. The underlying mechanism for the latter is called inheritance, and enables for the frames to share easily properties.

The statement that data and processes are the same, is a fairly deep recognition. Beyond its philosophycal consequences, it has led to the notion of knowledge whose well engineered collection (knowledge base) can, in a knowledge-based application system, alone represent all the information about the static (data) and dynamic (processes) characteristics of an application domain. The knowledge base is usually coupled with an inference engine which serves as a domain-independent mechanism for driving the knowledge.

The above outlined, general architectural framework for knowledge-based systems works well in several ES-applications. Moreover, it works so well, and the ES-applications have become so important for their users that the so far neglected (much rather technical than conceptual) shortcomings of the current AI-tools and technics have caused serious bottleneck in the spread of AI.

The most important deficiencies are the lack of tools to develop distributed and concurrently accessible knowledge bases, and also the lack of ways to integrate AI-systems with the more conventional computer systems to enable for the users to follow an evolutionary, as opposed to revolutionary, growing path. This is becoming rapidly crucial because, after a transient period when the primary emphasis was on the personality of computing, the capability of a system to manage information coming from a variaty of sources (human or computer), is beginning to gain vital importance. Unfortunately, but understandably, AI has not paid too much attention to this problem until recently.

4. SOFTWARE ENGINEERING

Unlike AI, software engineering was born hand in hand with the traditional programming. The aim was first only to introduce disciplines into the otherwise intangible art of programming (structured programming) but gradually its scope has been extended to other software development activities like analysis, design, test, documentation etc.

Nowadays, software engineering covers all the phases and aspects of developing various sorts of software systems (business, real-time, batch, on-line) and the knowledge how to develop software is organized into <u>methodologies</u> (YOURDON, SSADM, Information Engineering, JSD etc.). They are common in that each employs a consistent set of <u>technics and procedures</u> to control the development activity, albeit each uses different <u>life-cycle model</u>. On one hand, a technic is a more or less formalized framework to carry out a specific task (e.g. to draw a certain diagram) and can often be automated, on the other hand, procedures are basically applications of technics and manpower to perform certain steps in the life of a software system (e.g. perform quality assurance review, make system test plan, control change requests, perform logical data modelling etc.).

Though methodologies appeared first in the mid-seventies their spreading was substantially delayed until recently (mid of eighties) since their use was too tedious or expensive without cost-effective <u>automated support tools</u>. By that time the wide availability of microcomputers/workstations and advances in user interface technics (windowing, graphic tools) had made possible a breakthrough in the application of SE. As a matter of fact, every support environment has two basic components:

- data dictionary
- graphic (or at least screen-oriented) interface.

Throughout the development <u>different types of objects</u> (diagrams, specifications, designs, test data, programs etc.) are to be managed and among them <u>multiple relationships</u> (a diagram is exploded into other diagram, a data group used in several data flow diagram, etc.) are to be maintained. In addition, it is well-known that the software development as a creative activity is iterative and parallel in nature. For example, when, during programming, it is frequently recognized that the design has to be modified, or a routine has to be adjusted to different hw/sw environments. Unfortunately, this phenomenon is usual for all kind of the above mentioned objects. It implies that the objects can exist in different <u>versions and variants</u>. This problem is solved by the use of configuration and version control technics which is provided by most of the current dictionaries. That is why the data dictionary in software engineering is much more important even if, from attractivity point of view, a good drawing interface is more appealing at the first glance.

The main problem today the methodologies and their supporting environments faced, is that there are too many methodologies and none of them good enough. In fact, they cannot be good enough because every real project and every company (after all the final purpose of a methodology is to be used in projects) have some particular requirements and characteristics that make hardly impossible to apply any standard methodology. That is why the real skill of a software engineer, is how to deviate from standards. But having developed a company- or project-specific life-cycle model and a corresponding methodology, what about the tools? The tools are marketed and supported by software companies (ammendments, enhancements, new releases, new hardware platforms etc.), consequently, must have a stable set of features. However, this prevents them to be used in a too wide variety of circumstances.

5. DATA PROCESSING AND ARTIFICIAL INTELLIGENCE

An important point here is that more systems that are implemented, the larger and more complex the overall system becomes. More islands of automation are created, but only rarely do islands coalesce; hence more gaps between systems are also created, this process often results in an unmanageable collection of information and tools from which the user finds it difficult to make an appropriate choice. Gaps are formed where we cross from automated systems to manual systems, human decision making, judgement and selection.

