LINKÖPING UNIVERSITY AND INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science

Annual Research Report 1985

Hus E
House E

Institutionen för datavetenskap
Department of Computer and Information Science

LiTH-IDA-R-86-05
ISSN-0281-4250

January 1986

The Department of Computer and Information Science

Linköping University

Annual Research Report 1985

This report describes research on software (and to some extent hardware) technology within the Department of Computer and Information Science at Linköping Institute of Technology, which is a part of Linköping University. Main areas of current research are programming environments, artificial intelligence, application systems, computer-aided design of digital systems, representation of knowledge in logic, complexity of algorithms, logic programming, library and information science, and administrative data processing. The department has a well organized integrated program for graduate studies (PhD and Licentiate degree), with a large faculty engaged in research and thesis supervision. In addition to the research organization and the extensive undergraduate course program, there is also a knowledge transfer program at the department where we cooperate with large Swedish companies on medium to long term R&D issues.

Mailing address:
Dept. of Computer and Information Science
Linköping University
S-581 83 Linköping
Sweden
Tel: int + 46 13 28 10 00

Postadress:
Inst. för datavetenskap
Universitetet och
Tekniska Högskolan i Linköping
581 83 Linköping
Tel: 013 - 28 10 00

CONTENTS

1.

Introduction

1.1 Organization of the department.

The Department of Computer and Information Science (IDA) was formed in 1983, bringing together groups previously in the Mathematics and the Electrical Engineering departments. It has presently about 110 employees (about 15 with a PhD) and activities are divided approximately equally between research and teaching. The research and graduate education program is organized in common for the whole department, but research projects are carried out within smaller groups, research laboratories, which typically consist of five to ten persons. The undergraduate teaching encompasses the three subject areas computer science (*datalogi*), telecommunications and computer systems (*telesystem*) and administrative data processing (*administrativ informationsbehandling*).

The research program is coordinated by the board for research activities and graduate education, headed by Erik Sandewall. The current laboratories are PELAB (Lennartsson) for programming environments, ASLAB (Hägglund) for application systems, AILAB (Tengvald) for artificial intelligence, CADLAB (Lawson/Lyles) for computer-aided design of digital systems, RKLLAB (Sandewall) for representation of knowledge in logic, and LIBLAB (Hjerppe) for library and information sciences. In addition there are special research groups for logic programming (Maluszynski) and for complexity of algorithms (Lingas), which are somewhat smaller but with a higher proportion of researchers holding a PhD. The group for administrative data processing, although primarily a group for undergraduate teaching, also includes some research activities.

The major funding for our research is supplied by the Swedish Board for Technical Development, STU, (85/86: 5.2 MSEK) In addition to the STU support, funds are also provided by the Delegation for Technical and Scientific Information Supply (DFI, 85/86: 0.8 MSEK), NFR (85/86: 0.08 MSEK), other sources (85/86: 0.4 MSEK) and the ordinary university budget for research and postgraduate studies (85/86: 2.6 MSEK). The joint program for knowledge transfer to industry has an additional budget (85/86: 2.1 MSEK), provided by the participating companies. At present 1 SEK is approximately 0.13 USD.

The recruiting situation at the department is presently very good. Faculty include 15 persons with a PhD, all of whom are contributing also to undergraduate education. Several of them have their PhDs from universities abroad and there are many nationalities represented at the department, contributing to an intellectually stimulating environment. There are also a number of persons with a backgrund in industry working part time at the department. The number of students applying for graduate studies is also very high, including also a large number of applications from abroad.

1.2 Objectives of the research programmes.

The major part of the research is currently funded by the Swedish Board for Technical Development, STU. ~~The previous five-year programme for knowledge development in information processing 1980-85 and the national programme for microelectronics have been of decisive importance for the build-up of research in the department~~. The funding ~~thus~~ emphasizes not only basic research, competence build-up, and development of excellency in selected areas, but also that results from the work should be transferred to applications in industry, commercial users of computer systems, public administration, or in other areas of research.

These goals are sometimes competing or contradictory. We have tried to balance our efforts so that the different goals would be achieved reasonably well. The current state at the department combines a broad coverage of different specialities in computer science, which provide the basis for high-quality undergraduate and graduate education programmes, with critical-size groups in several central areas, such as AI, expert systems, programming environments, digital systems design, and theoretical computer science.

Interaction with industry is promoted and taken care of in several ways, with the intension to utilize our competence as efficiently as possible and to maximize the mutual outcome, without becoming dependent upon industry funds for financing of research. Our *knowledge transfer program*, which started last year, has become a success and provides an ideal framework for people from industry working together with the department's personnel.

Still the most important success criterion is the quality of research and the results produced. Important measures are the number of PhDs (and licentiates) produced, the number (and quality) of scientific papers published in international journals and conference proceedings, and the level of the interaction with the international research community, as measured by e.g. the interest for cooperation, visits, exchange of results, etc. ~~We feel the research at the department has reached a size and quality, which gives it a high international visibility. The following chapters, describing activities in the individual research groups, and the appendices listing publications, etc., should substantiate this claim.~~ The purpose of the present Annual Research Report is to show to what extent we have achieved those goals, qualitatively and quantitatively.

1.3 Build-up of Research Areas.

The research proposal in 1979 for the STU supported knowledge development programme proposed research in two areas: *programming environments* and *application systems*, the latter including the study of application modelling, dialogue systems, and office applications. Additional proposals were made later for a *laboratory for artificial intelligence*, with an emphasis on expert systems and for an *LSI Design Center* in cooperation with the Physics department, one part of which has evolved into the current CADLAB.

Looking in retrospect, we observe that these were fortunate selections, and that all these named areas are considerably more 'popular' now than five years ago. From the reports for the respective laboratories, we can also see that their initial plans have provided useful guidance.

But as could be expected, we have also seen additional research areas during this period that also were well worth covering, and which connect organically to the existing areas of study. To some extent those new fields have been assimilated within the existing laboratory structure:

- *formal specification methods* in the programming environments laboratory;

- *statistical information systems* in the application systems laboratory (Bo Sundgren).

In other cases the connections have been established by organizational means:

- in 1983, the new Department of computer and information science (IDA) was formed from the previous datalogi, telesystem, and ADP (Administrative Data Processing) groups. This has led to rapidly increasing interactions between the C.A.D. laboratory, where computer architecture and VLSI design techniques are studied, and the other groups;

- also in 1983, we joined the inter-Nordic SYDPOL (System Development environment and Profession-Oriented Languages) project, in cooperation with researchers from the universities of Oslo, Aarhus, and Stockholm in the areas of user interactions, and the effects of information systems on user milieu;

- since 1985, we participate in a European cooperation on *AI and Pattern Recognition* under COST.

- during 1985 a planning and initial build-up of joint activities in the robotics area has been carried out together with the department of mechanical engineering and the department of physics.

Finally, new areas of activities and new groups have been formed, under the leadership of arriving researchers:

- *logic programming* and *attribute grammars*, started by Jan Komorowski and continued with additional breadth by Jan Maluszynski, later joined by Wlodzimierz Drabent;

- *geometrical complexity*, started by Andrzej Lingas, later joined by Christos Levcopoulos and Rolf Karlsson;

- *library information science*, in the laboratory started a few years ago by Roland Hjerppe.

- *administrative data processing*, where Göran Goldkuhl and Annie Röstlinger, previously in Stockholm and Göteborg, will restart research activities.

We also experience a split-up of existing laboratories, when subareas grow to a critical size or when subgroups from several labs are brought together to form a new productive constellation:

- *representation of knowledge in logic*, where parts of ASLAB together with some persons from AILAB formed the new RKLLAB under Erik Sandewall.

- *natural language processing*, where we expect that one part of the AILAB will form a new laboratory during 1986 (Ahrenberg, Wirén).

Through all of these means, we now have a research environment with considerable diversity, where at the same time the existence of a joint department and a laboratory structure provides the cohesion or 'glue'. In this milieu, the graduate students are exposed to a multitude of research specialities, so that they can make an informed decision about which area to choose for themselves, and where interactions between specialities is an everyday reality. Appendix B presents a list of the available advisors for the graduate students in our department, and their background and present research interests.

One significant aspect of the departmental build-up is that we are now able to offer a comprehensive set of courses for graduate study. Both the necessary requirements for such courses (teachers, and students) are present now to a much larger extent than five years ago.Appendix B shows the courses that are offered during the present academic year.

Another significant aspect of the department is that the base of computer equipment has been strengthened, through the addition of a large number of powerful workstations (such as Xerox Lisp-machines and SUNs) on an Ethernet, as well as by the gradual extension of our previous DEC-oriented system (DEC-20, PDP-11:s, DECNET). The existence and reliable operation of this base has been significant, not only for our own work, but also for the knowledge transfer. See also appendix D.

These things, taken together, represent the results of our efforts to build up a viable research milieu and at the same time provide the basis for high-quality

undergradate and graduate study programmes in computer science.

1.4 Knowledge Transfer Activities

A main task for a research organization is to serve as a source of competence, bringing together and distributing not only its own results but also to import and collect state-of-the-art information from the international research community. A great deal of attention is paid at our department to the issue of organizing effective knowledge transfer procedures for the benefit of recipients outside the university.

1.4.1 Improvements in Undergraduate and Masters-level Teaching

In the long range, the most significant method for knowledge transfer is through undergraduate and masters-level education. The development of our research programme has contributed to that education in several ways.

Firstly, a new computer science 'line' (*datavetenskapliga linjen*) was started in 1982 in addition to the computer science and engineering curriculum (*datateknik-linjen*). This new line is in the school of engineering, but differs from ordinary engineering curriculums (such as electrical engineering, or mechanical engineering) in some significant ways:

= significantly more discrete mathematics, partly gained by reduction of the calculus courses

= courses in theoretical branches of computer science

= courses in AI and AI-oriented subjects

= Lisp as the first programming language

= relevant humanities, such as psychology and linguistics, are significant parts of the curriculum.

The first set of students from this curriculum are now in their fourth and last year. It is already quite clear that these students develop a different 'culture', and in particular a more solid basis for graduate research in computer science, than what students in our other lines do. While certainly our other lines will continue to be of very high importance, the computer science line has provided a significant addition.

Secondly, the set of courses that are available in the other lines has been extended, and many of the courses have been improved. Technically, this has often been done by making new courses from the computer science line

available to other lines as well, but it is the STU funded research that has provided the competence base for the new courses. In the computer science and engineering line, a specialization for telematics has been added, relying partly on our research in interactive systems and office systems.

Thirdly, the mechanical engineering line has been extended with a new specialization that combines mechanical and computer engineering. We believe that especially the STU-funded research in artificial intelligence will be significant within that specialization.

Details about these curriculums and the set of courses there are given in appendix C.

1.4.2 The Knowledge Transfer Program

Knowledge transfer via undergraduate education is efficient in the long run, but slow to take effect. We have instituted a *knowledge transfer program*, KTP, together with a limited number of industries:

Alfa-Laval
ASEA
Ericsson
S-E-Banken

(a few more may be added and there are presently several candidates wanting to start in the near future). The goal of KTP is to 'inject' competence derived from research into the existing industrial organization. The method is that at least one person, located on a middle level in the organization, comes to our department for a period of one or a few years, in order to learn new technology, and returns to his organization after that time. The participating company also pays a yearly contribution that helps pay for researchers (particularly guest lecturers) and equipment.

More details about KTP are given in appendix A.

1.4.3 Spinoff Companies

The significance of university spinoff companies for industrial growth is well known. One part of our artificial intelligence laboratory, lead by Uwe Hein, split off in the spring of 1984 and formed Epitec AB. The new company has presently 15 employees. The main effort goes into development of a commercial product for building expert systems based on experiences from the AILAB research. The company is also engaged in consulting and is presently assisting several Swedish companies in the development of knowledge-based systems.

A few years ago, Jerker Wilander and Kenth Ericson founded the company Softlab AB in Linköping. Softlab is working in the area of compiler design, and they have developed the front end for the PLEX compiler now used at LM

Figure 1.1. *Undergraduate project courses are inspired by KTP companies.*

Ericsson. The company is growing steadily and is also expanding its scope of applications.

Recently Grafitec AB has been founded by Michael Pääbo and others from the CADLAB group. Grafitec will be active in business graphics and, later on, in scene animation. Another earlier spinoff from CADLAB was DIGSIM.

Some other spinoff companies in Linköping have required a considerable number of software specialists, although their main business is something else. In particular, Context Vision (formed in 1983, for building picture processing systems) has recruited heavily from our department. The intensive communication with the many developing high tech software companies around the university is a vitalizing force for the department.

1.5 Environment

During 1985 the department moved into a new building (E), connected to the old building B. This new building is very well adapted to the needs of the research and teaching groups and provides a very inspiring environment, which promotes informal communicatation and cooperation. Unfortunately the building was too small from the beginning. Although one third of the department is left in the old buidling, we already have to rebuild some of the new laboratories into office space.

Figure 1.2. *Communication area in the E building.*

2.

PELAB
The Programming Environments Laboratory

Bengt Lennartsson

The concept *Programming Environment* is recognized and used in the research community. For some years it has appeared in titles of conferences, workshops, textbooks, and research reports. There is an increasing number of researchers sailing under this flag, meeting to exchange ideas and experiences. In software industry, on the other hand, the problems, methods, and tools are grouped differently. Software Engineering Environment, life cycle support, configuration management, *etc.*, are terms frequently used. The grouping of problems and methods in industry does not match the grouping of ideas and results in the research community. We will here elaborate on the different views, their backgrounds and their meanings. After that follows a presentation of activities in PELAB.

2.1 Programming Environments as a Research Area

The background of programming environment research is the software development situation in the sixties. There were several kinds of reactions to the support systems of that time:

* They were complex and impossible to grasp. UNIX is a typical reaction to that. The success of UNIX is to a high degree due to the simplicity in its design.

* They were hindering rather than supporting the user. INTERLISP is a typical reaction of this kind. The spirit of the INTERLISP development was to investigate and demonstrate how supportive a system could be. It was quite clear from the beginning that power was the dominating interest. Simplicity was not. The functionality of INTERLISP has been a source of inspiration for almost all later

---

The work in PELAB is mainly supported by STU, The Swedish Board for Technical Development.

programming environment projects.

* The old systems were implemented ad hoc and not based on a formal specification or on formal concepts. MENTOR (INRIA), The Synthesizer Generator (Cornell), and The Programming System Generator (Darmstadt) are typical projects aiming at using existing results from automata theory, formal languages, *etc.*, as a basis for the development of support systems.

All programming environment research projects have been more or less implementation oriented, and most fit into one or more of the three categories listed above. SMALLTALK is an example of system with a different origin. It was from the beginning designed for naive users: very young children. In spite of that, it has inspired specialists and researchers in programming environments in general. The use of high resolution graphics combined with a pointing device, its full integration of program, support system, and screen objects, have been and will continue to be important inputs to research as well as to industry.

2.1.1 The Different Schools of Computer Science

Among the researchers in computer science there is no general agreement on methods or central problems. There are at least two approaches. We temporarily assign the labels *the formal school* and *the explorative school* to the two approaches.

The Formal School

Computer science is based on logic and mathematics. The central problem is: *how to transform formal abstract specifications to executable programs.* If this transformation is semantics preserving, then the program is correct. Important subareas are: notations and their semantics, formal methods, correctness proving, and general formal theoretical foundations.

The Explorative School

Computer science is an experimental discipline. Computer systems are normally used by humans. Their needs and expectations are not available as formal specifications. The user and the using organization will change their behavior when a new system is introduced. There is a dynamic mutual influence between the system and its users. The central problem is: *how to develop executable programs that matches the users concrete and changing needs.* Computer science is influenced by linguistics, psychology, sociology, and application domain knowledge, as well as by logic and mathematics.

Unfortunately the communication channel between the schools has a very low bandwidth. The two sides do not in general appreciate each others methods or results. The discipline computer science is unique in this respect as evidenced

by a study of statements from reviewers of research proposals to National Science Foundation, USA.

One early use of the computer by scientists was as a calculator. They had analytical formulas and, given some input data, the program terminated after producing the corresponding set of output data. The analytical formulas served as a specification of the program. The formal view of computer science is highly relevant for this domain. In the explorative school interactive systems were considered. The input and output data were interleaved in a dialogue. Sometimes the user program "never" terminates. In some systems the current state is saved after each session, and this state is then resumed when next session starts.

The location of the programming environment area is to some extent in the battlefield between the two forces. The aim of many of the research project is to give support for experimental, or explorative, programming. That is, to make updates easy and cheap, to support updates of executing programs, and to accept considerable software development in spite of lacking formal specification of program under development. The software designer learns about the application area and about the user. Software design includes the selection of functions to be implemented. On the other hand, when the explorative phase is over, that is, when there is a stable match between the existing program and the needs and expectations of the users, then the system should support activities recognized by the formal school. That is, activities like extraction of abstract functional specification from the existing program and support for transformation of this specification to an efficient and maintainable executable program.

There are two reasons why the research area *Programming Environments* has one foot in the explorative school.
Firstly, by tradition most systems discussed and developed in the area have aimed at support for explorative programming. The design activity in general maps better to the non-terminating and information saving view, than to the terminating output producing program.
Secondly, the research area itself has to be *explored.* What kind of functionality would be appreciated by users of programming environments, how should the user interface be designed, *etc.*

During the last years, however, the area has matured, and today there is sufficient experience gathered to allow formal specification of the functionality of programming environments. The trend is to rely on results and methods from *the formal school* for the development of "environment generators". Thus the second link between programming environments research and *the explorative school* is weaker now than in the past. Instead, the link to *the formal school* is becoming more important.

The second link between programming environments research and explorative programming remains, however. Keywords as incrementality, support for updates and re-use of information, appear in abstracts from most current

papers on programming environments.

To summarize, the research area *programming environments* has an origin in the explorative school. Programming environments themselves are today often implemented according to ideas in *the formal school*, but they are aiming at support for software development also in the explorative tradition.

2.1.2 Programming Environments and Software Industry

The software crises is a well known phenomenon. Deadlines are broken and cost estimates fail. Programs are erroneous. Different organizations try different solutions. More people, new methods, new languages, new tools, new management, *etc.*

The current view in industry of the software development process is normally according to the life cycle model, where work and responsibility are divided into pieces, phases. The output of one phase (say module specification) is the input for the phase to follow (module design and coding). Backtracking is very expensive (the waterfall analogy). Many companies are developing, or looking for, a complete phase oriented tool set, a Software Engineering Environment.

The life cycle model is not consistent with the view of software development in the programming environment research community. Explorative programming considers requirement analysis, specification, coding, debugging, testing, and field test to be integrated and interleaved activities. In a large scale industrial project where hundreds of thousands, or millions, lines of code shall be developed, a strict distribution of work and responsibility is necessary. A large number of people will be involved. Persons will come and go. Such a project can't follow the explorative tradition. On the other hand, it can be assumed that the major parts of the application area are well known for such a project. Most functions can be specified correctly in advance and implemented top-down. However, certainly some of the functions have to be "explored". Bugs will occur and demand correction. During the lifetime of the system the requirements will change. The system will grow. Even if it is necessary to base the organization of the development work on a formal basis, support for updates, undoing, *etc.*, in the explorative tradition is required.

A full conversion in industry to the explorative style would require reorganization of current activities. Such a reorganization of the work would make some of the existing problems disappear and also introduce some new ones. Very few organizations are willing to take such a risk, in particular in a critical situation. It is also difficult for anybody involved to understand the consequences, as the programming environments obviously are lacking support for some of the existing problems and giving support for others, not existing.

A very strong conservative force is the back-ward compatibility. Very few users and organizations start from scratch. They want to improve or enhance current systems, methods, or procedures. Use and maintenance of existing systems

dominate over development of new ones. At each separate decision point it is assumed to be cheaper to add functions to an old system, to add people to an existing organization, to buy yet another computer from the old vendor, *etc.*, than to restart from scratch. Also the end users, and everybody else involved, are conservative. They may have several years of experience of the existing situation. Their skills, their value on the market, may consist of knowing how to handle tricky tools and systems. A situation, where the problems they are the experts to solve disappear, may not attract them.

One main idea in programming environment research has been, that at the time the results will be exploited, hardware will be much cheaper than when the research is done. Thus most implementation have been quite CPU consuming. As much work as possible has been transferred from the user, the software specialist, to the system. In many research projects it has been assumed that the user, the software specialist, has a powerful personal workstation on the desk. Today such equipment is available at a reasonable price. Only one company, however, has a long time experience of the use of personal workstations and powerful programming environments in a large scale, Xerox Corporation.

At Xerox PARC powerful personal workstations have been in use since the middle of the seventies. The support systems like Interlisp-D and the MESA Environment, later Xerox Development Environment, have required computers like the Dolphin or the Dorado. Today computers of the same power and functionality are available on the market, at the same time as the bottleneck is number of available software specialist.