Integrating knowledge-based technology with data processing in business can close, or at least shorten, these gaps, and can make complex information systems appear more comprehensive and manageable to users. Moreover, it can extend the scope of automation, tackle the high complexity, volatility and uncertainity which are common in business activities, and increase competitive edge.

One way to realize such an integration, is to implement knowledge-based languages or systems enhanced with functions required in DP -environments. As a first example, we mention the TOP-ONE system. TOP-ONE is a comprehensive applications implementation and delivery environment for business computer systems, which contains a multi-user, mainframe implementation of the logic programming language Prolog, specially designed and developed for the business data processing environment by Telecomputing plc. It supports fully transaction processing oriented features such as concurrency control, resilience, recovery and security. It also be coupled with existing systems, accessing conventional data stores.

Another example is the G-BASE object-oriented database from Graphael. It can manage all types of data: alphanumeric, text, graphic, video, audio, programs and data from formerly incompatible databases. G-BASE can store virtually any amount of information without wasting critical resources, because all data on disk rather than virtual memory. Implemented in LISP, it enables LISP methods and functions to be directly linked with data, describing how to manage the data, check or display them. Unlike other systems, G-BASE permits the data model and structure to be modified at any time during the life of the database, and the modifications can be performed dynamically. This is a very useful feature to construct, for example, generic database applications. G-BASE ensures data integrity through strict control measures. As data entered or the structure modified, control functions can be activated to check the validity of the data and its relationships to other data. G-BASE also includes built-in security features to maintain data consistency in the case of a system crash or power failure.

The concept of object-oriented databases become increasingly popular, as other notable development efforts show. Among others, at MCC (Microelectronics and Computer Technology Corp.: the US fifth-generation undertaking) at least two projects are devoted to object-oriented DBMS development.

Another product called SAPIENS from Software Craftsmen Inc., uses expert systems concepts to application generation. The built-in expert system uses a comprehensive data dictionary as its knowledge base. The ERROS system from ERROS Computing Services Ltd., too, focuses on intelligent application generation but uses a theasaurus-based Inference Engine Database (IED) as data dictionary. It is said to be a new concept employing AI-technics that use user data and data definitions to drive applications. IED can maintain an historical or a snapshot database, allows roll back and recovery by journalling all database changes, and integrates three record structures: relational, hierarchical, network.

The other way for AI and DP integration is the most straightforward solution: to interface existing AI-tools to conventional DP-environments. The best-known product taking this approach is the KEE (Knowledge Engineering Environment) from Intellicorp Inc. which provides for the ES developers the international standard SQL (Structured Query Language) interface to access data in conventional DBMSs (e.g. ORACLE). It is rumoured that other leading AI-companies (e.g. Carnegie Group Inc.) are going to take this approach, too.

But not only the AI-companies feel the need to interface to conventional DBMS, a DP-product vendor (Information Builders Inc.) decided to buy a smaller AI-company and its product (Level 5) with plans to unify the latter with its own FOCUS system. A mainframe vendor (Bull) has gone even further by having developed a distributed knowledge base architecture (ABCD) where it interfaces through an Ethernet local area network three types of machines. One is devoted to user communication, the second is running an ORACLE database and the third stores a knowledge base written by Bull's own ES shell (Kool).

6. ARTIFICIAL INTELLIGENCE AND SOFTWARE ENGINEERING

The story of using AI and SE technics together is not at all new, and goes back to the beginning of the eighties when the AI-technics were beginning to come out of the academic research facilities. To illustrate this we mention the CALYPSO project at the Carnegie-Mellon Univ. to explore the feasibility and possible advantages of the use of AI in SE. A series of follow-on projects carried out for private companies, shows well the conclusion. Even, there are some software gurus who do not even make any kind of distinction between AI and SE which is obviously false for the time being but it might not be in the future.

There are two basic approaches in combining AI and SE. The more obvious is to use flexible AI-technics to implement SE-tools. This approach has been already touched in the previous section when data dictionary (DD) based application generation tools were mentioned, and the DD was basically a knowledge base. But these tools are restricted to the business computing area. There are other AI-based SE-tools which are, however, generic.

One of them is Graphtalk from Rank Xerox. It is build around the XAIE (Xerox Artificial Intelligence Environment), and provides data dictionary, graphic and project management tools, code generators, documantation aids and a lot more. The main feature is its <u>tailorability</u> which makes good use of its underlying flexible AI-languages (LISP, Prolog, Loops), enabling to introduce specificity either on the applied method or on the project or even on the organization level. In this respect, Graphtalk heralds a new horizon for the applicability of SE-tools.

Graphtalk is far from being alone with its approach. The Virtual Software Factory (VSF) from Systematica Ltd., for example, uses similar concepts providing user-configurable design support system. It incorporates an Integrated Knowledge Base System (IKBS) as data dictionary. The IKBS has extremely generic structure and is capable of containing both of the method-specific information (method model, validation rules, object types, graph types etc.) and user-supplied design data. Rule-based graphic and text design editors manage and present the required information on the screen after having been filtered through the method database portion of the IKBS. VSF can be configured to support whatever structured methods and documentation standards the user requires, and provides direct integration with other tools. Currently, as standard configurations, it supports 5 methodologies (YOURDON, SSADM, JSD, CORE, MASCOT 3), and is marketed aggressively at a very competitive price (7000$ for each). Describing the core of VSF which makes possible the tailoring of VSF to a method's requirements, Systematica also coined a new term: Methods Engineering Workbench.

The other approach in combining AI and SE, is to take the other way round. The question here is what are the benefits of using SE-concepts in AI-system development? One must not forget that, to whatever extent a system is knowledge-based, it is still software, and as such, it is subjected to the more general rules of the software life-cycles (i.e. it should be designed, constructed, tested, there are versions etc.). In this respect, for example, knowledge acquisition is nothing more than system analysis, and an equivalent to software engineering is knowledge engineering.

Unfortunately, there is no established methodology for the development of Expert Systems. However, there are some on-going works in this area as well. One (less important) example, is the Telecomputing plc. company's efforts (marketing TOP-ONE, section 5.) to develop methodology to identify potential business applications suitable for expert system techniques.

The other one is more significant. CCTA (Central Computing and Telecommunications Agency) in collaboration with the Department of Trade & Industry of UK has launched a national Expert Systems Initiative - the GEMINI Project (Goverment Expert sysrems Methodology INItiative). The project aims to establish an ES or knowledge-based systems (KBS) development methodology which will be made freely available to the information technology (IT) industry and users, thus following the example of SSADM, the well-known standard. They will take full account of the methods used to develop conventional IT systems. The method will offer inrerfaces to SSADM and it should also be able to interface to other methods to ensure that ES components and systems can be fully integrated with existing systems.

7. SOFTWARE ENGINEERING AND DATA PROCESSING

It is well-known that SE can help much in DP-development, in particular, in data modelling. Still, the conversion of a model into an effective physical database design is a tedious task. This problem is solved by the ZIM database product from Zanthe Information, Inc. Using this software one need not to convert the data model already in the form of entity-relationships terms into database schema definition, since the entity-relationship specification directly understandable by the ZIM database manager which itself is a full-feature DBMS with transaction processing, recovery and application development facilities.

But the above approach is not too common. It is more general to use a DBMS to implement a data dictionary which is, as we have seen, the core of any SE-tool. In fact, every DBMS uses some sort of data dictionary for storing the metadata of a database. Some DBMS-product went even further in this direction putting application (or other) data into its own dictionary (e.g, IDMS's Integrated Data Dictionary, ORACLE'S Data Dictionary). Moreover, DEC has announced recently a new release of its Common Data Dictionary, stating that it has certain object-oriented flavour using an entity-relationship-attribute model.

ORACLE Corp.'s activity in combining DP and SE technics, is especially remarkable. It does not only supply a large set of DBMS-product but, also, it is becoming a big SE-tool vendor integrating state of the art drawing facilities with its ORACLE-based development dictionary to be used within its own proprietary development methodology.

Finally, there is one more activity worth mentioning in this context. Standards organizations (ISO, ANSI) are currently working on an interface standard for the Information Resource Dictionary System (IRDS). It is the only standardization activity in the DB-field, and if it is succesful it can impact to a large extent the future developments in the DP as well as in the SE areas.

8. DATA PROCESSING, ARTIFICIAL INTELLIGENCE AND SOFTWARE ENGINEERING

The message of this article is very simple, almost trivial: conceptually, DP, AI and SE are the same. It is so because a flexible data dictionary (or knowledge base) must be an architectural component of any DP, or AI-system or SE-environment.