An alternative to consider for those suffering from the software crises is to learn from the experience at Xerox. There, powerful programming environments have been used for several years in full scale product development projects. We expect personal workstations having the power of at least the Dorado, to be on every software specialist's desk within very few years. At least in companies able to recruit software specialists.

Another difference between the views in industry and in programming environment research is about the educational level and the skills of the user. In the programming environment research community it has often been assumed that the user has a grade or degree in computer science. In industry, the software designer often is an application domain specialist with some rudimentary training in programming, or at least in one or more programming languages. Just as in computer aided mechanical or electronics design, support systems can't create good designers. They can just improve creativity and productivity for those who are skilled specialists already.

2.1.3 Future Development of the Area

There are still several unexplored regions in the research area *programming environments*. So far, the target languages considered have been LISP dialects, SMALLTALK, and block structured languages in the Algol family. Powerful support systems for other types of languages have not yet been studied or developed.

Another sparsely explored area is knowledge-based facilities in programming environments. Expert systems, however, require the expert's knowledge and experience as the starting point. Most expert systems have been based on knowledge in specific application domains. The domain *software design* has, so far, been to large to be described in terms of a reasonable number of heuristic rules. For more narrow subareas, like configuration management, cost estimates, *etc.*, however, progress can be expected.

Most research projects in the past have considered the target computer to be a mono-processor and regarded the target program as monolingual. An important new area is programming environments for multilingual software running on distributed targets.

The steps from the idealized world in programming environment research to the real very large systems developed in industry have to be taken. The interpretation of *large system* is today a system with about one million lines of code. This figure increases roughly by an order of magnitude per decade. This enormous size is in itself an argument for support for updates and for incrementality. It is a challenge to the research community to study architectures for systems supporting the development and maintenance of very large software systems.

2.2 Activities in PELAB

The research in PELAB is a continuation of the work in program manipulation projects, that was done here during the seventies, and of the INTERLISP experience in general. PELAB was established as a separate research group in 1980. Since then four PhD dissertations and one licentiate thesis have been presented. The work has covered a large area, from partial evaluation and its application to Pattern Matching and to Specification of an Abstract Prolog Machine, to Compiler Writing Systems and Incremental Compilation. A very successful projekt, DICE (*Distributed Incremental Compiling Environment*), has been finished, and a new project, *Software Development Environment for Distributed Targets*, has just started.

2.2.1 The DICE Project

Peter Fritzson, Johnny Eckerland, et.al.

The DICE project is based on experience from previous PELAB projects, PATHCAL (*Jerker Wilander*) and Parser Writing System (*Kenth Ericson*). PATHCAL was a very early investigation of integration of command language, programming language, and debug language in an environment supporting incremental development and execution of Pascal programs..

In the DICE project (*Distributed Incremental Compiling Environment*) we have been aiming at the development of an appropriate architecture for a full scale integrated environment supporting the development of programs coded in block-structured languages.

A prototype of DICE has been implemented. The tools are running on a DEC-20, the host, where also all the information of the developed program is saved. The host is connected to a target, a PDP-11, where the developed program is executed. Among the results should be mentioned that
the flexibility normally available in an interpreting system could be achieved in a compiling system also, and that
the functionality of a high level target debugger can be obtained via the incremental compiler without any target code instrumentation, and without the existence of a target debugger at all.

The architecture of DICE has been developed under several constraints. The system should be able to operate on compiled code and the developed program should be kept separate from the development environment. Some of the more important points of the DICE system are:

- *Remote debugging and maintenance* is easy to achieve with the DICE system configuration.

- An incrementally compiling system like DICE which has a program data base is especially suitable for the *development of big programs*, on the order of 20 000 to 100 000 lines of code. Compiled code gives fast execution and incrementality gives fast program update and powerful debugging facilities.

- *Separability* - the compiled program is separated from the source code so that it can execute outside of the program development system.

- *Connectivity* - the DICE system can be connected to a malfunctioning production program in order to debug it or to correct it. This can be done after the error has occurred and need not be planned in advance.

The results from the DICE project are currently used in the development of the next generation of environment at SUN Micro Systems. Peter Fritzson, the DICE architect, is on leave at SUN for one year and a half.

2.2.2 Structure-Oriented Text Editor for Large Programs

Ola Strömfors

The ED3 editor designed and implemented by Ola Strömfors has been in use for structured documents in general for several years. It has recently been used as a template for a powerful language oriented editor-prettyprinter-syntax checker. The language oriented features are available for Ada and Pascal.

Most program editing today is done with text editors. A compiler is then used for both syntax checking and code generation. This means that detection and correction of syntax errors take long time, especially for large programs. Some text editors, *e.g.* EMACS , have modes for different programming languages. Commands to insert templates for different programming language constructs will save typing and avoid misspelled keywords. When the program text is modified only limited syntax checks, such as parentheses matching, are performed. For a language with simple syntax, e.g. Lisp, this perhaps can be enough.

Many users find it difficult to handle large programs (or documents) with a text editor. The reason is that most text editors only support a flat sequence of characters or lines, and not the subprogram structure of a program (or the paragraph and chapter structure of a document).

Some program editors are based on a parse tree representation of the program. The editing commands interact with the parse tree and guarantees that no syntax errors are introduced in the program. The ED3 editor is a bit different. The tree structure is *not* used to build parse trees. Instead the user is free to build any tree he wants. Often the structure follows the subprogram structure. But the user often wants to divide a long sequence of procedures on the same level into different groups. As *head* of each tree node he can put a comment describing this group of procedures. This is useful for programming languages without nested procedure declarations, such as *C* or even assembly languages. Different nodes can even be written in different programming languages.

The following example will show how ED3 can be used to enter a program and how syntax errors are removed interactively.

- The user enters program text as usual, but does not care about indentation or new lines. The user then hits the *syntax check and pretty-print* key.

- When a syntax error is found, the cursor will be placed just before the first erroneous token. The error message will contain a list of legal tokens at this position. The user then perhaps enters one of the alternatives and hits the *syntax* key again. The text is parsed again from the beginning (of the current text node).

- If no syntax errors are found, the editor replaces the text with a *pretty-printed* copy. If the user does not like the result, he just has to hit the *undo* key. He can also redefine the *syntax* key to just do syntax check and no *pretty-printing.*

- If the entered text contains more than one procedure, the user normally will hit the *split* key, so that each text node contains only one procedure.

The user has complete freedom how to cut his program (or document) into text nodes and how to build a tree of them. ED3 just supplies a default. Dividing the program into text nodes, with for instance a procedure in each, gives the user a good overview of his program. The tree structure given by the author may also help other people to understand the program.

The user will also feel more safe editing one procedure at a time. Replace and delete commands then just affects the current node. Every time the text editor is entered, a copy is taken of the current node. This makes it possible for the user to get back the old version or to compare them.

The node structure also acts as well defined starting points for syntax check (parsing) and *pretty-printing.* The speed of the parser and *pretty-printer* (between 10 000 and 20 000 lines/minute) will make it possible for the user to have text nodes with several hundred lines if he wants to. A syntax check will still take only about one second to perform.

The structure editor in ED3 handles tree structures, which can be thought of as hierarchical directories of nodes of different types. The structure editor has commands to walk around in the tree, automatically displaying the current node and its subnodes, and commands to copy, delete and move subtrees or individual nodes.

ED3 must also contain an editor for each node type, like the text editor for the text nodes and the graphics editor for picture nodes.

A syntax-directed editor could also be integrated to handle program nodes, which can be parse trees for anything from a small program fragment to a complete program. A procedure would be the most common choice as it is today for text nodes.

The parser can transform a text node to a program node when the *syntax-directed* editor is entered, and the *pretty-printer* could transform a program node to a text node when the text editor is called (and when the program is sent to a traditional compiler). The user will then have a free choice between text editing and *syntax-directed* editing.

Both the text and parse tree representations could be kept until one of them is modified to reduce the need for reparsing or *pretty-printing.*

Since the parse tree representation is space intensive, scanning and parsing is time consuming and almost all compilers today only accepts program text as input, a large program could be stored as tree of text nodes and only one node at the time is parsed when the *syntax-directed* editor is entered.

The parse tree representation will be chosen as the main representation as soon as the compilers will accept it as input.

2.2.3 Static and Dynamic Program Flow Analysis

Mariam Kamkar, Nahid Shahmehri

There is a growing attention paid to the reuse of already existing information in programming environments. An important piece of information, for the user, is indication of the consequences of an update. In the debugging situation it is important to know whether it is meaningful to continue execution after a change to a variable, or, in case of an incremental system, after a change in the program. Such information is also important in the maintenance phase. Cross reference information, or indications of dependencies in general, should be interactively available when the user in editing mode is considering a change in a program module.

The purpose of this project is to study methods and to develop algorithms and interactive tools which can give the user the information required when needed. The system should be able to answer queries such as:

* Which procedures/functions call a named procedure/function
* Which objects are declared/referenced/modified by a named procedure/function
* Where can the value of a named variable affect the value of another named variable

Such a tool is usually absent in programming environments designed for block-structured languages.

The approach in the project is to specify language semantics in terms of an attribute grammar and to store local information as relations for each procedure. This approach enables incremental update of dependencies.

In a future step dynamic information, which depends on the current state of an executing program, will be combined with static information. Such a combination will, for instance in debugging and testing situations, sharpen the answers given by the system to queries like:

* Where may a variable have been assigned its current value

* Where (and if) will the current value of a variable be used later on in execution

2.2.4 Requirements for Version Control

Kristina Ernstsson

During its lifetime, software will exist in many versions. The reasons for this can be variations in application demands, variations in hardware configurations and the opportunity to improve a program, but still let the old versions exist due to experimental improvements of algorithms. The differences between all these versions are unavoidable and purposeful.

There are many approaches to the version control problem and there are great differences between the systems implemented. The requirements vary with the different desires that programmers, project managers, documentation writers, and maintenance people have.

With the concept of *VERSION CONTROL*, we mean methods or models to

- store different versions of an object and the relations between the versions

- give information about each version stored and the history of the versions

- retrieve some specific version

- give support in producing and maintaining versions.

We will use the term *VERSION* as a general term for the objects we wish to control. The versions are often described as being of two different kinds.

The first kind of versions is the result from the improvements of a program in successive editing sessions, *i.e.* versions changing with time. We will call this type of versions for *REVISIONS*. The term successors is also used in some papers.

The second kind of versions results from the adaption of a program to a specific use, for example for different hardware configurations or for different application demands. Here we will call these adaptions of a program, for different *VARIANTS* of a program.

In a program database we store different kinds of information about a program. The program can simultaneously exist in various shapes, as source

code, as machine code, *etc.* There are different *representations* of a program. We have also *descriptions* of the program, like cross reference information. Therefore, it is important to augment the domain of version control normally defined, to control representations and descriptions or, as we will call it, *ASSOCIATED INFORMATION.*

In order to review existing systems and ideas about version control, we have tried to make a classification of the different requirements that a user can have on a tool for version control.

* The most important requirement is to store different kinds of information about the versions. To mention some of them: the version history must be easy to access, the relations between variants and revisions must be stored in a convenient way, creation and access to the associated information must be available for the user or the system in order to make different views of the program.

Of course the need for a sophisticated way of identifying the different versions is also an important requirement.

With this basis for a version control tool, the following classification of the user oriented requirements can be done:

* supporting the technique of modularization, assisting when merging and splitting versions, and when configuring a software system

* supporting for reuse and change of old versions, and also for modifying the existing version structure

* support in assessing all the impacts of a modification to a specific version, and advice on what has to be done

* control and record authorizations to a specific version (A similar problem to solve is simultaneous access of versions, both important requirements in a software project)

The first step in this project is a review of existing systems and ideas on version control.

2.3 Next Research Area for PELAB

Lars Strömberg, Yngve Larsson, Bengt Lennartsson, et.al.

The area for PELAB research for the next period will be *Programming Support Environments for Distributed Targets.* Software for distributed targets is a topic highly relevant for Swedish industry. Critical applications could be found in areas as industrial robots, process control, telecommunications, defense systems, and for system software for office applications.

We have chosen this new profile for PELAB for three reasons. Besides its relevance for Swedish industry, it raises a number of scientifically challenging problems, and we have a very suitable background to attack this area. It is a natural extension of the activities in the past. Taking the step, however, will be far from trivial.

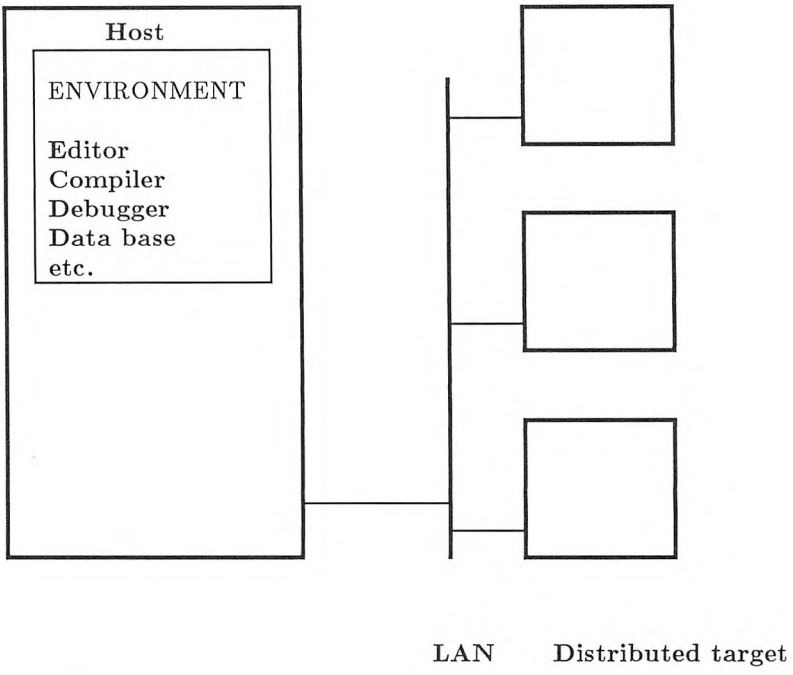


Figure 2.1. *Support system for distributed target.*

Among the specific problems to study are:

* How to extend the powerful incremental architecture developed in the DICE project to handle the much more complex situation with distributed targets.
* How to monitor, debug, and test concurrent programs in general.
* How to design an appropriate user interface for communication with an executing concurrent system.
* How should the communication links between the host and the distributed target be designed and used.
* How to compose a tool set supporting the code management for a distributed target possibly consisting of different types of processors.

The goal is not to give exhaustive contributions to every problem listed above, but rather to focus on the architecture of the supporting host system and on its communication with the target and the target processes. The research will be based upon existing modern hardware (powerful work stations for the host, high speed local area networks for the links, *etc.*). We will also use existing programming languages and notations, for example, CCS, Edison, PASCAL, MESA, or Ada.

2.3.1 Our Approach

We will use the same approach for the new area as we have had for the DICE project. In DICE we have developed an architecture for an environment supporting development of software coded in a sequential language for a mono-processor target. The basic idea in DICE is to define an abstract machine maintained by the host system. The environment gives the user the impression of an interpreting system, but the actual execution takes place in a connected target with program compiled to machine code. The environment maps the abstract machine, with concepts on the level of the supported language, onto the target hardware and vice versa.

There are several advantages with this architecture. All hardware and software below the supported language level are invisible and irrelevant for the programmer. The user has the interactivity and incrementality normally available in interpretive systems only, without having to pay by slow execution. The abstract machine is a distinct and clean interface between the user-oriented tools and the actual target hardware. The architecture enables a very high degree of both rehostability and retargetability.

The abstract machine is also a proper interface between different thesis projects. Some of the problems listed in the preceding paragraph are about communication between the user and the abstract machine. The rest of the problems only have to do with the mappings between the abstract machine and the distributed target.

2.4 PELAB Personnel

Bengt Lennartsson, Tekn. Dr, lab. leader
Gunilla Lingenhult, secretary

Supervisors:

Pär Emanuelson, Fil. Dr	With EPITEC AB from July 1985
Peter Fritzson, Tekn. Dr	At SUN March 1985 - July 1986
Anders Haraldsson, Fil. Dr	

Employed graduate students:
Rober Bilos, Civ. Ing.
Kristina Ernstsson, Civ. Ing.
Mariam Kamkar, BSc
Yngve Larsson, Civ. Ing.
Nahid Shahmehri, BSc
Lars Strömberg, Civ. Ing.
Ola Strömfors, Civ. Ing.

Research engineer:

Ralf Nilsson	With Ericsson from August 1985

KTP participant:
Tom Rindborg, Ericsson Information Systems

Associated persons:
Johnny Eckerland, Tekn. Lic., EPITEC AB
Kenth Ericson, Fil. Kand., SOFTLAB AB
Pär Emanuelson, Fil. Dr, EPITEC AB
Jerker Wilander, Fil. Kand., SOFTLAB AB

2.5 Selected Publications

J. Eckerland, *Retargeting of an Incremental Code Generator*, Licentiate Thesis, Department of Computer and Information Science, Linköping University, Linköping, Sweden. Nov. 1984.

P. Fritzson, *A Systematic Approach to Advanced Debugging through Incremental Compilation*, Proc ACM SIGSOFT/SIGPLAN Software Engineering Symposium on High-Level Debugging, in SIGPLAN Notices. Vol. 18, No 8, Aug 1983.

P. Fritzson, *Adaptive Prettyprinting of Abstract Syntax applied to ADA and PASCAL*, LITH-IDA-R-83-08, Department of Computer and Information Science, Linköping University, Linköping, Sweden. Sept 1983.

P. Fritzson, *Symbolic Debugging through Incremental Compilation in an Integrated Environment.* Journal of Systems and Software 3, pp. 285-294, (1984)

P. Fritzson, *Towards a Distributed Programming Environment based on Incremental Compilation*, PhD Thesis, Department of Computer and Information Science, Linköping University, Linköping, Sweden. April 1984.

P. Fritzson, *The Architecture of an Incremental Programming Environment and Some Notions of Consistency*, Proceedings of the Workshop on Software Engineering Environments for Programming-in-the-large. Harwichport, Massachusetts. June 9-12, 1985. pp. 64-79.

O. Strömfors, L. Jonesjö, *The Implementation and Experiences of a Structure-Oriented Text Editor*, SIGPLAN/SIGOA Symposium on Text Manipulation, Portland, Oregon, June 8-10, 1981.

O. Strömfors, *Editing Large Programs Using a Structure-Oriented Text Editor.* To be presented at the *International Workshop on Advanced Programming Environments.* Trondheim, Norway. June 1986.

J. Wilander, *An Interactive Programming System for PASCAL*, BIT 20:2, (1980). pp. 163-174. Also in Interactive Programming Environments. *D.R. Barstow, H.E.Shrobe, E. Sandewall, eds.* McGraw-Hill Book Company. 1984. ISBN 0-07-003885-6.

3.

ASLAB
The Application Systems Laboratory

Sture Hägglund

The research program in the Applications Systems Laboratory (ASLAB) is oriented towards the study of theory, methods and tools for the development and maintenance of a non-trivial range of applications software with a significant increase in productivity, maintainability, understandability and user control. One important issue to be studied is how AI methodology and software techniques, in particular knowledge-based expert systems, can be integrated with more traditional information technology. Projects usually take an experimental approach and emphasize participation in application-oriented projects with industry and the public sector.

3.1 Projects and Researchers

This section summarizes the current achievements in the laboratory and lists the personnel currently active in ASLAB research.

3.1.1 Summary of research 1985

During the last year, the main efforts in the laboratory have been in the area of investigating the prospects of using knowledge-based techniques in application development. Our group has actively participated in the knowledge transfer program at IDA and we have worked together with several industries on applied projects.

Projects have been centered around the following issues:

The work in ASLAB is mainly supported by STU, The Swedish Board for Technical Development.

- exploration of the advantages connected with a software architecture where the control information needed for procedural execution of the program is separated from the domain knowledge expressing the rules and facts of the problem. In particular the possibilities to improve efficiency, focusing in problem solving and transparency in knowledge representation, through the introduction of domain-dependent control in the form of prototypes strategies based on typical cases from the application area[1]. Another aspect is the reuse of knowledge bases for other purposes than problem solving such as e.g. tutoring.

- Study of knowledge base migration from environments supporting knowledge acquisition and knowledge engineering into possibly diverse delivery environments, including requirements on interfaces to existing systems, such as e.g. databases. A practical case has been carried through in cooperation with the medical informatics department, where the Antibody Analysis Advisor (for giving expert advice on the intelligent selection of analysis techniques for identifiction or irregular antibodies in blood samples) has been migrated from the Lisp-based EMYCIN into the database system MUMPS[2].

- Continued study of software architecture of knowledge systems tools and of implementation techniques for rule-based inference engines, for instance in connection with the implementation of an EMYCIN-compatible system in MUMPS, to be used by the knowledge base migration project. We have also had undergraduate students working on experimental projects in this area: EMYCIN for a PC (Doherty) and Expert-Trees (similar to Expert-Ease) on a Lisp Machine (Moen). In this connection also short-comings of pure backward-chaining, rule-based systems have been investigated, starting from the experiences from a constraint-satisfaction configuration problem[3].