This statement has far-reaching implications. It suggests that the DB is the most important component of any application system. The question can then be raised: what about a real-time system which, because of performance reasons, cannot afford the extra overhead of the run-time data dictionary accesses? Obviously, there will be always systems which will not employ run-time DD. However, even their development environments will utilize some sorts of data dictionary. And what is the difference between the system as represented in the development environment and the system running in a production environment? The only essential one is performance.

It seems to us that, for every system, there are a development and a production version, and the two differ only in performance (or other non-functional) requirement. The situation is very similar to the one when somebody writes a document with the help of a word processor. There is a version of the document stored in the computer in a format which one can easily modify, and there is a printed version which can be used effectively (no need for computer, one can page it as he wants etc.). Analogically reasoning, the generation of the production version from the data dictionary which embodies all the knowledge about the system under development, will be as easy as printing a document, and one can generate a running system for different hw/sw environments with the same ease as one can print a document on different printers.

This is not to say, however, that DP, AI and SE-technics are currently interchangable, still we have got the feeling that they are coming closer and closer to each other, and, perhaps, it is not too risky to predict that in the next millenium the software development will be mostly knowledge acquisition. This again will not solve the problem of how to construct good quality software once and for all, but it will lift the problem onto a higher level.

LIST OF AUTHORS

ARATÓ, A.,	Central Research Institute for Physics, Hungarian Academy of Sciences, Budapest
BENCZÚR, A.,	Eötvös Loránd University, Budapest
BODÓ, Z.,	Computer and Automation Institute Hungarian Academy of Sciences, Budapest
BRÜCKLER, E.,	Institute for Machine Documentation, Graz
BRÜCKLER, H.,	Institute for Machine Documentation, Graz
CSABA, L.,	Computer and Automation Institute Hungarian Academy of Sciences, Budapest
CSER, L.,	Technical University of Budapest
CSUHAJ-VARJU, E.,	Computer and Automation Institute Hungarian Academy of Sciences, Budapest
CSURGAY, A.,	Hungarian Academy of Sciences, Budapest
ECSEDI-TÓTH, P.,	Computer Research and Innovation Center, Budapest
FELLNER, W.D.,	Institute for Information Processing, Graz
HAASE, V.H.,	Technical University of Graz
HARING, G.,	Institute of Informatics - University of Vienna
HERNÁDI, A.,	Computer and Automation Institute Hungarian Academy of Sciences, Budapest
HUBER, F.,	Institute for Information Processing, Graz
KAPPE, F.,	Institute for Information Processing, Graz
KNUTH, E.,	Computer and Automation Institute Hungarian Academy of Sciences, Budapest
KOCH, P.,	Computer Applications and Service Company, Budapest
KÓPHÁZI, J.,	Computer Research and Innovation Center, Budapest
KRAUTH, P.,	Central Research Institute for Physics, Hungarian Academy of Sciences, Budapest
LEWANDOWSKI, A.,	IIASA, Laxenburg
MALLE, B.,	Institute of Psychology - University of Graz
MÁRKUS, A.,	Computer and Automation Institute Hungarian Academy of Sciences, Budapest
MITTERMEIR, R.,	Institute of Informatics - University of Klagenfurt
MOIK, H.,	Technical University of Graz
MOLNÁR, C.,	Computer Research and Innovation Center, Budapest
OLASZY, G.,	Linguistic Institute, Hungarian Academy of Sciences, Budapest
PENZ, F.,	Institute of Informatics - University of Vienna
PFURTSCHELLER, G.,	Institute for Electro- and Biomedical Technology, Graz
RAUCH, W.,	Institute for Information Science - University of Graz
RUTTKAY, Z.,	Computer and Automation Institute Hungarian Academy of Sciences, Budapest
SCHULTER, G.,	Institute of Psychology, University of Graz
SCHWARZ, G.,	State Hospital, Graz
STÖGERER, J.K.,	Institute for Information Processing, Graz
SZŐTS, M.,	Computer Applications and Service Company, Budapest
TAMÁS, P.,	Technical University of Budapest
TSCHELIGI, M.,	Institute of Informatics - University of Vienna
VAINA, A.M.,	University of Tampere, Finland
VASPÖRI, T.,	Central Research Institute for Physics, Hungarian Academy of Sciences, Budapest
VÁMOS, T.,	Computer and Automation Institute Hungarian Academy of Sciences, Budapest
WEINHART, H.,	Institute of Informatics, University of Klagenfurt
WILLEGGER, B.,	Technical University of Graz
WINKLER, F.,	Research Institute for Symbolic Computation, Linz