- Knowledge-based techniques and methodological issues in the area of statistical information systems (SIS). The SIS group, under the leadership of adj. professor Bo Sundgren, has conducted applied projects involving the development of a PC-based consultation system for statistical analysis, in particular the elimination of seasonal variations in data (Block). Another project has studied the need for more appropriate and precise methods for strategic interpretation of administrative business data (Wallgren and Wallgren). More theoretical issues are treated in the development of an algebra of statistical operators, to be used as an extension of relational algebra for databases[4] (Sundgren, Nilsson).

---

[1] **Nordin,** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching. LiTH licentiate thesis No 48, 1985. Condensed version to appear in *Proc of the 3rd Annual Conf. on Applications of Expert Systems*, Orlando, 1986.

[2] **Sandahl** et al., The Antibody Analysis Advisor and its Migration into a Production Environment. *Proc. of the 1st Int. Conf. on Expert Systems*, London 1985.

[3] **Rehmnert** et al., Knowledge Organization in an Expert System for Spot-Welding Robot Configuration. *Proc. of the 5th Int. Workshop on Expert Systems and their Applications*, Avignon, 1985.

[4] **Bo Sundgren,** Outline of an algebra of base operators for production of statistics, ASLAB Memo 85-04.

More details on these and other activities within the laboratory are given in a later section.

3.1.2 Personnel, ASLAB, spring 1986.

The following list presents persons currently active in ASLAB.

Project leadership/thesis supervision:

Sture Hägglund, PhD, Lab leader
Gunilla Lingenhult, secr.

Kevin Ryan, PhD, guest researcher 1985-86
Bo Sundgren, PhD, adj. professor

Graduate students:

Shamsul Chowdhury, BSc, MSc
Tim Hansen, MSc (starting spring 1986)
Henrik Nordin, Tekn. Lic. (at CMU 1985-86)
Ivan Rankin, MA (starting spring 1986)
Roland Rehmnert, MSc
Kristian Sandahl, MSc
Pål Sørgaard, MSc (visiting from Århus 1986-88)

Associated persons:

Lars Bengtsson, S-E-Banken AB, Stockholm
Hans Block, SCB, Stockholm
Patrick Doherty, undergraduate student
Ove Hanebring, Alfa-Laval Automation, Lund
Christer Hansson, undergraduate student
Christian Krysander, lecturer
Sven Moen, undergraduate student
Gösta Nilsson, Högskolan i Örebro
Pablo Lozan-Villegas, ASEA, Västerås
Lars Reshagen, Dept of medical informatics
Börje Rosengren, Alfa-Laval Automation, Lund
Nosrath Shasavar, undergraduate student
Toomas Timpka, Dept of medical informatics
Anders Wallgren, Dept of math/statistics
Britt Wallgren, Dept of math/statistics

3.2 Background.

There are strong indications that the next generation of computer and software systems will support "knowledge-based" approaches, i.e. systems containing a body of generalized facts and rules which may form the basis for dynamically adaptable problem solving mechanisms. In the near future we can expect

exciting break-throughs for new approaches to information processing. Hardware development (e.g. the japanese and other efforts on the next generation computer systems), availability of inexpensive workstations supporting broad-band human-computer interaction with voice I/O and high-resolution color graphics, and AI-inspired software technology should provide the basis for utilizing the computer as a powerful tool in many new application areas.

A fundamental issue in this development is the software environment supplied for the life-cycle support of information processing systems. We believe that the ability to create this environment will depend on an integration of experiences from many sources of knowledge. Important contributions will come from areas such as formal specification techniques, programming systems, database management and artificial intelligence, as well as from application-oriented research on personal computing environments and office information systems.

Contributions to this development from research in our laboratory can be expected regarding those aspects of software architecture which have to do with the combination of the formalized structures of traditional data processing with the management of unstructured informations in office systems. In particular we pursue a deeper understanding of how program systems can be created directly by users who are experts in their own application domain, rather than specialists in the efficient organization of computations and storage structures.

The importance of this development is reflected by the massive current interest in tool systems for application development captured in such buzzwords as "*application generators*" and "*forth generation software*". Considerable increase in software productivity has been demonstrated for tools which eliminate the need to write application-independent parts in a general-purpose language. We believe that this is an important trend which has to be matched by basic research on the foundations for application modelling and automated generation of software. In addition we believe that the next generation of application development systems will incorporate important ideas from artificial intelligence research on knowledge-based systems, and that substantial contributions to this development can be made by research in our laboratory. There also seem to be a unanimous agreement on the fact that knowledge represented in the computer, in particular domain knowledge, is the key to successful realisation of "intelligent" application systems.

3.3 Overview of current research activities.

Work in the laboratory is organized in two subgroups, one for *Knowledge-based applications software environments*, led by Sture Hägglund and one for *Statistical information systems and database technology*, led by Bo Sundgren.

3.3.1 Knowledge-Based Application Software Environments.

(*Hägglund, Nordin, Rehmnert, Sandahl*, et al.)

This work is oriented towards the application of knowledge-based techniques for software development, in particular the integration of methodology for developing expert systems with more conventional information technology such as database management and office information systems. In this process we emphasize the potential benefits of AI methods for producing more useful, easy-to-change and understandable software, rather than as a way to solve computationally difficult problems, but also as a way to introduce the following qualities in the software development, maintenance and use:

1. Interactive support for application modelling, through the use of AI-inspired representation techniques which allow incremental modification and maintenance knowledge stored in the system.
2. Advanced dialogue management, including (restricted) natural language explanations, queries and result presentations.
3. Learning support, based on the fact that information inside the system may be inspectable and also reusable for teaching purposes.
4. More maintainable systems, since the distance between what is stated by the domain expert and what is entered into the system can be shorter than in conventional programming.
5. Less rigid tools for application development than today's fourth generation languages, which can not represent and utilize non-trivial domain knowledge.

We assume an architecture where the final system should run not only inside the development environment but also on e.g. a personal computer or as a part of a corporate database system. This implies that we have to study how a core system can be realized in different environments and how the application dependent part can be "compiled" or migrated (automatically or through a manually supervised transformation process) to the target system. Several alternatives should be contemplated for the migration process in addition to moving to an equivalent (Lisp) system, including such possibilities as generating compilable programs in a suitable language or some kind of decision tables to be interpreted. The purpose of migration may be:

- to promote target system independence,
- to interface to existing software,
- to improve economy,
- to increase efficiency,
- to hide development features, or
- to make the system more robust.

We will approach this problem by designing a development, or rather a knowledge acquisition, environment for the Interlisp-D workstation, probably using some existing software for building knowledge systems as the core of the

system. Each component in this development system will be designed under the constraint that a counterpart, when appropriate, should be possible to realize in the target environment. Part of the project is to define reasonable target environments, with interfacing and integration with database management as central topics. In fact, we expect to start with an investigation of features, which are reasonable for a target system and then proceed to design to corresponding development support needed in the knowledge acquisition environment.

In order to further limit the task and make the problem area manageable, we intend to restrict ourselves to consultation systems, i.e. systems providing advice or decision support. In particular we will study initial-advice systems, where the system supports a non-expert user in handling routine or almost-routine cases while more exceptional cases to be forwarded to a human expert are recognized as such. This basic pattern reappears in several expert-systems projects, where we are engaged in external cooperation, which makes it very promising as a study object. These applications are:

a) sales support in process industry,
b) financial and legal advice in banking and
c) medical treatment in primary health care.

In these areas we are, or has recently been, involved in projects, where the main body of the application-oriented work is undertaken by persons from industry or other departments. Our style of research is thus heavily oriented towards participation in joint application-oriented projects with external parties. We are however sincerely conscious about the importance of emphasizing application-independent methodological issues in such work and do not engage ourselves in projects which do not contribute to the advancement of the research goals stated above.

Using typical cases for knowledge-based consultation and teaching.

This project started from the experiences in a small-scale project conducted within the knowledge transfer program with the purpose to demonstrate how an expert system for advising on legal and economical issues in a bank's back office could be built (Bengtsson). In that project it was clearly demonstrated that a pure rule-based backward-chaining system (in this case EMYCIN) does not provide an effective model of problem solving in the application domain. One main drawback concerns the need for non-monotonic reasoning, i.e. the need to allow new evidence to invalidate previous assumptions and conclusions. (For instance, when a client learns about the tax effects of a suggested transaction, he might change his mind on previously given information.) Another obstacle is the difficulty to separate control information from domain knowledge in the rule base, since pure rule-based systems often force you to extend the rules with control information for efficiency purposes.

Starting with the objective to improve reusability of knowledge, a new design of a system, POZZO, was made (Nordin). This generalized approach was based on the use of prototypes derived from typical cases, as expressed by a domain

expert. The typical cases and thus the prototypes represent the expert's experience, such as knowledge about strategies, i.e. how to work with the domain knowledge to obtain a certain goal. Since the implementation was made in a system supporting reasoned control of reasoning (WATSON by Jim Goodwin), the prototype strategies could also involve non-monotonic problem solving.

A major benefit with this approach is that it allows a more focused consultation and improved efficiency in the reasoning process while avoiding to pollute the domain knowledge base with irrelevant control information. The reusability of the domain knowledge was then convincingly demonstrated by a master's thesis project (Christer Hansson), where the knowledge base was reused as the basis for a teaching system. WATT[1].

The task for the WATT system was to generate a hypotetical case from information in the POZZO knowledge base, extended with some additional information defining reasonable ranges and constraints for different parameters, and to run a consultation where the user asks the questions. The student, who is assumed to be a person with some proficiency in the concerned area but not a specialist, is expected to ask the system for various pieces of information, which are relevant for reaching a conclusion and solve the hypotetical case. When the sequence of questions diverge from those generated by the concurrent internal reasoning guided by prototype strategies, the student is informed and entertained in a dialogue with the purpose to make sure that he knows and understands all the relevant information in the knowledge base.

The POZZO project was carried out by Henrik Nordin and reported in his licentitate thesis. During the fiscal year 1985-86 he is at Carnegie-Mellon University in Pittsburgh, working at the PRODIGY learning apprentice project under Jaime Carbonell. His interest in methodology for expressing and representing domain-dependent control information is important also for continued work in ASLAB on the design of a knowledge acquisition and development environment, as described above.

Expert systems tools and knowledge-base migration

A typical experience from successful implementations of expert systems, which are taken into productive use, is that there is often a distinct step where a satisfactory working prototype is reconfigurated or translated into the production version of the system. In general the very nature of the task to develop a knowledge-based system is so unstructured and ill-defined, that it is not surprising that a shift of technology is forced once the basic structure of the solution has been developed, and also that optimizations for regular production use should not be done at a premature stage.

In addition to the studies of systematic methods for migration, it is important to study how core functions of inference engines, or the equivalent, can be

[1]**Hansson**, WATT - A Knowledge-Based Case-Oriented Teaching System. Report LiTH-IDA-EX-86-01 and ASLAB Memo 86-01.

realized in different kinds of software environments, e.g. personal computers or mainframe database systems. As a first step in the direction of gaining a deeper insight into architectural issues involved in our approach, an experimental implementation of a portable version of the EMYCIN knowledge representation scheme and inference engine has been made (Rehmnert). The implementation follows quite closely the original system and has been used as the basis for the A^3 migration project described below, as well as for a PC-version of EMYCIN.

The Antibody Analysis Advisor, A^3, is a medical expert system developed for the purpose of providing guidance in the initial selection of analysis techniques for antibody identification in blood samples[1]. The system was developed as a joint effort between the departments of computer science, medical informatics and the blood center at the regional hospital in Linköping. It is a medium-size, quite typical rule-based consultation system in the MYCIN tradition, with provisions for reasoning under uncertainty (with was however used only to a very limited extent), explanations and a backward-chaining control regime.

Evaluation of the prototype system showed that it produced more reliable recommendations than those actually carried out in a historical comparison and that a 10% increase in efficiency (eliminating uninformative tests) could be expected. Using the system on a routine basis would however presume a migration from the current DECsystem-10 Interlisp-based implementation to the existing laboratory systems environment using MUMPS on a Vax computer. Then substantial parts of the interactive data entry process could also be substituted by accesses to computer-stored patient data.

In a previous project (MEDICS) we successfully used a corresponding strategy. In that system medical computer-aided learning programs (for simulation of clinical decision making) to be run on a PC are generated from an expert-system-like development environment.

The first step in such a migration project has been undertaken for A^3. An EMYCIN-compatible core system has been written for MUMPS and a semi-automated translation system of the rule base from Lisp to MUMPS makes the migration smooth[2].

However the basic goal from our point of view is to develop a systematic methodology and a general architecture supporting a division between a knowledge acquisition and development environment and multiple target environments. Central topics are which degree of automation can be achieved, to what extent the rule system should be "compiled", how much flexibility should be allowed for maintenance of the production knowledge base, etc. In this migration project we also study how *transparency*, i.e. the ability to display and explain the knowledge in the system, can be supported effectively

---

[1] **Sandahl**, Creating an Antibody Analysis Advisor as an Exploratory investigation into Expert System Development. Report LiTH-IDA-R-85-20.

[2] **Shasavar**, Portering av A^3 kunskapsbas från EMYCIN till MUMPS, Report LiTH-IDA-Ex-85-26.

also in routine use.

Knowledge-based human-computer interaction.

Previous work in ASLAB has emphasized the importance of human-computer interaction in various respects. Thus we have worked on models for dialogue management systems and their use for software prototyping, as well as on authoring environments for eductional software, in particular for medical simulations.

In our view, human-computer interaction can not be studied out of context. It appears that generally applicable results concerning dialogue design guidelines and interaction techniques are scarce and that the application-dependent aspects of a particular human-computer interface are of prime importance. We also believe that knowledge-based systems provide an appropriate background for development of high-quality interfaces, where aspects of dialogue initiative, sequencing, help and explanation facilitites, division of tasks between user and system respectively, etc. are primary, while syntactic details of the language used are secondary factors.

Important subjects for study in ASLAB are knowledge-based models of human-computer interaction, effective methods for producing explanations of system behaviour and results, and tutoring techniques for buildup and maintenance of user competence.

Thus work on expert systems has e.g. clearly demonstrated the great practical value of even simple schemes for producing natural language presentations of facts and inference structures represented inside the system. There are at least two areas which deserve to be studied further and where obvious applications are available in industry-related projects at our department:

- *More effective ways of producing help and explanations.* Most tools for developing expert systems which provide support for explanations use very simple techniques, e.g. display what is essentially a trace of the computatation with a limited explanatory value for human. Thus the translation of the internal representation for each piece of information needs to be supplemented with an intelligent selection strategy based on a model of the user's cognitive understanding of what is going on. Especially in the knowledge transfer applications where legal and economic advicing in business and banking is studied, such topics are of prime importance.
- *Alternative ways of presenting aggregated (numerical) information with an emphasis on "interesting" figures.* Traditionally computers have been used as a tool to summarise and present aggregated information with the help of tables or graphical diagrams. However expert systems such as e.g. PUFF[1] have shown that very high quality interpretations expressed as a piece of text can be produced from time-series of data,

[1] **Aikins, J.S.,** et al., PUFF: An Expert System for the Interpretation of Pulmonary Function Data, Report STAN-CS-82-931, Stanford University, 1982.

with a verbal summary of noteworthy deviations from normal figures and conclusions based on the aggregated information. The development of such presentation techniques is relevant also for the group working with statistical information systems (*Sundgren*), as well as for e.g. the previously mentioned banking applications where financial data is to be supervised and different trends identified and commented.

This area will be covered primarily by Ivan Rankin, who previously has been affiliated with the natural language group in AILAB.

We also expect to continue efforts aiming at techniques for providing tutoring capabilities in software systems, as demonstrated in the experiment conducted by Henrik Nordin and Christer Hansson. (See the discussion of the POZZO and WATT systems above.) It should be an important motivation for the approach to applications software development taken in ASLAB, that domain knowledge motivated by problem solving needs could be reused with little effort to provide tutoring and explanation capabilities.

Knowledge-based systems development

The impact of knowledge-based techniques on systems development methodology can be twofold. Either we use these techniques to support the development process, e.g. by introducing new tools or improving the old ones, or else we change the methodologies, e.g. by substituting automated procedures for work previously carried out manually.

We believe that a combination of those effects will happen in the near future. Thus for instance the availability of powerful techniques to represent and manipulate domain knowledge about objects, concepts and procedures, etc. will in a decisive way improve the possibilities to employ methods in the tradition of the *rapid prototyping* approach to systems development[1].

Other aspects of this issue are currently studied within the laboratory. During 1985-86 we have a guest researcher, Kevin Ryan, from Trinity College in Dublin, who is working for the ESPRIT ToolUse project. That project is concerned with the study of software engineering environments, and in particular with the possibilities to integrate tools supporting method-based software development. Dr Ryan's work here is concentrating on the investigation of knowledge-based support tools for a method-driven environment. Problems studied involve support for requirements engineering and knowledge-based systems design. Experimental implementations are carried out, e.g. concerning support for systems design with the JSD method using the KEE system on Xerox Lispmachines.

Starting January 1986, Pål Sorgaard from Aarhus has joined the group for two and a half year to complete his doctorate studies. His background is in systemeering and systems development practices, and his interest now is in the study of potential impacts from knowledge-based techniques on the systems

[1] **Hägglund,** From Rapid Prototyping to Stepwise Structuring and Knowledge-Based Software Development, ASLAB Memo 85-03.

development process.

3.3.2 Statistical information systems and database technology.

(*Sundgren, Krysander, Block, Chowdhury*, et al.)

This group was formed during 1983-84 when Bo Sundgren joined the department as a part time adjunct professor. The group is partly funded from separate sources. The following main issues for potential studies have been identified:

1. **Utilizing administrative data for statistics production.** The idea is to study how existing data can be used also for statistical purposes, without loss of quality.
2. **Metods and tools for automated production of statistics software.** This area can be seen as a specialization of query languages and program generation techniques for databases in general.
3. **Human-computer interaction in statistics production.**
4. **Management of uncertain and incomplete data.** Especially the relation between quality measures in statistics production and uncertainty as handled in expert systems will be studied.
5. **Systemeering methodology for statistical information systems.** This area continues Bo Sundgrens earlier work on infological and conceptual modelling.
6. **Medical applications of statistical information systems.**

In these areas a basis of competence and partners for cooperation are available. The areas also represent domains where significant research problems can be identified. The work carried out so far has exphasized areas 1-3, as will be explained in greater detail below.

The main area of study for this group is *statistical information systems*, i.e. systems for observation, collection, entry, storing, processing and retrieval/presentation/distribution of aggregated information concerning groups of objects (or higher level objects) in the current universe of discourse. Important aspects here are problems regarding quality of information (e.g. incomplete, unreliable, or misused data), support for selection of methods and tools for statistics production, techniques for interpretation and presentation of results and formal methods for description of statistical operators.

Reusing administrative data for statistics production.

Many companies experience a growing need to improve their ability to make rapid and appropriate decisions in areas where the access to current and correct information is crucial. For instance, large computer-stored databases of administrative information might be used also for purposes regarding market strategies, investment decisions, organizational planning, etc. However such a

reuse of data collected for other purposes involves serious problems with respect to data quality and interpretation. On the other hand, the costs for collecting data exclusively for decision support purposes would be prohibitive.

In order to gain an understanding of current practices and problems involved, a study has been carried out of the information systems within a few manufacturing companies[1]. These studies indicate serious problems in the companies with a systematic misinterpretation of trends in sales and invoicing, due to a lacking understanding of the statistical methods underlying available software.

Continued work involves the development of a statistical information system, with a design which minimizes the problems with using data collected for administrative purposes, selecting appropriate statistical methods and software, and correctly interpreting the results. (Wallgren and Wallgren.) The part of the work being done in ASLAB is primarily concerned with employing knowledge-based techniques as explained below.

The Statistician's Workstation.

Current efforts include the study of consultation systems for statistical analysis. The background is the well-known problem of understanding how to apply different tools for statistical analysis as correctly as possible on a given data material. The broad availability of statistical library software as well as computer-stored information bases will significantly increase the danger of misuse or even making faulty conclusions due to a lacking understanding of the often intricate problems involved in the proper use and interpretation of statistical data.

It should thus be important that intelligent advicing facilities are incorporated in a computer-based environment for statistics production. Issues to be treated include how to prepare the materials before processing, which methods to apply, how to present the results in order to avoid misleading interpretations and how to handle the data quality problem. The last aspect can be exemplified with the desire to use already existing data produced for administrative purposes as a substitute for making special inquires (which may be costly or result in incomplete data). In this case it is extremely important to be able to control problems relating to possibly different semantics for existing and required data respectively (for instance, there are something like 30 different working definitions of the concept "income" used in Swedish social security systems).

The goal is to build an integrated environment to support the analysis and effective presentation of aggregated information. Subtasks involve quality control of available data, assistance for selection of appropriate statistical methods, for adjustment of data, and for preparing parameters for the corresponding analysis programs, support for interpretation of results and for

---

[1] **Wallgren**, and **Wallgren**, Företagets informationssystem. Statistisk analys med företagets administrativa data. Memo LiU-MAI 86-01-16.

tabular, graphical and verbal presentation of abstracted information. In a preliminary study a knowledge-based assistant for eliminating seasonal variations in statistical studies of industry and trade has been implemented on a PC, using the SAGE shell system (Block). However, more advanced support facilities would be needed, such as e.g. graphical interaction techniques at various stages of the consultation session. The current project studies the design of a *statistician's workstation* implemented in Lisp on a Xerox Lispmachine. Windowing techniques will be used to run standard library software on backend computers in parallel with the preparation and analysis session. (Sundgren, Chowdhury, et al.)

An algebra of base operators for production of statistics.

On a high level the statistical production process may be regarded as a system of production functions like editing and correction of data, tabulation, graphical presentation, and statistical analysis. Generalized statistical software is usually developed for functions on this level. However, the high-level functions may be defined in terms of simpler more general, and logically better defined subfunctions like selection of certain objects on the basis of certain criteria, creation of new variables in terms of existing ones by means of logical and arithmetic operations, aggregations of data in accordance with cross-defined and/or hierarchically defined aggregation structures, etc.

If these subfunctions were generally recognized, and common definitions were widely accepted, it would pave the road for a much more powerful and dynamical development of generalized statistical software, in comparison with the situation today, where packages differ significantly between themselves with regard to the above-mentioned breakdown of major functions into subfunctions, and where the subfunctions are often mixed up even within one and the same, monolithically designed piece of software.

A relational database algebra containing the traditional set operations and the special relational operations can be proven to be "complete" in a certain sense. However relational completeness is not enough to ensure that a language will be practically (and probably not even theoretically) sufficient to formalize the mechanics of statistics production, such as e.g. the typical statistical aggregation processes (counting, summation, average calculation, etc.) if no additional base operators were permitted.

A set of such *statistical base operators* is being defined in an international joint effort with participation of Statistics Sweden. In connection with that work, we are developing a theoretical framework in the form of an algebra, with the purpose of giving a firm basis for implementation and application of such operators as well as a conceptual integration with current practices in the field of relational databases. (Bo Sundgren, Gösta Nilsson.)

3.4 External cooperation.

Aslab projects emphasize joint efforts with other groups and industry. The following are the main current involvements:

1. Department of Medical Informatics. Previous cooperation on advanced CAI systems (MEDICS) is now followed by joint work on medical expert systems (Gill, Reshagen, Timpka).

2. Alfa-Laval Automation. Joint work in the area of fault diagnosis and maintenance expert systems within the Knowledge Transfer Program (Rosenberg, Hanebring).

3. ASEA. Previous cooperation on a consultation system for robot configuration is now followed by Knowledge Transfer Program activities (Lozan-Villegas).

4. Ericsson Information Systems and S-E-banken. Joint activities in the Knowledge Transfer Program with an emphasis on intelligent knowledge-based consultation systems in business and banking and on other end-user systems. (Bengtsson, et al.).

5. National Bureau of Statistics. Study of the design of statistical information systems. (Block, Nilsson, Wallgren and Wallgren. See also above.)

6. Nordic cooperation with Oslo (Kristen Nygaard) and Århus (Lars Mathiassen) in the SYDPOL programme (System Development Environments for Profession-Oriented Languages.) This programme is partly supported by Nordforsk and consists of national projects and four inter-nordic working groups, where Aslab participates in the one concerned with systems developments methods.

3.5 Publications

For a full listing of published papers, including departmental reports, see the "publications" appendix. Below the most recent external publications are listed for an easy reference.

External publications:

1. **Sture Hägglund** et al., AI in Engineering Applications at Linköping University, *SIGART Newsletter*, no 92, April 1985.

2. **Sture Hägglund**, Kunskapsbaserade expertsystem, rapport Sv. Mekanförbund, 86001.

3. **Hägglund, Nordin, Rehmnert, Sandahl**, Utveckling av kunskapsbaserade expertsystem i samarbete högskola-näringsliv. (Collection of contributions to *NordDATA 85*, Copenhagen, 1985.)

4. **Henrik Nordin:** Using Typical Cases for Knowledge-Based Consultation and Teaching. To appear in *Proc of the 3rd Annual Conf. on Applications of Expert Systems*, Orlando, Fla., 1986.
5. **Roland Rehmnert, Kristian Sandahl**: Knowledge Organization in an Expert System for Spot-Welding Robot Configuration. In *Proc. of the 5th Int. Workshop on Expert Systems and Their Applications,* Avignon, 1985.
6. **Kristian Sandahl, Sture Hägglund, Jan-Olof Hildén, Roland Rehmnert, Lars Reshagen:** The Antibody Analysis Advisor and its Migration into a Production Environment. To appear in *Proc. of the 1st Int. Conf. on Expert Systems*, London 1985.
7. **Bo Sundgren:** How to Satisfy a Statistical Agency's Need for General Survey Processing Programs. *Proc. of the 45th Session of the International Statistical Institute*, Amsterdam, Aug 12-22, 1985.

In addition to external publications and departmental reports, ASLAB has instituted a Memo series containing working papers etc. The following reports have been issued up till now.

ASLAB Memo series 1984-

84-01 **Brüer, Chowdhury, Fäldt, Gill and Rönnquist:** Office Models.

84-02 **Sundgren, et al.,**: Statistiska Informationssystem (Statistical Information Systems).

84-03 **Rönnquist**: Customizing Command Languages Dialogues with the YAKI package.

84-04 **Moen**: En implementering av Expert Ease under Interlisp-D.

84-05 **Rehmnert, Sandahl**: Implementing an Expert System for Spot-Welding Robot Configuration.

84-06 **Hägglund**: Introduktion till kunskapsbaserade expertsystem i ekonomiskt-juridiskt arbete.

84-07 **Hägglund, Nordin, Rehmnert and Sandahl**: Towards Knowledge-Based Applications Software Environments. Project Proposal.

85-01 **Nordin,** Knowledge Reuse in a Back-Office Expert System.

85-02 **Bengtsson,** LUCKY System Documentation.

85-03 **Hägglund,** From Rapid Prototyping to Stepwise Structuring and Knowledge-Based Software Development.

85-04 **Sundgren,** Outline of an Algebra of Base Operators for Production of Statistics.

85-05 **Doherty,** A Rule Interpreter for an EMYCIN-like Expert System Tool.

85-06 **Moen**: Expert-Trees User Manual

85-07 **Hägglund et al.**, Feature Catalogue of Tools for Building Expert Systems. Draft version.

86-01 **Hanson**, WATT - A Knowledge-Based Case-Oriented Teaching System.

4.

AILAB
The Artificial Intelligence Laboratory

Erik Tengvald, et al.

The research activity of AILAB is concentrated in two main research areas namely: *knowledge representation* and *natural language.*

On the Knowledge representation side the mainstream of activities has focused on the design of programming systems for A.I., as studied through applications selected from Mechanical Engineering. A major early work is OBS an operations planning system for turning [Tengvald 84]. This system was based on the object oriented representation system PAUL [Hein 83]

Most of later and current research in knowledge representation at the AILAB is based in the considerable experimental experience of the OBS/PAUL project.

One of the experiences of this project is that the mainly procedural object-oriented programming systems are insufficient for handling geometrical problems. The object-oriented programming system has to be supplemented with a more declarative programming paradigm.

In the below described ICONStraint project Jalal Maleki has researched the possibility of using the constraint programming paradigm as basis for such a declarative system. The choice of constraints as opposed to other declarative paradigms was based in another experimental experience of the OBS project, namely the very great computational demands of geometric reasoning.

Preliminary findings from the ICONStraint project indicate that the expressibility of constraints is indeed sufficient for handling most geometric problems encountered in the OBS setting. Unfortunately it seems as if

---

The work in AILAB is mainly supported by STU, The Swedish Board for Technical Development.

constraint based systems running on todays machines is too slow to make the explorative programming method applicable in a geometric reasoning context.

Constraint based systems are maybe the most efficient of the declarative programming systems. Consequently, our experience indicate that the the possibility of constructing expert systems with a substantial content of geometric reasoning is slim indeed.

Alas, if you do not increase the raw processing power of the hardware. This is the intended method of attack chosen in the below described AIM project. The AIM project is a major undertaking with eleven members in the project group. The project was initiated in this summer after prodding from industry.

A subsidiary knowledge representation research activity at AILAB has been Jim Goodwins extremely promising but more formal research on non-monotonic logic. The formation of the RKLlab, a group with a research program in the formal knowledge representation area, made it possible for Jim Goodwin to pursue his work together with more like-minded colleagues. Consequently, he left AILAB for RKLLAB. The continuation of his work is reported under the RKLLAB heading.

During the years 82/83 and 83/84, the AILAB did not manage to create a critical-size natural language activity. For example Carl Wilhelm Welin and Ludmila Ohlsson left the group for Ericsson. However, we have always considered it important to develop an appropriate research program in this area, especially with research on the Swedish language.

This year has consequently been a year of recruiting activity. We have succeeded in employing a total of five new researchers. The natural language subgroup have increased from one to six persons. With this increase combined with an appropriate spread of competence, including linguistics, psychology, computer science and artificial intelligence, we consider the natural language group as stably formed.

Due to the increase of both the natural language and knowledge representation sides of the AILAB research group, we are planning to split AILAB into two research laboratories in the middle of the coming summer. It is our experience that too big research groups do not function well.

4.1 Research on knowledge representation

The knowledge representation research is primarily directed towards geometric reasoning or more precisely on the combination of symbolic and geometric reasoning.

During this year a project, ICONStraint, concerned with research within the constraint programming paradigm has been carried through by Jalal Maleki. This work will result in a Masters thesis, during the spring 1986.

Furthermore a new large scale project, Artificial Intelligence for Manufacturing, has been initiated. It is planned to run for 2.5 years. The project can be seen as a new experiment in the OBS tradition.

It was our intention to invest more work in purely representational questions before undertaking such major practical experiment again. However, preliminary results from the ICONStraint project are so promising that we consider it justified to restart a full scale project this early.

4.1.1 Laboratory members working in the knowledge representation area

AILAB members working on knowledge representation during 1985 has been:

Erik Tengvald, Ph.D., 75%
Bernt Nilsson, 30%
Jalal Maleki, B.Sc., 50%

New members joining AILAB for the aim project at January, 1th, 1986 will be:

Leif Finmo, M.Sc., 75%
Johan Andersson, B.A., 40%
Patrick Doherty, 75%
Peter Haneclou, 75%
Mikael Svensson, 75%
Anders Nyberg, 75%
Sven Moen, 25%
Jonas Wallgren, 25%
Håkan Jakobsson, 25%

4.1.2 The ICONStraint project

ICONStraint is an interactive declarative programming language based on the constraint model of computation. ICONStraint is a fourth generation language which is quite similar to systems such as Visicalc systems, but is a more general system. This language allows declaring a set of relationships (constraints) that are to hold among a set of variables. Once a number of these variables are known, the interpreter enforces the declared relationships and as a result the values of other variables are computed. This process is called constraint propagation.

In constraint based languages, objects (usually representing physical or computational devices) are described in terms of a set of variables and the constraint relationships that hold among them. Each variable is supplied with a set of rules that compute its value when certain other variables are known. The set of all rules defined for the variables of an object define the

computation local to the object. Restricting computation to be local to objects takes away the burden of defining the global control in the program (very much like a logic programming language). The effects of local computation within objects are passed to other objects via so called "wires" that are equality constraints that force variables to have the same values.

Restricting the computation to be local to individual constraints provides a suitable model for parallel computation. The current system could with some relatively small effort be modified for a parallel computer. This is possible because the propagation process works by taking tasks from an agenda and executing them. These tasks are independent rule applications, and therefore carrying them out in parallel is possible. On the other hand, a disadvantage of not having a global control in the system raises the problem of inconsistency. An inconsistency occurs when two wired variables are assigned different values. In order to facilitate resolution of such contradictory states, dependency information concerning the way in which the value of a variable was computed is explicitly stored and is inspected when contradictions arise. This is done by following the dependency chains to the premises of computed values, and then retracting these premises either automatically or under the guidance of the user until the contradictory state is resolved.

By keeping the dependency information we are also able to avoid unnecessary computation. That is, under the same premises a given computation is only done once. We have applied this principle of avoiding unnecessary computation for early detection of value assignments that cause contradictions. This is done as follows: once a contradiction occurs, the premises of the contradiction, that is the set of variable bindings causing the contradiction, are explicitly stored for future use. The interpreter ensures that this set will never be allowed in future. In this way, already visited blind alleys are not tried again. But one question concerning the gained efficiency remains that we leave for future research.

4.1.3 The AIM project

The purpose of the AIM-project is to significantly reduce the cost of the majority of the knowledge production activities in the manufacturing industry. The knowledge production activities is rapidly becoming the major part of the industry's overall activity. For an advanced product like a modern jet engine, the specification and documentation can very well weigh more than the product itself.

The majority of the knowledge production activities in the manufacturing industry are based on reasoning processes where there is many and complex interdependencies between the geometric and symbolic reasoning steps.

We have found it useful to coin a new word for this kind of reasoning, namely geombolic reasoning. The characterizing trait of geombolic reasoning is the

intense interdependence between the geometric and the symbolic reasoning processes.

The intended mode of attack on the geombolic reasoning research topic is by the classic AI approach of creating a proper research vehicle. In such a research vehicle one is able to perform experiments to acquire experience grounded in real world problems. The goal of the AIM-project is to create such a research vehicle.

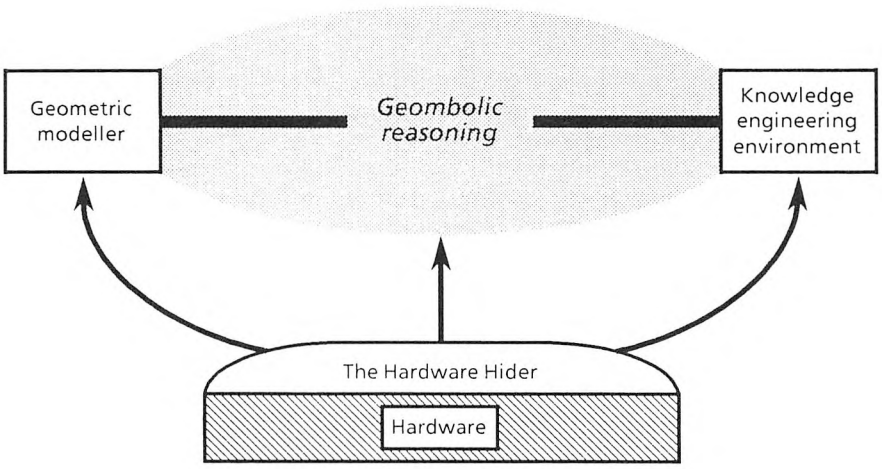


Figure 4.1. *The overall structure of the research vehicle.*

It is immediately apparent that the geombolic reasoning topic is situated in the middle ground between the two poles of geometric modelling and knowledge engineering.

The geombolic reasoning problem is an almost open problem. To our knowledge there is only one paper [Ballard 84] which addresses the problem in a systematic manner. Because of the subject's importance there are of course many papers which touch on the geombolic reasoning problem.

We believe that the main reason behind the lack of research on the problem of geombolic reasoning is the lack of sufficient computing resources. An increase of at least two or three orders of magnitude over the computing power of today's AI machines will most certainly be needed.

Consequently, the research vehicle will have to be supported by a parallel processing machine. We have been searching the market for a suitable machine, but have not found any. Currently, there is no machine which can meet the general computation needs of the geometric representation system and knowledge engineering environment together with the special graphics

capability requirements of the geometric presentation system. At least not at a reasonable price. We will have to build the hardware ourselves.

The software requirements of the knowledge engineering environment are considerable. Knowledge engineering environments are complex systems containing a plethora of detail. If we were to implement the knowledge engineering environment directly in assembler or even a systems programming language like C or Occam, our chances of meeting the project deadline would be slim indeed. Consequently, we must introduce system software defining an intermediate language higher than the systems programming languages. We call this system software: The hardware hider.

A hardware hider is basic systemware, which hides the complexity of the hardware. More precisely, it frees the programmer from the tedious tasks of memory and process management. This must be done without introducing any unnecessary restrictions in the use of the hardware. Metaphorically: A hardware hider is hiding the nitty gritty details of the hardware, without hiding it's soul. This is illustrated in Fig. 4.1.

Based on the above considerations we have found it appropriate to divide the AIM project into four main subprojects, namely:

1. Powershape: The hardware
2. Hideshape: The hardware hider
3. Solidshape: The geometric modeler
4. Knowledgeshape: The knowledge programming environment

Powershape.

The parallel hardware will have a message based architecture. We will probably use the now available Inmos T414 transputers as processors in our first machine. This has four on chip inter-processor links. Our main candidate net topologies is cross connected cube connected cycles and toroidal square mesh.

We are designing for a net of 64 transputers. The nominal execution speed will be 640 Mips. But only 3.2 Mflops. According to Inmos's, maybe a bit slanted benchmarks, the power of a transputer is 4 times that of a Motorola 68020. Taking this at face value we can say that our machine will execute 256 times faster than a Sun-3. Extending this with the Dhrystone benchmarks of [Olafsson 85] we can say that our machine is: 562 times faster than a VAX 11/780, 989 times faster than a VAX 11/750, 2872 times faster than a IBMPC/XT.

In a later version we will replace the T414 with the announced but currently unavailable Inmos F424 transputer. This will result in something like a tenfold increase of the floating point capacity up to 32+ MFlops.

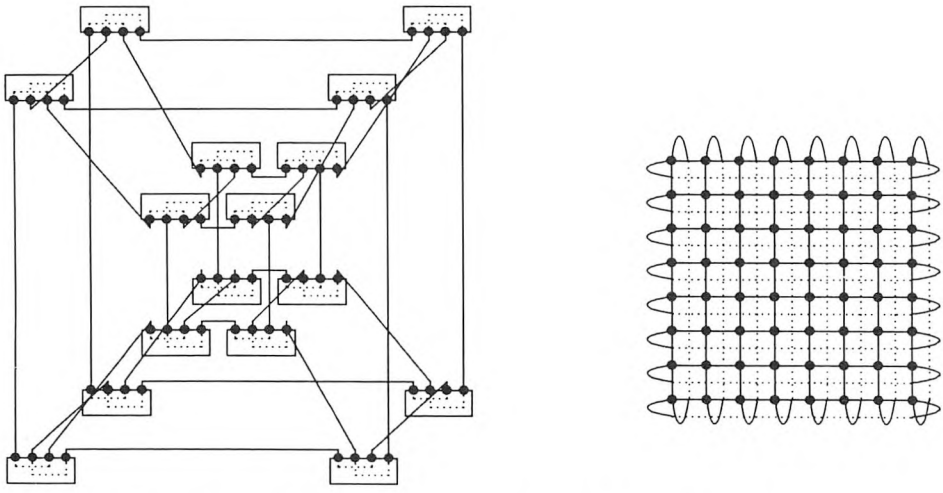


Figure 4.2. *Cross connected cube connected cycles and toroidal square mesh.*

The T414 version is planned to be up and running in October 1986. Inmos hopes to be able to introduce the F424 in the first quarter of 1987, we should have the F424 version up and running 2 weeks after deliveries.

Hideshape.

The hardware hider is called Hideshape, since it hides the complex shape of the hardware. It is responsible for memory and process management. Its two main components are a garbage collector and a process allocator.

The hideshape system defines a programming language. More precisely, it is the interpreter/compiler for this language. This language can be viewed as a very high level assembler, in which the higher levels of the knowledge system are to be implemented. Our current working hypothesis is that this language should be a functional language with assignment. The language is intended for systems programming and should be close enough to the hardware.

Solidshape.

Our geometric modeler is to be a standard Constructive Solid Geometry system with Euler capability. It is to be implemented in Hideshape. Thus it will be easily extendable as need for more esoteric geometric operation arise.

Knowledgeshape.

Our knowledge engineering environment is intended to be pluralistic environment supporting many programming paradigms under an integrated debugging environment. During the AIM planning period, up to 1 july 1988 , we hope to incorporate the object oriented and constraint programming paradigms.

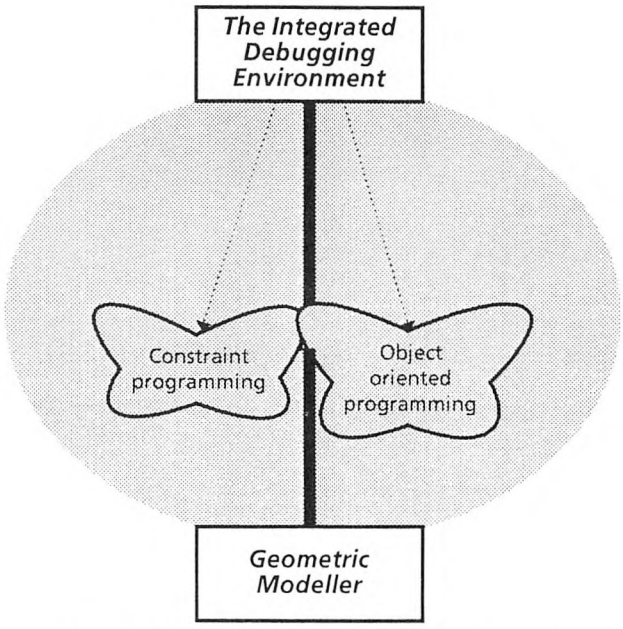


Figure 4.3. *The knowledge engineering environment*

Object oriented programming is very natural when describing systems of interacting things. Such systems are very common in the manufacturing industry. Moreover the window interface is best implemented using object oriented programming. Constraints are perfect for describing invertible relationships. Most relationships of physics and geometry are invertible.

We hope to establish practical cooperation with logic programming groups at our department and at SICS with the intent to begin incorporating the logic programming paradigm in knowledgeshape during the later half of the planning period.

Research cooperation.

We participate in the COST-13 project number 21, Advanced Issues in Knowledge Representation. We are members of the PARSYM computer mail group and regularly get the PARSYM digest via the computer mail network.

We are in contact with the companies: SAAB, Sandvik, McAuto and Computer Vision. Hopefully we will be able to establish contacts with other companies during the planning period.

4.2 Research in natural-language processing

The research of the group for natural-language processing (the NLP group) is primarily directed towards the development and use of natural-language interfaces (NLIs) to computer systems. The emphasis is put on communication in the Swedish language. Most of this research is carried out within the project "Analysis and Generation of Natural-Language Texts", financed by STU.

The following four research areas are of special interest to us:

1. Parsing and semantic interpretation;
2. text generation;
3. the integration of different knowledge bases — linguistic knowledge, pragmatic knowledge, domain knowledge — into a working system having a high degree of transportability; and
4. NLIs and the human user.

In conjunction with the first area we maintain a library of standard NLP software with Swedish applications, and sometimes other languages as well. Work during this year has mainly gone into the first area with a concentration on syntactic and morphological processing, while plans for work in the other areas have been developed to the point of implementation. Results and plans for the future are reported in more detail below, for each of the four areas.

4.2.1 Personnel

The members of the NLP group during 1985 have been:

Erik Tengvald, Ph.D., 25%
Mats Wirén, M.Sc., B.A., 50%
Nils Dahlbäck, B.A., 85%
Bernt Nilsson, 30%
Arne Jönsson, M.Sc., B.A., 50% (joined AILAB 1985-07-01)
Lars Ahrenberg, B.A., 85% (joined AILAB 1985-07-01)
Magnus Merkel, B.A., 75% (joined AILAB 1985-07-01)

In addition, two C-line students — Lotta Månsbacka and Ivan Rankin — have worked part-time with the group during 1985.

4.2.2 Parsing and semantic interpretation

A. Processing of Swedish Morphology

Any natural-language project concerned with the Swedish language has to pay considerable attention to morphology since the Swedish language is substantially more complex in this respect than e.g. English. Thus, one important project of ours has been to explore different theories and techniques

for processing Swedish morphology. The ultimate goal of this work is to develop a robust and efficient morphological analysis and synthesis component, which could be part of a more general system for natural-language processing.

The most important concrete result so far has been the completion during 1985 of the implementation of a parser based on Hellberg's system for Swedish morphology, described in Hellberg (1978). (The system has been implemented in Interlisp-10 and Interlisp-D.) Hellberg's system provides a highly explicit description of Swedish inflectional, derivational, and compounding morphology, all collected in 235 different "paradigms". In particular, with respect to inflections the description is the most substantial so far and could be regarded as exhaustive. The current dictionary (in the description as well as in our implementation) consists of some 8,600 word stems together with paradigm information. (The dictionary and the paradigm information was kindly provided in machine-readable format by the Department of Computational Linguistics in Göteborg.)

The experiences of the system and the implementation has been reported at the Fifth Meeting of Nordic Computational Linguists in Helsinki, December 10—12, 1985 (Rankin 1986 a). There is also a user's guide available for the program (Rankin 1986 b).

Another very promising candidate for a morphological component is the so called twolevel model, the general morphological theory of Koskenniemi (Koskenniemi 1983), which has so far been tested for about fifteen languages. During 1985 we received Lauri Karttunen's and Kimmo Koskenniemi's joint Interlisp-D implementation of the theory (i.e., a morphological parser and generator), together with a fairly substantial description of Swedish developed by Blåberg (Blåberg 1984). We are currently looking at this system to see if it can be further developed to the kind of component we are looking for.

B. LFG and The Grammar Writer's Workbench.

The Grammar Writer's Workbench, a system developed at Xerox PARC by Ron Kaplan for writing and debugging lexical-functional grammars (LFGs), has been available to the group since March 1985. We have used it to develop a small Swedish grammar and have tested it on a variety of Swedish constructions. Present findings indicate that the theoretical constraints imposed on LFGs by its creators (Bresnan 1982) are too restricted to allow for efficient use in a NLI, although the general framework could well be kept. We have presented our findings about LFG at the Fifth Meeting of Nordic Computational Linguists in Helsinki, December 10—12, 1985 (Ahrenberg 1986). A larger Swedish LFG is currently being developed as part of a Master's thesis by Lotta Månsbacka.

C. Additional parsing systems

As a basis both for continued work on natural-language processing, as well as for undergraduate courses in this field, several standard parsers have been put

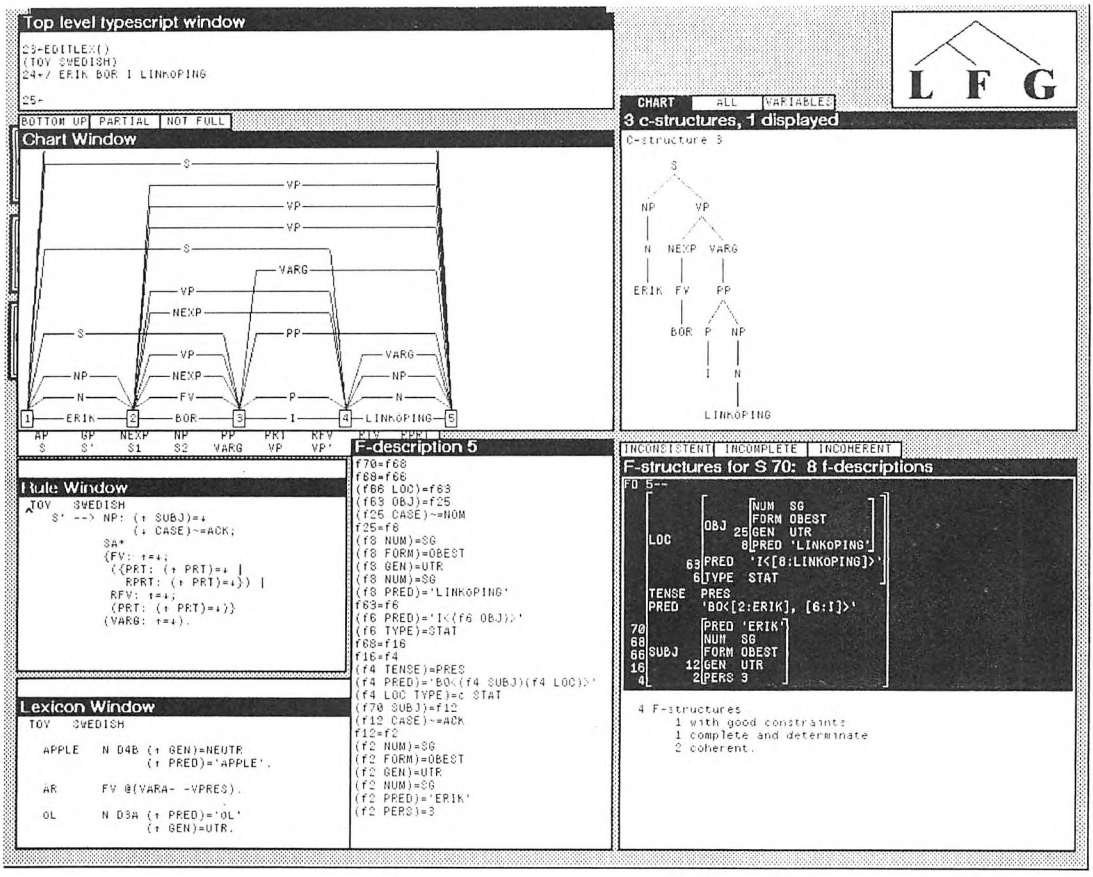


Figure 4.4. *Working with a Swedish LFG grammar.*

together by Wirén, to some extent using ideas of Winograd (1983). Wirén has also implemented a parser based on semantic grammar in the LIFER tradition (Hendrix 1977), using a simplified database as a background system. Semantic grammars, however, are only applicable when a relatively small subset of a whole language needs to be recognized, and it seems that the only way to make use of this technique in a more general system would be to regard it as just one of several accessible parsing strategies.

In addition to this, we have incorporated a number of systems from other institutions, most notably the LFG Grammar Writer's Workbench from Xerox PARC (see above), the ProGram system (a GPSG workbench from University of Sussex) (Evans & Gazdar 1984), Thompson's MCHART (a modular chart parser) (Thompson 1983), and Karttunens HUG (a chart parser with a

unification package).

4.2.3 Text generation

Work on generation of natural language has until recently been a more or less neglected area in AI and computational linguistics (cf. Mann 1982). Although the interest in the field is growing, it is still a small area compared to language understanding.

Needless to say, understanding and generation of natural language are not two completely separated research areas. Especially for "lower" linguistic levels, the theories, models, and formalisms will be basically the same. (In fact, some of the work in this area, such as functional grammar (Kay 1979) is explicitly aimed at building a system which will work both ways.) However, as has been pointed out by McKeown (1985), generation of a text poses some important problems of its own. McKeown mentions different aspects of the planning process, i.e. *what* information to communicate, *when* to say what, and *which* words and syntactic structures to use to express the intended meaning. (Of course, there is every reason to believe that more sophisticated understanding systems will have to make use of this sort of knowledge as well.)

While our group maintains a strong interest in text generation, we do not actively pursue any research in this area, mainly due to lack of resources. However, as has been suggested in the previous paragraph, work in pragmatics is highly relevant to this area, and this means that the work in the group on integration of different knowledge bases in a NLI, as well as the work on coherence mechanisms described below, means that we are building up our competence in this area, as a preparation for future work.

4.2.4 The integration of different knowledge bases in a NLI

Most existing NLIs have no or only rudimentary capabilities for handling dialogue. A reason for this is that NLIs primarily have been used as translators between natural language and an existing query language of a database, and that the user therefore has not been allowed to do anything else but ask one question at a time and be satisfied with the answer. For NLIs to be useful, in particular if we have an expert system as the background system, they must be much more flexible than that. But this means that there must be much more knowledge built into them, in particular knowledge about how to behave in a dialogue, knowledge of the user (user models) and knowledge of the dialogue that the system is presently engaged in. We refer to this as pragmatic knowledge.

Research is now underway which attempts to build pragmatic knowledge into NLIs. A problem with pragmatic knowledge is that its content is not all too well known and even less is known about how it is used in conjunction with linguistic and factual knowledge in interpretation and generation. We therefore

plan to approach these problems in a stepwise fashion.

The first pilot project will be an NLI to a simple database, a calendar. Entries, deletions as well as queries will be performed in (a subset of) natural language. The system will incorporate a primitive user model, a primitive dialogue memory and a fairly simple strategy of dialogue behaviour. The purpose of the project is (a) to study such dialogue phenomena as ellipsis, anaphoric and deictic reference, in particular temporal deictic reference, and (b) to study the integration and moduling of the different knowledge bases.

4.2.5 NLIs and the human user

Basic research in this area is being conducted by Nils Dahlbäck as a part of his thesis project on cognitive and computational aspects of text coherence. The aim of the project is to study the interplay between structure and content in the construction of coherence relations in descriptive texts. In the first phase (which is the project for a licentiate thesis), methods from experimental psycholinguistics are used to uncover the basic factors influencing human subjects construction of coherence within a text. In the second phase, the intention is to use the acquired knowledge from the first phase in the implementation of a system for establishing coherence relations in a NLI.

Psychological theories and methods can also be used in other ways in research on NLIs. First, in the initial phase of a project, simulations in which prospective users of a NLI communicates via terminal with a human actually simulating the NLI, may be very valuable. These simulations can show what the dialogue will look like, which in turn determines the necessary linguistic capacity of the system to be built.

Secondly, once the system or a prototype thereof is built, psychological assessment methods may be used to evaluate the system in a systematic way, giving important suggestions on what to improve, as well as determining the system's applicability to different prospective domains.

Methodologically, the first type poses no particular problems, as far as the collection of data is concerned. The analysis of these, however, requires extensive linguistic knowledge, especially on discourse processes, and is (as is all discourse analysis) a very time consuming work. For the second type of investigation, several methods are possible, or even necessary, as they supplement each other. A rough classification could be in "objective" measures, of which there are two types, on line and result measures, and subjective measures, of which there also are two types, interviews and questionnaires.

We plan to work in both these areas. As a first step, we will run an experiment of the first type the coming spring, as a part of the NLI pilot project. Research of the second type obviously will have to wait a while, but some of the methods being developed for or used in Dahlbäck's thesis work, such as a text presentation system for on-line measurement of readability by Jönsson and

Dahlbäck, will also be useful here.

References

[Hein 83] U. Hein, PAUL - the kernel of a representation and reasoning system for knowledge engineering tasks. 1983

[Tengvald 84] E. Tengvald, The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning. 1984

[Ballard 84] D. Ballard, Task Frames in Robot Manipulation, *Proceedings AAAI-84*. 1984

[Olafsson 85] Computer mail from: M. Olafsson, University of Alberta, Edmonton, Alberta Canada. 1985

Ahrenberg, Lars. Lexikalisk-funktionell grammatik på svenska. *Proc. of the Fifth Meeting of Nordic Computational Linguists,* Helsinki 1986. [Forthcoming]

Blåberg, Olli. *Svensk böjningsmorfologi: en tvånivåbeskrivning.* Pro gradu-avhandling i allmän språkvetenskap, Helsingfors universitet 1984.

Bresnan, Joan (ed.). *The Mental Representation of Grammatical Relations.* MIT Press 1982.

Evans, Roger, och Gerald Gazdar. *The ProGram Manual.* Cognitive Studies Programme, University of Sussex 1984.

Hellberg, Staffan. *The Morphology of Present-Day Swedish.* Almkvist & Wiksell International 1978.

Hendrix, Gary G. *The LIFER Manual. A Guide to Building Practical Natural Language Interfaces.* SRI Technical Note 138 1977 a.

Kay, Martin. Functional Grammar. *Proc. 5th Ann. Meeting of the Berkeley Ling. Soc.* 1979.

Koskenniemi, Kimmo. *Two-level Morphology: A General Computational Model for Word-form Recognition and Production.* University of Helsinki, Department of General Linguistics, Publication 11.

Mann, William. Text Generation. *American Journal of Computational Linguistics* 8:2 1982.

McKeown, Kathleen R. Discourse Strategies for Generation of Natural-Language Text. *Artificial Intelligence* vol. 27 1985.

Rankin, Ivan. On the Implementation of Hellberg's Morphology System. *Proc. of the Fifth Meeting of Nordic Computational Linguists,* Helsinki 1986 a. [Forthcoming]

Rankin, Ivan. SMORF User's Guide. IDA 1986 b. [Forthcoming]

Thompson, Henry. MCHART: A Flexible, Modular Chart Parsing System. *AAAI* 1983.

Winograd, Terry. *Language as a Cognitive Process. Volume I: Syntax.* Addison-Wesley 1983.

5.

CADLAB
The Laboratory for Computer-Aided Design of Digital Systems

Harold W. Lawson, Jr.
Professor of telecommunications
and computer systems.

5.1 Introduction

The laboratory for Computer Aided Design of Digital Systems, CADLAB, is concerned with the behavioral and structural aspects of the specification, design, simulation, optimization, partitioning, synthesis and evaluation of digital systems, especially those involving very large scale integrated circuits (VLSI).

CADLAB was formed from Professor Harold Lawson's Telesystem group when Telesystem, Datalogi and ADB merged to form IDA in 1983. In addition to its being part of IDA, CADLAB also cooperates with Professor Christer Svensson's group in IFM (Physics) and the Applied Electronics group in ISY (Electrical Engineering) to form the VLSI Design Center at Linköping.

The first years of VLSI Design Center were devoted to building competence and acquiring basic software. The year 83/84 marked the transition from the building phase to initiating new research. The "fruits" of these early years are now being harvested and CADLAB is able to report on some significant progress made during 1985. This progress will be highlighted in a later section. CADLAB is broadly concerned with many aspects of the problem of silicon compilation; the process of translating a high level description of a system to a

---

The work in CADLAB is mainly supported by STU, The Swedish Board for Technical Development.

silicon layout. One model of the silicon compiler is that of a translator which takes a high level description of a chip and transforms the semantic content of the description into a machine as indicated in Fig. 5.1.

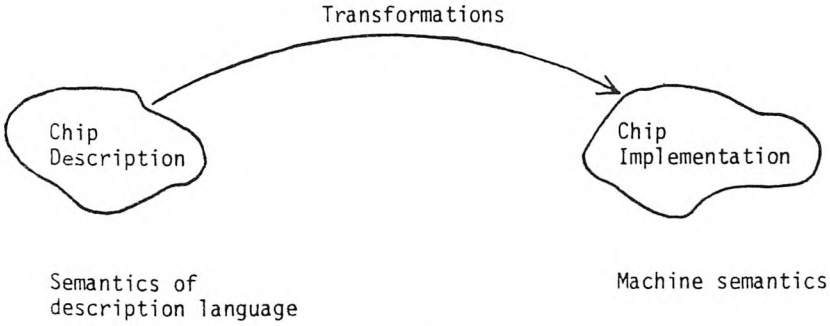


Figure 5.1. *Silicon Compiler Model.*

We do not expect a single programming language to be equally useful for all applications, a single uni or multi-processor machine architecture to run all applications equally well, or a single set of compiler optimizations to match all languages and machine architectures. We still have need for LISP, COBOL and ADA; for VAXs and CRAYs and IBMs as well as newer languages and architectures for parallel computation. Likewise, we should not expect a single hardware description language and its associated silicon compiler to solve all problems.

5.2 Current Work

CADLAB is currently engaged in a research project called ASAP (An Architectural Strategy for Asynchronous Processing) which is an attempt to provide an architectural basis for a new generation of sophisticated CAD tools. Specifically, we are interested in exploring the implications of asynchronous design and distributed control. Rather than attempting to compile systems that are general purpose, we are interested in special purpose systems that are embedded in a wide variety of products such as telecommunication systems, electronic and biological instruments, robots and automatic control systems. The basic assumption is that in embedded, or special purpose systems, the structure of the application is well defined. In such environments, we are faced with only a small number of programs and the payoff in being able to specialize the system is quite high. In sections 5.3 to 5.7, we now identify the major premises of the ASAP project.

5.3 VLSI layout and Timing Problems

Shrinking VLSI geometries promise both denser chips and decreased switching times at the gate level. Unfortunately, while gate delays scale linearly, the RC-line delay for communications between gates does not scale, thus leading to a situation where the chip speed is limited by the interconnections. Even with today's technology "it takes about as long for a signal to cross a chip of side .5mm as it does to go along a coaxial cable 75cm long."

The constant propagation delays and decreasing gate delays mean that a system-wide clock whose rate is proportional to the gate delay is not possible if we are to maintain propagation delays at a small (five to ten percent) fraction of the clock period. However, within regions of a VLSI circuit, called isochronous or equipotential, the system may be considered synchronous at the maximum clock rate permitted by the circuit technology. The problem is then to synchronize independent isochronous regions. This may be implemented as either a self-timed discipline or based on a hierarchy of clocks.

The concept of isochronous regions on a chip is generalizable to that of a packaging hierarchy for computers. Conventional packaging groups circuits into chips, sometimes chip carriers, printed circuit boards, and cabinets. Current high performance technologies group the chips into modules and the modules into frames. Future technologies, including wafer scale integration and 3-D structures, should be thought of as extensions of the goal of optimizing machine performance by minimizing communications costs.

What is lacking is a framework for a design system in which machines can be designed without regard to the implementation or packaging of their component modules. The design system should allow the designer to explore the performance and cost implications of different implementations or attempt to automatically provide an "optimized" solution from a library of function implementations.

5.4 Timing and the Exploitation of Natural Parallelism

Typical tasks to be performed in embedded systems are data capture, processing, control signal generation, display maintenance and possibly statistics gathering. These heterogeneous tasks when implemented for a *processor* (which we shall abbreviate as *pr*) are viewed as *processes* (which we shall abbreviate to *ps*). For conventional general purpose *pr's* (including microprocessors), the *ps's* are programmed in a "language" suitable for the *pr* and mapped into the sequential execution domain of the *pr*. The treatment of timing idiosyncrasies of the application domain are placed under the jurisdiction of a real time operating system (also operating in the sequential execution domain of the *pr*). Thus, the potentially highly parallel environment of the application tasks is placed into the procrustean limits of the sequential execution of a single *pr* under an operating system with a potentially

complicated interrupt and process management structure. Any potential parallelism implicit in the application tasks is thereby lost.

In embedded systems, it would be desirable to exploit the inherent and natural *parallelism* of the application tasks. In order to accomplish this property and to eliminate the complexities and eventual problems with centralized real time operating systems, we require asynchronous processing (at potentially very high levels of granularity) and distributed processing logic.

5.5 Extensibility and Specialization

Instead of binding the solution of *ps* realization to a single general purpose *pr*, we establish the possibility of selecting the best specialized form of *pr* for accomplishing the task of the *ps*. A family of potential *pr* realization methods should be available in the library of the accompanying CAD system. In the simplest case, the *pr* could be a combinatorial network of logic and/or an analogue circuit. We shall term such logic as *base logic pr's*. Above this level, we introduce *programmed logic pr's* which in their simplest form could be programmable logic arrays (PLA's) or some form of structure resembling various microprogrammed architectures. Further, the notion of programmed logic *pr's* is to be interpreted in its widest meaning to include logic for finite state machines, extended finite state machines, as well as instruction set processors for concrete realizations of programmed logic at any level of abstraction.

One can potentially think of a specialized *pr* for each and every *ps* of the application. This specialization may even be the case in certain realizations; however, when the nature of the problem (including timing constraints) permits, several *ps's* may indeed be multiplexed onto a single programmed logic *pr*. The individual *ps's* are suitably decomposed into the execution representation required by the *pr*. In this case, local scheduling logic for allocating the use of the *pr* must be provided.

By distributing the processing logic and utilizing the asynchronous approach to timing, extensibility is easier to attain. New processing tasks of *ps's* can be added without making new demands upon the processing capability of a single *pr* upon which all *ps's* are multiplexed. The extensibility is provided in the framework of the asynchronous processing strategy, by the use of generalized interprocess communications which can be automatically "tailored" to the requirements of the *ps's* of the embedded system.

5.6 Hardware - Software Tradeoffs

By providing an environment where the selection of *ps:pr* pairings is based upon the criterion of finding the best *pr* for a given *ps*, and where timing is handled by the asynchronous strategy, we have an ideal environment for evaluating hardware-software tradeoffs. The nature and complexity of the algorithm of the *ps* will be the dominant factor. Simple algorithms may result in base logic *pr* solutions; whereas, complicated *ps* algorithms may require one or more levels of programmed logic *pr's*.

CAD tools must provide various forms of synthesis and support functions for the potential variations of programmed logic *pr's*. These tools must include simulation and evaluation mechanisms so that insight and potential expert advice can be extracted for local *ps:pr* mappings as well as for the entire set of *ps's* with associated *pr's* of potential embedded system realizations.

ASAP is concerned with the design of *systems* rather than single chips. With increasing clock rates as a result of decreasing gate delays, packaging plays an increasingly important role in determining the performance of a computer system. Given performance goals for a system and packaging constraints such as limited real estate on a chip or printed circuit board, we want to find "optimal" system solutions.

ASAP is based on decoupling the specification of functions from the implementation of those functions. Asynchronous processing has been quite widely used in order to resolve problems of interfacing computing elements which operate at different rates of speed or for which a common clock cannot be established. Asynchronous processing decouples the implementation of a process from its communications.

5.7 Design Methodology and the Man-Machine Interface

Designs utilizing previously defined parts and designs undertaken by groups benefit from an asynchronous design style since the design efforts are decoupled. However, it is important that the different design teams agree on the external behavior of the various parts. Thus an important aspect of the future CAD systems being explored is to conveniently provide for various views of design that can be illustrated graphically as well as database philosophy which permits automatic updating of various design views. These aspects of the CAD system should also, in addition to specification and design, influence such issues as the testing and evaluation of various design alternatives.

5.8 Ongoing ASAP Projects

The following concrete project areas have been identified and are being actively pursued by members of CADLAB.

Process Description

Specification, Synthesis and Analysis

To support architectural specification, synthesis and analysis; a specification language has been proposed in an ongoing licenciate thesis project by Tony Larsson. The language will support synthesis, analysis and simulation tools. A set of calculus- hiding- binding and event reduction-rules are indented to form a framework for the design of higher level verification and synthesis tools. Enabling semantic preserving syntactic transformations, the rules will support deductive verification methods; however, exhaustive verification methods may also be tractable if hiding and binding rules are used to prune a design description.

Simulation

As part of the overall goal of studying architectures and silicon compilers, this project aims at a detailed description of the main components in the ASAP architectural strategy. A related goal is to study the various trade-offs a designer can consider when designing an asynchronous design based on these components. These trade-offs are studied as an attempt to uncover which sub-parts of components must be treated with the full generality of asynchronous communication and which may be treated via simpler direct synchronous communications that can be compiled out as isosynchronous regions in specific instances of the ASAP architecture. Finally, it is also a study of the programming language OCCAM and its usability for this type of application. A register level simulator for a family of architectures has been produced.

Optimization and Partitioning

This subproject of ASAP has been concentrating on the problem of how to optimize and partition a design during the synthesis process of VLSI systems. For this purpose, a unified design representation model, called extended timed Petri net (ETPN), has been developed. This design representation consists of separated, but related, models of control and data part. It allows formal manipulation of the design space and different optimization trade-offs between performance and cost. Partitioning of systems into submodules is provided both on the data part and on the control part, producing a set of pairs of corresponding data subparts and control subparts. As such, asynchronous operation of the designed systems as well as physical distribution of the

modules is possible. The use of such a formal representation model also leads to the effective use of CAD and automatic tools in the synthesis process and the possibility of verifying some aspects of a design before it is completed. CAMAD, an integrated design aid system, has been partially developed based on the ETPN model.

Pipeline Extraction

Pipelining is a fundamental technique in computer design, but since it today usually is a manual process, it is prone to design errors. This project tries to find a way of describing a system so that extraction of "pipelinable" parts can be done automatically. One needs to describe both the communication between the pipeline stages, as well as the computations performed, i.e., formalisms for process definition also need to be developed.

A pipeline can be described in terms of processes (ps's) and processors (pr's). There are two fundamental views:

1. The computation stages are processes, exchanging data.
2. The computations are processes, moving over the stages.

The first view facilitates description of communication, while the second simplifies description of the state of the computations. This project is trying to combine these features into a single view, and then use this description as the basis for a CAD tool for designing pipelined systems.

Representation and Reasoning

To support the ASAP architecture approach of VLSI System Design, knowledge and design representation should be such that architectural decisions may be reasoned about both manually and automatically.

The Representation and Reasoning subproject of ASAP aims at embedding the ASAP methodology into AI based tools, such as Design Assistants (Expert Systems), on different levels and phases in the design project.

One particular problem is the representation of the design itself, the design task, and the design knowledge. A number of international research groups are very active in this area, e.g. PALLADIO design environment at Stanford University COMPUTER, Dec. 1983, pp. 41-56 and the Knowledge-based Design System at Carnegie-Mellon University.

Through the use of design representation in the form of dependency networks, it is possible to reason about the interaction between subparts and their interrelation so that partitioning is possible. Thus support can be provided to the partitioning task in the Optimization and Partitioning subproject of ASAP.

The Representation and Reasoning project is currently in the state of investigating these issues and the different AI based strategies for Design Assistants and Automation.

Now that we have considered the general course of research activities, let us highlight the progress mode during 1985.

5.9 Progress During 1985

The CADLAB group made several important advances on the ASAP project during 1985. This has resulted in the completion of two licentiate theses and significant progress on a third licentiate to be presented in the spring of 1986. Further, four papers have been accepted for publication and presentation and two more have been submitted for conference publication.

We feel that we have made progress in all of the areas mentioned in the previous section. However, based upon the two licentiate theses, we can identify some more specific progress.

Via the licentiate of Peng, we have an experimental CAD framework for further research on many ASAP issues. Peng has developed a "granularity" independent view of control and processing activities that can be utilized as a basis for optimization and partitioning algorithms (the main contribution of his thesis). The highlights of this work are presented in section 5.10.

Via the licentiate of Fagerström, we have a proposed model for the asynchronous view of design that has been put forth as an ASAP goal. Further, a simulator for the model has been expressed in the OCCAM language thus providing a simulator for the model. In the process, a deeper understanding of the advantages of and problems with OCCAM has been developed. The highlights of this work are presented in section 5.11

In the areas of CAD database design, the work performed by a previous guest researcher Piotr Siemienski has been spread to other groups in Sweden, particularly the NMP CAD project in Kista. The ideas expressed by Siemienski about methods of maintaining design consistency will undoubtedly be useful in the Kista project, as well as influencing our further work on Design Methodology and Man-Machine Interface issues. Siemienski, who will submit his doctoral thesis on this subject in Poland in the spring of 1986, will be invited to join the NMP CAD project in 1986/87.

5.10
Steps Towards the Formalization of Designing VLSI Systems

The design of VLSI systems has been, by and large, the province of skilled engineers; recently, however, automatic and computer-aided tools have been and are being developed to accomplish some aspects of these complex design tasks, making the province of VLSI available to larger groups of designers. The present research is an attempt to formalize the design process as a sequence of semantics-preserving transformations such that automatic synthesis of VLSI systems from a program-like behavioral description into a structural description is possible. The produced structural description may then also be partitioned into several modules with well-defined interfaces.

The proposed strategy is based on a formal design representation model derived from timed Petri net and consisting of separated, but related, models of control and data part. One small example of such design representation is illustrated in Fig. 5.2. Petri nets are used here to represent the control flow, which permits explicit expression of concurrency and parallelism. Such a control structure can be later transformed into control sequencing realizations including a microprogram, or several microprograms. We have also developed a set of transformation algorithms to manipulate both the data part and control structure so as to obtain optimal designs. One of the important properties of such model is that it can be represented graphically; therefore it is possible to develop a graphic design tool based the present approach.

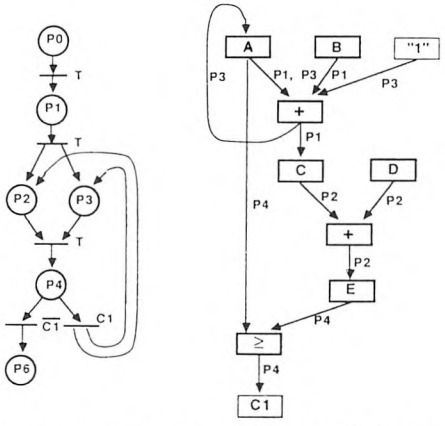


Figure 5.2. *An Example of the Design Representation.*

The data part of this design representation consists of data nodes which can be in different levels of granularity. Therefore, the representation can directly supports a hierarchical design approach. Further, partitioning of systems into submodules is provided both on the data part and on the control part, which produces a set of pairs of corresponding data subparts and control subparts. As

such, it allows potential asynchronous operation of the designed systems as well as physical distribution of the modules. Each of the modules can also be implemented independently, thus reducing the complexity of design to a manageable scale.

In the present approach, the control/data path allocation and module partitioning problem is formalized as an optimization problem. This differs from previous approaches where *ad hoc* algorithms and predefined implementation structures are explicitly or implicitly used, and where a centralized control strategy is assumed. To solve the optimization problem, a set of design space exploration strategies and heuristic algorithms have been developed.

Based on the present research, CAMAD, an integrated design aid system, has been partially developed. The overview of the CAMAD is illustrated in Fig. 5.3. Our ultimate goal is to construct a design environment in which VLSI systems can be first specified by its high level behavioral description without regard to the detailed implementation or packaging of the modules. This behavioral description will first be transformed into a data flow representation and then into the proposed intermediate representation. This design representation is then manipulated by a set of semantics-preserving transformation algorithms, which collapse the possible data elements or control elements to reflect the decision to share hardware resources. The module partitioner, on the other hand, will help the designers to choose a partitioning which divides the system into submodules.

Finally, the problem of how to automatically design control structures from the Petri net description has been studied by Krzysztof Kuchcinski and Zebo Peng. A set of algorithms have been implemented as part of CAMAD which analyzes the properties of a control part and creates a microprogram to implement the control function.

5.11
Simulation and Evaluation of an ASAP Architecture

ASAP, is an architectural strategy based on function unit composition via communication. There can be various implementations of this strategy. The components of the architectural strategy are introduced via a Fig. 5.4:

(The last units, the bus and the bus interface, are used only as a glue to connect system components.) Furthermore, all of the elements of the strategy do not have to be present in every design; simplification is possible. An ASAP system can consist of a number of system elements, among these are ports, function units, buses and bus interfaces, schedulers and gateways. A short description of each type of component is given.

Port: This type of module provides an abstraction of a function. By function abstraction we mean that a port "abstracts out" specific behaviours from a

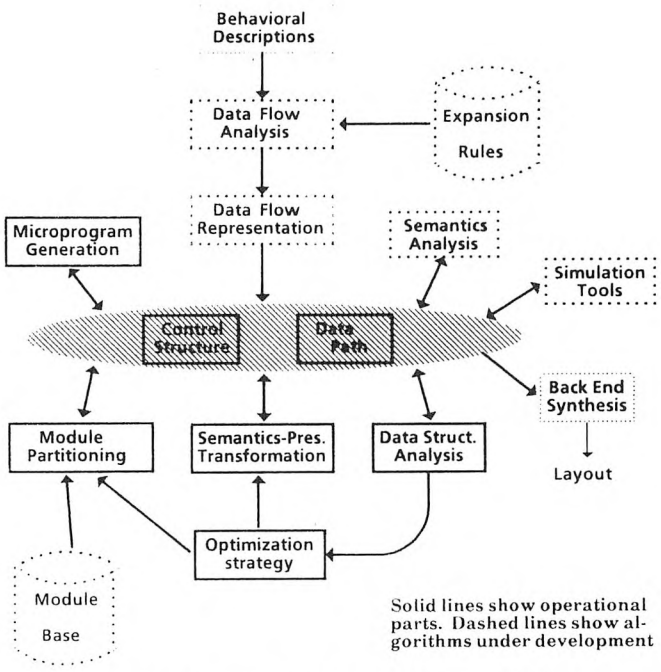


Figure 5.3. *Overview of the CAMAD Design Aid System.*

particular implementation of a function. We set up virtual circuits between ports by storing pointers in special registers associated with a port. Each port consists of a port controller, a queue for buffering and an output pointer used for virtual circuits. Data is thus sent to the port via some communication structure (e.g. bus 1 in the figure), processed by the function unit and finally sent to the port which the output pointer points at.

Control unit: When switching contexts, an explicit machine state is needed; therefore, the need for a control unit arises. A program executing in the control unit is responsible for initiating and setting up virtual circuits and sequencing a computation by transmitting values. When a computation has terminated, a value is ultimately stored back into the control unit, a state which we call a "clean point". It is when a clean point is reached that there is no outstanding computation and the control unit can store the necessary state and switch to a new job.

Scheduler/resource manager: This unit is responsible for scheduling processes onto the control unit with which it communicates. The scheduler also acts as a resource manager since it is possible to create different pipelines at different times; thus, there is a possibility of deadlock.

Function unit: These units implements functions, for example a FFT. The unit is intended to be an autonomous module which, when supplied with data,

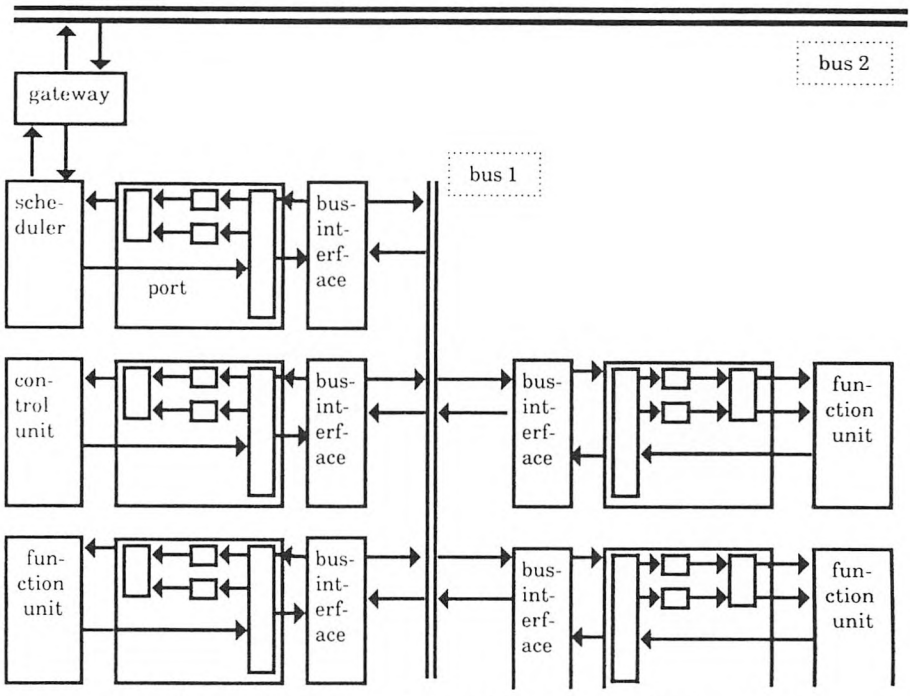


Figure 5.4 *Component of the ASAP Architectural Strategy.*

functions independently of the rest of the system.

Gateway: Gateways are used to connect "computing engines" consisting of the above mentioned system components. Each such engine is a self-contained system capable of useful computations. Gateways provide various services such as naming services. It is thus very natural to regard a computing engine connected to a gateway as a single component. A silicon compiler can, when partioning a system, use this component for implementing self-contained sub-parts of a design.

The system shown in the figure has been simulated in the language OCCAM, a language where parallel processes communicate by sending values on channels.

5.12 Cooperation With Other Groups

CADLAB has cooperated with the Computer Systems Laboratory at Uppsala as well as with Piotr Debinski in Gothenburg in applying the formal design techniques developed for description of control and processing structures and communications protocols to the description of integrated circuits. The CADLAB group, in November, visited the new Swedish Institute for Applied Computer Science (SICS) in Kista and presented the ASAP project. We expect to follow their progress and make our results available to various SICS

projects. Further, we have had contacts with Gunnar Carlstedt of HYLAB AB concerning the exchange of ideas for VLSI design.

5.13 Industrial Significance

Many VLSI experts have come to the conclusion that the possibility to design and implement complex systems composed of heterogeneous processes will require wide spread use of asynchronous control strategies (see "Logic Designers Toss Out the Clock", Electronics December 9, 1985). Thus the asynchronous approach which has for many years been a premise for the architectural research and development of Professor Lawson is becoming wide spread. The industrial relevance of this research for future complex system construction is rapidly increasing.

During 1985, CADLAB has met with the defense products group at Ericsson Radio Systems (Kista). After studying the requirements for future on-board computer systems for the JAS airplane, it was determined that the ideas being studied and experimentally evaluated in ASAP can very likely provide a design approach for future products in this area where the control and potential interaction of many complex heterogeneous processes are "modus operandi" prerequisites.

Mr. Rob Ragan-Kelly, the architect of Pyramid Computer Products has expressed an interest in exploring asynchronous methods as a functional complement of commercially available Reduced Instruction Set Computer (RISC) architectures. Mr Ragan-Kelly has preliminary plans to join CADLAB in 1986 in order to further explore asynchronous control and to pursue a doctoral degree.

5.14 Other Related Activities

As a result of Professor Lawsons assistance to the Prime Minister of Malaysia Dr. Mahathir in planing a National Microelectronic Programme, a new institute MIMOS (Malaysian Institute of Microelectronic Systems) was established and inaugurated during 1985. Professor Lawson presented a paper on the project at the ERSA (Economic Relations between Scandinavia and ASEAN) conference at Stockholms University in October 1985. Further contacts between MIMOS and various Swedish Institutes and the Ericsson Corporation are being pursued.

5.15 Personnel

Professor Harold W. Lawson Jr., Ph.D.
J. Bryan Lyles, Ph.D.

Britt-Marie Ahlenback, secr.
Johan Fagerstrom, Tech.Lic.
Bjorn Fjellborg, MSE
Tony Larsson, MSE
Mikael Patel, Tech.Lic.
Zebo Peng, Tech.Lic.
Krzysztof Kuchcinski, Ph.D. (Guest Researcher September-December)

Professor Harold Lawson has been acting laboratory leader from July 1985 after the departure of Dr. Bryan Lyles. As of January 1986, Michael Patel will actively participate in the leadership of CADLAB. Further, we are negotiating with a new potential leader for the group. We expect that 2 new doctoral students will be added to the group during 1986.

5.16 Licentiate Theses

Vojin Plavsic, Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory.

Arne Jönsson and Mikael Patel, An Interactive Flowcharting Technique for Communicating and Realizing Algorithms.

Zebo Peng, Steps Towards the Formalization of Designing VLSI Systems.

Johan Fagerström, Simulation and Evaluation of an Architecture based on Asynchronous Processes.

6.

RKLLAB

The Laboratory for Representation of Knowledge in Logic

Erik Sandewall
Professor of computer science

The area of interest for RKLLAB is *theoretical aspects of knowledge based systems.* The activity of "knowledge engineering", or the design of expert systems and other knowledge based systems, is at this time a rather *ad hoc* activity, but there seems to be good opportunity to apply and extend logic (and discrete mathematics) so as to strengthen the theoretical basis for knowledge engineering. It is the objective of RKLLAB to contribute in that respect.

6.1 Researchers and Projects.

RKLLAB was formed in January, 1985 from the office systems project previously in ASLAB and the reason maintenance systems project previously in AILAB. During the previous years, the office systems project had gradually extended into formal specification aspects and knowledge engineering aspects of office systems. In particular it started studying the topic of non-monotonic logic, which of course is closely related to reason maintenance. This explains the reason for the reorganization, and the name of the new laboratory.

The work in RKLLAB is mainly supported by STU, The Swedish Board for Technical Development.

6.1.1 Activities.

The activities in RKLLAB during 1985 have been in the following areas:

Non-standard logics, in particular:
- non-monotonic logic (written NML in the table below)
- reason maintenance (RM)
- fuzzy logic (FL)

Work on *office systems*, in particular:
- theories of office software (ThOS)
- office systems components (equipment and software) (OSC)
- startup of a new project, called LINCKS

Work on *representation of knowledge about machinery*, particularly the representation and analysis of action-plans that involve concurrent actions with a duration in time. (RKM)

6.1.2 Laboratory Members

The following researchers have been members of RKLLAB during 1985:

	Activities
Laboratory leadership:	
Erik Sandewall	NML, RKM
Lillemor Wallgren, secr.	
Project leaders and senior graduate students:	
Dimiter Driankov	FL, LINCKS
Jim Goodwin	RM, RKM
Lin Padgham	LINCKS
Ralph Rönnquist	ThOS, LINCKS
Graduate students and masters thesis students:	
Johan Andersson	OSC, LINCKS
Peter Haneclou	RM, LINCKS
Johan Hultman	RKM
Per Leander	RKM
Stefan Wrammerfors	ThOS
Technical services:	
Jan Axing	OSC
Leif Finmo	OSC
Arne Fäldt	OSC

6.1.3 Main current achievements.

The major achievements during 1985 have been:

a) a characterization of non-monotonic reasoning in terms of a four-valued logic, and an application to inheritance with exception (Sandewall)

b) a fast algorithm for calculating some admissible probability assignment for propositions, based on given probabilities for some logically related propositions (Driankov)

c) a set of linguistically motivated estimates for belief intervals, which are needed because for stability reasons, conventional numerical estimates of probabilities appear to be "too precise" (Driankov)

d) a characterization of a relation over binary networks which can be seen as the relation of "containing more information" (Rönnquist)

e) a characterization of the relationship between schema and relations in relational data bases, using binary networks for expressing the schema (Rönnquist)

and (not yet published):

f) a systematic treatment of meta level reasoning in the same framework as object level reason maintenance (Goodwin)

g) an analysis of recurring cycles composed of actions with duration and parallellism (Sandewall et al)

6.2 Non Standard Logics

The term "non-standard logic" is popularly used for "everything except first order predicate logic". In a more positive vein, we are interested in two kinds of extension over the "standard":

- special semantics, such as "fuzzy" semantics and "multiple world" semantics;

- special reasoning mechanisms, such as non-monotonic reasoning (which is able to backtrack when it reaches an inconsistency), and reason-maintenance mechanisms (where inference steps are stored in a data base, in such a way that the property of being a theorem can be turned on and off as logical support arises and is lost).

There is ample evidence that such extensions are necessary for the further development of knowledge-based systems.

6.2.1 Non Monotonic Logic

Ordinary logic is monotonic in the sense that if A is a set of propositions, and Th(A) is the set of theorems that are derivable from A, and A is a subset of A', then Th(A) is a subset of Th(A'). In other words, the function Th is an (increasing) monotonic function of its single argument. Non-monotonic logic is the popular term for variants of logic which violate that property, and is widely recognized to be of crucial importance for A.I.

The standard way whereby non-monotonicity arises is through the use of the *Unless* operator (first introduced in [Sandewall 1972]), where one may write for example

if	A
and	Unless B
then	C

meaning that if A has been proven, and if B can not be proven, then C may be obtained as a theorem. (References are at the end of the RKL-lab section).

Non-monotonic logic has turned out to be an enigmatic problem. In ref. 9, Erik Sandewall analyzed non-monotonic reasoning in terms of a four-valued logic, with the four "truth-values" *true*, *false*, *undefined*, and *contradiction.* Very briefly, one of the results in the paper was that one may computationally re-interpret a rule using the Unless operator, as, in the example: if A has been proven, and if B has not yet been proven, then consider C as proven and B as disproven (i.e. "not B" is added to the set of assumptions, and is treated just as if it had been proven). However, one must take one precaution: if a contradiction arises later on during the derivation process, for example of the form "B and not B", then one must be prepared to backtrack.

More technically, the paper shows that the following property of monotonic logic:

> if A is a set of axioms, and A' which is a superset of A is a *minimal* fixpoint for the operator defined by the inference rules, then there exists a derivation from A to A'

does not apply for non-monotonic logic of Reiter's "default logic" variety, *but* if the minimality criterion is strengthened so that A' is furthermore an *approachable* fixpoint, in a sense defined in the paper, then a derivation exists.

In a later paper (ref 10), these results have been applied to a particular but very common case, namely inheritance in posets (e.g. classification structures) where exceptions are allowed. For example, one may wish to state in a biological data base that "elephants are gray" but also that "albino elephants are white, constituting an exception", and there may be exceptions from the exceptions etc. The paper shows how the "obvious" deductions with such exceptions can be formulated as a straightforward special case of the non-monotonic logic developed in the previous paper.

6.2.2 Reason Maintenance

Work on reason maintenance has primarily been done by Jim Goodwin, who is now finishing his Ph.D. thesis. This work was described in last year's progress report, and we refer to it and to (ref. 4).

Also during 1985, Peter Haneclou has started a project to express reason maintenance operations in terms of Sandewall's four-valued logic as described in the previous section.

6.2.3 Fuzzy Logic

In fuzzy logic, the classical truth-values 0 and 1 (false and true) are replaced by number between 0 and 1, or other entities which provide a measure of certainty level. Dimiter Driankov is doing research in this area.

One paper (ref. 1) considers the case where the truth-values are replaced by probabilities between 0 and 1. Suppose probabilities of some propositions are known, and probabilities of some other propositions (usually containing the same atomic propositions as the given ones do) are to be calculated. This problem is in general under-determined, and algorithms have been proposed, in the research literature, for calculating a "most likely" probability for the new proposition(s). Such algorithms are however so resource consuming that they are probably not practical. Mr. Driankov has developed an algorithm which merely calculates *one admissible* probability assignment, i.e. one which is consistent with the given probabilities, but which can do so fairly rapidly, namely by solving a set of linear inequalities. Also, if there is no admissible probability assignment, meaning that the given probabilities are inconsistent, then the algorithm indicates which subset of the given probabilities is culpable, which of course is of significant value for "debugging" the set.

In some other papers (ref. 2-3), the truth values were instead replaced by *belief intervals* which are pairs of probabilities, and therefore represent some segment of the interval [0,1]. A calculus of belief intervals was proposed where disjunctive and conjunctive operators are based on the Dubois family of parameterized T-norms and T-conorms, where the parameter was interpreted as a measure of the independence between the respective evidence.

One of the problems which arises in such systems is that small changes in the numerical values given by the user, may significantly change the derived values for other propositions, and in particular, it may change admissible situations to unadmissible ones. This suggests that numerical values are "too precise", and that cruder estimates might be desirable for stability reasons. Mr. Driankov introduces a set of nine linguistically based estimates - a kind of nine-valued logic - and analyzes under which conditions the resulting system is "well behaved" with respect to introducing contradictory belief intervals.

6.3 Office systems

Our work on office systems has the following long-term goals:

- development of appropriate representations of office knowledge, i.e. such knowledge as is processed in office work;

- development of models for "office procedures", or representation of knowledge *about* office work;

- development of appropriate designs for knowledge based office systems, particularly through experimental implementations;

- development of a theory that (in an empirical sense) accounts for observed phenomena in conventional office software (e.g. editors, formatters).

The first three of these goals are obviously in line with the overall focus of RKLLAB; the fourth goal is a necessary complement in order to obtain a full understanding of office systems. Besides these research goals, software, and sometimes even equipment, is developed if that serves to advance us towards the research goals.

Some of the development during previous years have been:

- development of a theory of information flow, which provides a good model for albeit fairly simple forms of office procedures.

- development of a theory of information management systems, or IMS theory, which provides a basis for formal characterization of office software. In particular, the semantics of editing operations in hierarchical structures was defined using IMS theory.

Both of these developments have been accompanied by experimental implementations. The objectives for 1985 were to approach "representations of office knowledge", in the above sense; also to extend the approach for "office procedures" so as to deal with much more flexible routines than in the information flow model; and finally to pursue the theoretical work. Experimental implementations should continue to play an important role.

6.3.1 Theories of office software.

The effort during the reported period was based on the previously developed IMS theory. The key idea in this theory is to view a data structure or data base as a simple binary network, with nodes, arcs from nodes to nodes, and arcs from nodes to attribute values. Such networks are viewed as interpretations for propositions in a variety of three-valued first order logic (with undefined as a truth-value because interpretations may be partial). At the same time, algebraic operations are defined for composing networks into

larger networks. There is of course an analogy with the propositional and algebraic ways of treating relational data bases.

The main framework as well as the rationale for it were outlined in a set of lecture notes (ref. 8). For earlier work on IMS theory, see also last year's progress report.

This line of research has been continued during 1985 as follows:

a) **Relational Algebra in IMS Theory** (Ralph Rönnquist, ref. 5).

The relational database model, particularly the relational algebra, is given an interpretation in terms of an underlying semantic network, which is then viewed as a kind of "deep structure" for the relation. The relation is understood as a composition of a categorizing pattern (the relation schema) together with augmentations as given by the tuples constituting the relation contents. The network database is thought of as containing each schema-tuple combination explicitly, so that the schema part is a recurrent pattern. In light of the relational view however, a set of such schema-tuple pairs is abstracted into a schema-relation pair by allowing the combination to distribute over set formation.

The operations of relational algebra are then extended to schema-relation aggregates, and are defined in terms of network operations. It is shown that each operation on relations has a corresponding operation on schemas. The schema operation yields a new schema, the relation operation yields a new relation, and the combination of these forms a schema-relation pair that is the outcome of the extended operation.

b) **The Information Lattice of Networks Used for Knowledge Representation** (Ralph Rönnquist, ref. 6).

Networks are viewed as information containers, so that a network contains as much information as can be interpreted from it. One network then contains more information than another if there is a way to increase the information contents from the latter so as to obtain the former.

Addition of constituents, such as arcs and nodes, to a network is one way to increase its information contents. In other words, a network contains at least as much information as any of its sub-structures.

Another, and perhaps less obvious way to increase the information in a network is by identification of nodes. When a network is created for expressing some amount of information, it may happen that the same "thing" is represented by two or more, distinct nodes in the network. Typically, this happens if the information is given piece-wise and a "thing" is referred to in different ways. The addition of information that identifies two nodes as the same one, is an information increase. Operationally, it results in a network

contraction, i.e. the identified nodes are brought together into a single node while retaining all their arcs.

Taken together, the extension and contraction operations form a transitive relation between networks. It defines a lattice of equivalence classes of networks. Each such class consists of networks which are equivalent with respect to information contents. It is shown that we can choose a canonical element in each class, such that all other networks within the same class are extensions of the canonical one, i.e. can be obtained by adding nodes and arcs. Therefore, the canonical element is the "smallest" one among the networks that contain the same information. In some special cases, the structure of the lattice has been identified in detail.

c) Formal definition of formatting operations (Stefan Wrammerfors)

An idealized text formatter is defined using two mechanisms:
- editing operations on attributed trees, as defined in the IMS theory and as implemented in the IM4 system (see below);
- attribute propagation, as used e.g. in attribute grammars.

This work is still in progress.

6.3.2 The case for non-integrated workstations

Before we proceed to descriptions of the actual implementation experiments, let us remark on a choice of strategy that underlies those experiments, namely the case for non-integrated workstations.

The trend during the last decade has been towards *integrated* office workstations, i.e. the single computing system which provides high quality services in several dimensions: strong computing power, large working memory, high resolution screen, etc. We believe that the time is now ripe to work on the opposite concept. The "workstation" should be seen as a place where a person works, rather than as a computer box, and it should be explicitly recognized that different types of equipment, with different levels of capability, will co-exist in workstations. The challenge of course is to arrange the total system so that the best possible use can be made of *both* the more capable and the less capable computer systems.

The emergence of portable computers is one of the strong reasons for non-integrated workstations. Portable computers are very convenient to have, especially for people who travel much. However, weight is a very significant convenience factor in portable computers. It is likely that users will be happy to stay with the classical, 24x80 character screen type services, if that can help to cut down on weight. But when the portable computer is brought back to the

office, the work that started in the portable must be able to continue on higher quality, stationary computers in the workstation.

Also, many types of offices may continue to use special purpose computer equipment, related to the specific business at hand. Again, general purpose equipment must be able co-exist peacefully and constructively with the special purpose equipment.

Furthermore, the present uncertainty about the medical effects of working with CRT screens, may make it necessary to increasingly use other display technologies where high resolution is more difficult to achieve.

For these reasons we have started to address the issue of how to design office systems in a non-integrated way, so that there are several computing systems which in some sense "know about" each others' capabilities and are able to relate to them. We also wish these relationships between participating computing systems to be visible for the user, and to be expressed in terms of high level concepts.

6.3.3 Office Systems Components: the IM4 project

The IM4 project was started in 1983 as an implementation experiment of an office information system based on (an early variant of) IMS theory. The core of the system is a data base organized as an attributed tree structure, and an interactive editor with powerful macro facilities for operating on the data base. The original plan called for several lines of growth from that core: "high level", building on the facilities in the core for implementing high-level services; "low level" providing interface to operating services (such as forms, windows, and computer mail transmission) using the attribute mechanisms; and "towards theory" by using attribute propagation models for various types of office systems software (such as formatters).

During 1984 and the first half of 1985, the IM4 project focussed mostly on the "low level" direction of growth, and it therefore resulted in a library of office systems components, including some associated hardware. The project of Stefan Wrammerfors (previous section) represents work "towards theory" during 1985. We returned to the "high level" direction of work through the LINCKS project that was started in October (next section).

In line with the design hypothesis of non-integrated workstations, the IM4 core system was viewed as a supervisor which would direct a number of lower level processes, on several processors. The core system was implemented in Lisp (Elisp for DEC-20), and the subordinate processes would be both separate DEC-20 processes, and processes in M68000 series computers, particularly for display purposes (both on screens and on paper). It was also intended that the entire system (including the services in the subordinate processes and computers, and the protocols) should eventually be formally characterized as a part of the "towards theory" line of work.

The following are the major results of work on IM4 during 1985:

- the (previously implemented) protocol for communication with display processors was extended to allow incremental update of the screen, using a window concept;

- the protocol was also made available using function calls in Elisp;

- the existing 'Aform' text formatter, implemented in Pascal, was modified so that it allows incremental formatting. This was done by establishing a link from Elisp to Aform whereby Elisp can sense and set the parameter settings of the formatter (current left margin setting, current spacing, etc.), and whereby Elisp can also request Aform to format a segment of text relative to the current setting of the parameters, and to exit non-destructively from the formatting process when the text segment is done;

- in order to implement the previous, a general purpose interface between Elisp and Pascal was implemented;

- all documentation about IM4 code was stored in the IM4 data base itself, where it was closely intermeshed with the actual program code. This was an interesting experiment, as a step towards easier maintenance and re-organization of documentations.

These activities in the IM4 project did not in themselves result in an operational system, but they prepared the way for the next project and provided a number of tools for it.

6.3.4 Experimental high level system: the LINCKS project

During the period of October - December, the "high level" project was initialized and its goals were defined. The new project is called LINCKS, for Linköping Intelligent Knowledge Communication System, and is led by Lin Padgham. Since the project has just started, and the goal statement for the project was confirmed in mid-December, we will only give a very brief outline here.

LINCKS takes as its primary goal to support *communication*, in the sense of creating, transmitting, and receiving information. For example, if communication takes place using a book, then the authoring, the publishing, and the reading activities represent those three stages. Each stage may consist of a number of "smaller" communication tasks. For example, publishing a book may involve a number of letters between the author and the publisher.

The goal of the LINCKS project is now to build a system which supports communication in such a way that some of the low-level communication acts can be automated, using the technology of knowledge-based systems. One of the requirements is then that the goal structure, where higher level actions are

decomposed into lower level actions, is stored in the computer system. Only then can the built-in "expert systems" have a chance to "understand" the context of what they are supposed to do, and apply whatever common sense they have been endowed with.

LINCKS uses a multi-level structure for office knowledge, where one level is a "notecard" database, i.e. a database where each object may contain attributes, and where typical objects have a "text content" (also represented as an attribute) whose length is one or a few paragraphs. A longer text is constructed as a set of notecards, where links between them are represented using some of the other attributes. A message in computer mail may be a single notecard; one user's mail file will be a structured set of notecards.

Such a notecard structure strikes a balance between two possibly conflicting goals: the representation structure should be so rich that the user can conveniently express himself in it, and at the same time the structure should be so uniform and clear that the "expert system" type software in LINCKS is able to manipulate the structure.

6.4 Representation of Knowledge about Machinery

Besides the office application, we have also started to work on the application area of "machinery", with the following concrete examples in mind:

- machines that do manufacturing operations automatically (NC-machines, industrial robots, etc)
- machines that move around - unmanned vehicles
- machines that lift cargoes (cranes, trucks, etc)

We are particularly interested in one aspect of machinery, namely that in order to form an adequate description of its behavior we must use action structures, in the following sense: An action structure is a generalization of an action sequence, where actions may take place simultaneously as well as sequentially, and where each action lasts for some time (not just momentarily). A good formal way of dealing with such action structures was the first priority. Based on theoretical work in early 1985, an experimental software tool called HIAS (for "handlingsplanering i intelligenta autonoma system", i.e. action planning in intelligent autonomous systems) has been implemented by Jim Goodwin and Per Leander. (The publication for this work is still in the pipeline).

The problem of modelling such action structures is shared with many other types of applications, and occurs also in the office systems application, particularly when portable computers are used. There are however some other aspects of machinery which are also important, and more specific: the need for

handling geometry, and the need for integrating the action-plan with the sensorics and the motorics that occurs when each elementary action is performed. During 1986 we are also beginning to address these issues.

6.5 References

The following are those RKLLAB publications referenced in the text above. For the full set of publications, please refer to the appendix of this progress report.

1. Dimiter Driankov: *Inference with consistent probabilities in expert systems.*

2. Dimiter Driankov: *A calculus for belief-intervals representing uncertainty.* to appear in *Proc of the Int. Conf. on Information Processing and Management of Uncertainty,* Paris, June 1986.

3. Dimiter Driankov: *A calculus with verbally defined belief-intervals.*

4. Jim Goodwin: *A Process Theory of Non-Monotonic Inference.* Proceedings of IJCAI-85.

5. Ralph Rönnquist: *Relational Algebra in I.M.S. Theory.* Report LiTH-IDA-R-85-06.

6. Ralph Rönnquist: *The Information Lattice of Networks Used for Knowledge Representation.* Report LiTH-IDA-R-86-02.

7. Erik Sandewall: *An approach to the frame problem, and its implementation.* In Meltzer and Michie (eds): *Machine Intelligence 7.* Wiley, New York, 1972.

8. Erik Sandewall: *Theory of Information Management Systems.* Report LiTH-IDA-R-83-03.

9. Erik Sandewall: *A Functional Approach to Non-Monotonic Logic.* Proceedings of IJCAI -85, and *Computational Intelligence,* Vol. 1, No. 2, pp. 80-88.

10. Erik Sandewall: *Non Monotonic Inference Rules for Inheritance with Exception.* To appear in *Proceedings of the IEEE,* special issue on Knowledge Representation.

7.

The Group for Logic Programming

Jan Maluszynski

7.1 Introduction

The Group for Logic Programming was created in spring 1985 as a result of division of the former Group for Theoretical Computer Science into two independent research groups. The research program is oriented towards the development of more efficient methods of execution of logic programs. The research concentrates on the foundations of logic programming systems and on the relation of logic programming to other computational paradigms.

An important objective of the group is also to contribute to the research activities of the other laboratories by offering courses and seminars on logic programming, theory of programming and formal language theory.

7.2 Researchers and Research Activities

7.2.1 Personnel and External Researchers

The following persons were involved in the research activities of the group:

Jan Maluszynski , Ph.D. *group leader*
Wlodzimierz Drabent, Ph. D. *visiting researcher*

Some of the research was done in external cooperation with:

The work in the Logic Programming Group is mainly supported by STU, The Swedish Board for Technical Development.

Piotr Dembinski at Chalmers
Pierre Deransart at INRIA, France
Jan Komorowski at Aiken Computation Lab., Harvard

The following undergraduate students contributed to the research:

Håkan Jakobsson
Simin Nadjm-Tehrani
Ulf Nilsson

The main research activity concentrated around the project *Research in Efficiency of Logic Programming"* funded by STU-F (grant 85-3166).

7.2.2 Background

Research in logic programming at IDA was initiated by Jan Komorowski who in his Ph.D. thesis specified an abstract Prolog machine and described some experiments with partial evaluation of logic programs. In 1982 Jan Maluszynski joined the department and investigated relations between van Wijngaarden two-level grammars and logic programs. It was shown that two-level grammars can be considered logic programs where many-sorted terms are used instead of the terms of the Herbrand universe. A continuation of this research was a joint project with Pierre Deransart aiming at a formal comparison of attribute grammars and logic programs.

7.2.3 Research in Efficiency of Logic Programming

We are searching for methods of static analysis and optimization of logic programs. The suggested approach is based on the formal comparison of Attribute Grammars and logic programs by Deransart and Maluszynski. We specifically focus on possible transformation of a logic program into an equivalent Attribute Grammar. Such a transformation makes it possible to use the dependency relation of the Attribute Grammar for data-flow analysis of the logic program. Furthermore, it opens for application of attribute evaluation techniques for execution of logic program. A long range objective is development of a logic programming environment implementing these ideas.

Preliminary Results

We recall first the motivation for our work, as presented in the original proposal.

The attractive characteristics of logic programs is their declarative semantics. The logic programmer provides the logic part of a problem while the interpreter is supposed to supply the control. This approach has proven very convenient for writing small executable specifications but suffers seriously from inefficient execution. Since the interpreter which supplies the control component has to be very general it is unlikely to be very efficient. The

motivation for our work is the conviction that there is no need to pay the cost of a very general procedure for executing specialized programs. Therefore, we are searching for properties of logic programs that allow more efficient execution, and for method of static analysis which check whether a given program has such properties. Our approach assumes that the declarative reading of a logic program cannot be sacrificed. We share the opinion expressed by Mellish in the recently published paper that many logic programs "are not radically different in kind from programs written in conventional languages. For those programs it should be possible for a compiler to produce code of similar efficiency to that of other compilers." For this it is necessary to develop methods of static analysis of logic programs. Mellish suggests the use of abstract interpretation techniques. This assumes sequential execution with a fixed control strategy. However, more flexible control strategies may be worthwhile for some applications, e.g. coroutining. Furthermore, AND-parallel execution is sometimes advocated as a way of improving efficiency. We are searching for methods of analysis of logic programs applicable also for these execution models. Since the execution of a logic program relies on the proof-theoretic semantics, what we really need is a proof technique that makes it possible to prove properties of the trees constructed for a given program by a resolution procedure. We suggest to adapt for this purpose the techniques for proving properties of attribute grammars discussed by Katayama and by Deransart.

The research in 1985 concentrated on foundations of our approach. The results are summarized below.

A formal comparison of logic programs and attribute grammars.

Logic programs are closely related to the attribute grammars defined by Knuth. This fact is sometimes pointed out in the literature without discussing the nature of the relationship between these formalisms. Logic programming proved to be useful for implementation of attribute specifications. Our objective was to give a more formal account of the relations between the formalisms to facilitate transfer of expertise. Research on this topic was completed and the results were published in:

Deransart,P. and Maluszynski,J.: Relating logic programs and attribute grammars *Journal of Logic Programming* 3, No. 2 (1985) 119-158

We now summarize some results of this paper.

Both attribute grammars and logic programs refer to similar notions of labeled trees. These are proof trees of logic programs and decorated parse trees of attribute grammars. It was shown that it is often possible to transform a logic program into an attribute grammar whose decorated trees are isomorphic to the proof trees of the program. A construction implementing such a transformation was described and discussed. This opens for application of proof techniques known for attribute grammars for static analysis of logic programs.

The paper discusses also some differences between the formalisms. It is an interesting question whether the features of attribute grammars which distinguish them from logic programs could be possibly used in logic

programming.

The most important differences between attribute grammars and logic programs can be summarized as follows:

Semantic domains

According to the model-theoretic semantics the semantic domain of a logic program is its Herbrand universe. However, this may be rather inconvenient, since this means that any data type used in an application should be in principle implemented by terms, while its operations should be expressed as predicates of the program. Prolog implementations introduce "evaluable predicates", e.g. arithmetic, which facilitate programming but cannot be described in terms of Herbrand models. On the other hand, attribute grammars may use arbitrary data types. These are, however, assumed to be specified and implemented outside the grammar. The grammar itself can be seen as a mechanism for combining such external specifications during a computational process. The operations of the external data types are referred to by the operator symbols used in the semantic rules of the grammar. To make possible a proper interaction between external specifications and the grammar some restrictions on the form of the semantic rules should be observed. This concerns a many-sorted type discipline and the structure of the semantic rules related to the splitting of attributes into "synthesized" and "inherited". This classification of attributes resembles to certain extent the idea of mode declarations for predicates of logic programs.

Construction of the tree

Proof trees of a logic program are constructed by resolution. The construction is based on unification. Construction of a decorated tree of an attribute grammar is in principle done in two phases (which may be merged in some implementations). First a skeleton parse tree is constructed for a given terminal string. For this a context-free parser is used. Then an evaluation procedure is called to decorate the tree.

Decoration of the tree

Partial proof trees of logic programs are constructed and updated during subsequent resolution steps. One label may be updated many times during different steps. Backtracking may also change labels. Decoration of a parse tree by attribute values is controlled by data flow. The value of each attribute position is determined by a semantic rule associated with this position. Formally a semantic rule is a term whose variables denote some attribute values. The term cannot be evaluated as long as the values of its variables are unknown. Evaluation of the term determines a value of the attribute which may enable evaluation of some other attributes. Thus, attribute grammars apply the concept of data dependency for controlling computations.

Some interesting subclasses of logic programs

The proof technique for attribute grammars has been exploited to formulate

sufficient conditions for some runtime properties of logic programs. This research is not yet completed. We plan to use the experience from this experiments to give a more comprehensive account of this proof technique and to search for more properties of logic programs which may be relevant for the efficiency of execution. The preliminary results are published in the following papers:

Maluszynski,J. and Komorowski,J.: Unification-free execution of logic programs, *1985 IEEE Symposium on Logic Programming*, Boston July 1985, IEEE Computer Society Press, 78-87

This paper gives a sufficient condition to replace runtime unification in a Prolog program by term matching. This seems to be important for efficiency since it opens for use of parallel term matching instead of unification. The class of programs satisfying this condition is non-trivial. The paper discusses also a methodology of using this condition as a programmer's tool.

Dembinski,P. and Maluszynski,J.: AND-parallelism with intelligent backtracking for annotated logic programs, *1985 IEEE Symposium on Logic Programming*, Boston July 1985, IEEE Computer Society Press, 29-38

This paper gives an improved scheme of AND-parallel interpretation for a class of logic programs with a particularly simple scheme of data-flow. The class corresponds to a well-known subclass of Attribute Grammars. The advantage of the scheme over well-known Conery's approach is that the data flow analysis can be performed in compile time. However, the class of the programs is restricted. Presently it is the subject of an experiment which aims at checking how often people write programs outside this class and what are the programming techniques which require violation of our restrictions.

Continuation of the Research

The preliminary results mentioned above give a theoretical framework for development of implementation principles for a logic programming system based on the concept of attribute evaluation. Development of such a system should be preceded by :

1. Implementation of algorithms for static analysis of logic programs. These include:
- *data-flow analysis;* checking whether a given program can be transformed into an attribute grammar of some special type;
- *unification analysis*; checking whether the program fulfills some conditions which allow to compile-out unification, to replace unification by one-way pattern matching, to avoid occur-check, etc.
- *mode analysis*; checking whether "evaluable predicates" and external functional procedures satisfy some conditions for the run-time instantiation of their arguments.

2. Implementation of a many-sorted type system for logic programs and algorithms for static type checking.

3. Implementation of a bottom-up parser for Definite Clause Grammars.

The group is currently engaged in the following activities:

- development of an improved model of parallel execution of logic programs (H. Jacobsson, J. Maluszynski). This is a continuation of the previous work.

- development of programs for static analysis of logic programs (W. Drabent, S. Nadjm-Tehrani). One of the aims is also to check what programming techniques are used in logic programming. Presently we have a program that checks whether a given Prolog program fulfills some restrictions concerning data flow upon which our scheme of AND-parallelism is based. An experiment with a sample of logic programs shows that many of them fulfill the restriction and discloses some programming techniques specific for logic programming. The results of the experiment are being prepared for publication.

- development of an alternative implementation of DCG's (Ulf Nilsson). The objective is to extend well-known table-driven bottom-up parsing techniques for context-free grammars for the case of DCG's. This should allow to deal with the case of left-recursive grammatical rules which cannot be handled by Prolog systems, and to avoid backtracking in parsing when the underlying context-free grammar of a DCG is a deterministic grammar.

7.2.4 Research in Theory of Programming

This was a continuation of the previous work of Wlodzimierz Drabent. An attempt to find another construction of domains for denotational semantics was undertaken. The traditional, so called P-omega, approach is not easily comprehensible. It seems that domain construction is a main obstacle in good understanding of denotational semantics. W. Drabent's work on this subject was connected with his previous work on building operational definitions which were close to denotational ones. The basic idea of the approach is to determine a domain element through its finite, "representable-in-computer", approximations. Domains are built from so called kits which are quasi-ordered sets of such approximations. Informally, the quasi-order reflects the "amount of information" carried by single kit elements. Domain operations and equations may be expressed in terms of kits what gives a gain in simplicity. The approach is similar to that known as Scott's information systems.

7.3 Contacts within the Department

An important objective of the Group for Logic Programming is to contribute to the activities of the other laboratories by organizing courses presenting mathematical theories relevant for these activities, and by informal discussions.

7.3.1 Courses for Graduate Students

The following courses were given in the academic years 1984/85 and 1985/86 by the members of the group or by visiting lecturers invited by the group:

Formal Language Theory (J. Maluszynski)

Introduction to Logic (J. Maluszynski)

Introduction to Logic Programming (J. Maluszynski, W. Drabent)

Formal Methods in the Design and Verification of Microprogrammed Hardware (P. Dembinski)

Attribute Grammars and Attribute Evaluation (P. Deransart)

Edinburgh LCF (J. Leszczylowski)

Algebraic Specifications (seminar J.Maluszynski)

Logic Programming (research seminar)

7.3.2 Direct Contacts

The major area of interaction have been with CADLAB (esp. regarding parallel architectures and specification techniques) and PELAB (esp. regarding theory of programming languages). There are also some plans for possible cooperation with AILAB concerning logic programming.

7.4 External Contacts

7.4.1 External Cooperation

A great deal of the results has been obtained in cooperation with researchers abroad (see p.7.2.1.). We hope to continue such a cooperation also in the future. We consider also a possibility of starting new joint projects.

7.4.2 Conferences and Seminars

During 1985 the work of our group was presented at the following conferences:

IEEE Symposium on Logic Programming, Boston, July 1985 (2 papers)

The 3d Japanese-Swedish Workshop on Fifth Generation , Tokyo, Nov.1985

The Workshop on Programs as Objects of Computation, University of Copenhagen, Oct. 1985

We have regular contacts with the logic programming group at the Department of Computer Systems of the Royal Institute of Technology in Stockholm. These contacts are supported by mutual presentations of the current research at the research seminars of both groups.

8.

The Group for Complexity of Algorithms

Andrzej Lingas

8.1 Introduction.

The group for Complexity of Algorithms is concerned with the design and analysis of efficient algorithms and data structures for combinatorial and geometric problems arising in computer science, and the study of the inherent complexity of these problems in simple models of computation. Members of the group believe that work on algorithm and data structures efficiency can often give greater gains than the development of new, faster computers.

The group originated from a part of the former Group for Theoretical Computer Science in the spring of 1985, when dr. Rolf Karlsson extended its research scope to include data structures.

The first year of the group has been mainly spent on a project called Efficient Algorithms and Data Structures for Geometric and Graph Problems funded by STU. The objectives of the project fall into three mutually interrelated categories of data structures, computational geometry and graph algorithms. The idea of the project has been to concentrate on the problems in which the group members have already gained international recognition like data structures on bounded domains, geometric decomposition problems and subgraph isomorphism. The considered problems have applications among others in VLSI chip design and fabrication, graphics, robotics, numerical analysis, chemistry and optimization.

In addition to the research, the group undertakes important consulting and educational tasks in the aspects of algorithm analysis and complexity theory within the department, and it is open to cooperate with other groups. Interactions with other groups are a source of new research problems for the

---

The work in the Group for Complexity of Algorithms is mainly supported by STU, The Swedish Board for Technical Development.

group members. For instance, the problem of minimum number rectangular covering (see 8.3.2) has been posed by CADLAB, and the group has recently become interested in a parallel processor allocation problem and a trajectory problem posed by AILAB, and graph-theoretic concepts in network semantics considered by RKLLAB.

8.2 Group Members

	% 84/85	% 85/86
Group leadership : Andrzej Lingas, Ph.D.	100	80
Bodil Mattsson-Kihlstrom, secr.		7.5
Supervisors: Rolf Karlsson, Ph.D		70
Graduate Students: Christos Levcopoulos	85	85

8.3 Current Research

8.3.1 Efficient Data Structures on Bounded Domains (Rolf Karlsson)

Computational Geometry on a Grid

Computational geometry studies the computational complexity of finite geometric problems. This research focuses on problems where geometric objects are defined by edges between points taken from multi-dimensional grids. Typical problems we consider are: finding closest points, determining point containment, and computing all line segment intersections. The efficient methods we present should be useful within computer graphics and VLSI. For instance, when implementing geometry routines in computer graphics the domain is a moderate sized raster. Our attention is concentrated on orthogonal objects (the edges are parallel to one of the coordinate axes). VLSI technology, for example, often uses only a fixed number of orientations for the object boundaries and wires. Some of the results have been published or submitted for publication in the proceedings of international conferences.

Inherent Problem Costs on Bounded Domains

The research focuses on designing a realistic lower bound model suitable for problems that use a bounded domain. We have developed a *segment graph model*, where a single-source directed graph represents an algorithm solving the problem under consideration. Using versions of this model, we proved lower bounds for the *dictionary* (support insert, delete and search) and *nearest*

neighbor (support insert, delete and find closest) problems. These results have been published in the proceedings of international conferences. In future research, it would be worthwhile to further unify these problem-oriented techniques, and to make the lower bound model we have introduced more general.

8.3.2 Geometric Decomposition Problems (Levcopoulos, Lingas)

Fast Heuristics for Decomposing Polygons into Rectangles

(Christos Levcopoulos)

We have considered heuristics for two problems concerning rectangular decompositions of polygons: (1) Covering general polygons with minimum number of rectangles, and (2) Partitioning isothetic polygons into rectangles by drawing edges of minimum total length. Both problems have applications in VLSI, the first in fabrication and the second in the design of VLSI chips. The only result concerning the computational complexity of the first problem is that its decision version is decidable, thus only suggesting a double-exponential algorithm for computing the optimal solution. The second problem is *NP*-hard if the polygons have holes.

For the first problem, we proposed and analyzed a heuristic which produces results within a logarithmic factor of the optimum in time $O((n + theta(P)) log\ theta(P))$, where $theta(P)$ is the minimum number of rectangles required to cover the input polygon, and n is the number of vertices. A part of our results has been published the proceedings of an international conference.

For the second problem, we proposed an square-time heuristic. The solutions it produces are in worst case within the constant factor 5 of the optimum, but they are much better in practice. Also, we presented and analyzed an $O(nlogn)$-time heuristic which produces solutions within a constant factor of the optimum. Finally, some heuristics have been presented for certain non-trivial classes of polygons. A part of these results has been published in the proceedings of an international conference, and another part has been submitted for publication.

Geometric Partition Problems without Steiner Points

(Andrzej Lingas)

Partitioning planar figures by drawing straight-line segments without introducing new vertices (called Steiner points) has been considered. In particular, a simple proof for the known theorem on polygon cutting often used in divide-and-conquer geometric algorithms, and generalizations of the theorem to include polygons with polygonal holes have been derived. On the other hand, the research from previous years on the minimum weight triangulation problem has been continued. Among others, a linear-time heuristic for

minimum weight triangulation of convex polygons, and novel, more efficient implementations of the greedy triangulation method have been obtained. Also, several combinatorial properties of greedy triangulations of convex polygons have been derived which have made it possible to perform a non-trivial analysis of the approximation behavior of the greedy method for convex polygons. Two of the mentioned results have already appeared in proceedings of international conferences.

8.3.3 Graph Algorithms (Andrzej Lingas)

Efficient algorithms for computationally feasible instances of so called subgraph isomorphism problem have been developed. The problem consists in determining whether a graph is isomorphic to a subgraph of another graph, it is NP-complete and has a wide spectrum of applications in computer science. A general algorithm for the problem has been designed. When the input graphs are connected, have a good separator (e.g. planar graphs) and relatively small valence, the algorithm runs in subexponential or even pseudopolynomial time. Also, a separate algorithm for subgraph isomorphism constrained to biconnected outerplanar graphs running in cubic time has been designed, in contrast to a published, false proof of NP-completeness of this particular problem. Both algorithms have appeared in proceedings of international conferences.

8.3.4 External contacts

Rolf Karlsson: Written a paper together with Mark Overmars, Rijksuniversiteit Utrecht, The Netherlands. The joint research took shape during a visit he made in Utrecht in November. Joint research to extend a previous conference paper is under way with Ian Munro, University of Waterloo, Canada, and Ed Robertson, Indiana University, USA. Presented papers at the *2nd Symposium on Theoretical Aspects of Computer Science* in Saarbrucken, West Germany (January), and at the *12th International Colloquium on Automata, Languages and Programming* in Nafplion, Greece (July).

Christos Levcopoulos: Presented papers at the *5th International Conference on Foundations of Computation Theory*, Cottbus, East Germany (september), and at the *23rd Allerton Conference on Communication, Control and Computing*, Urbana, Illinois, USA (October).

Andrzej Lingas: Presented papers at the *1st Symposium on Computational Geometry* in Baltimore, USA (June), at the Dept. of Comput. Sci. of Penn State University, USA (June), at the *11-th Workshop on Graph-theoretic Concepts in Computer Science* , West Germany (June), and at the Dept. of Compt. Sci. of Dortmund University , West Germany (June). A cooperation with Dr. J. Sack from Carleton University, Ottawa, Canada, has been established in the area of geometric partition problems. A longer visit to Carleton University and a revisit of Dr. Sack are planned.

On a domestic level, the group has established contacts through mutual visits with an active research group (Svante Carlsson, Arne Andersson) at the Computer Science department, Lund University.

Courses for Graduate Students

An important task of the group is to spread the knowledge of algorithm analysis and complexity theory among graduate students within the department. The following graduate courses are offered for the academic year 85/86:

Search Structures

Analysis and Complexity of Parallel Algorithms

Previously, the following courses were given by the group members:

Algorithm Analysis and Complexity Theory (83,84/85)

Mathematical Aspects of VLSI (84)

9.

LIBLAB

The Library and Information Science Research Laboratory

Roland Hjerppe

9.1 Introduction.

LIBLAB, which is a joint project of the Department of Computer and Information Science and the University Library, is funded by the Delegation for Scientific and Technical Information.

Two major themes Document description and representation, and Users and library (systems), and one minor, Networking, especially questions of central vs. local handling are specified in the research program for LIBLAB.

During the first three years, 1983-1985, all of the themes and their subthemes have been looked into but the focus has mainly been on the first theme, Document description and representation.

The first year was spent on surveys, learning available systems and resources, finding personnel, and planning. In the second year document description and representations in the form of rules for cataloging and catalog records were studied by building expert systems.

In 1985 the activities have been concentrated on finishing the work on expert systems for cataloging and starting the long-range HYPERCATalog project, in which a gradual convergence towards one specific area, extended and enhanced catalogs on the basis of hypertext ideas, has taken place, and in which all of

---

The work in LIBLAB is mainly supported by DFI, The Swedish Delegation for Scientific and Technical Information.

the themes have important roles.

9.2 Cataloging and document description.

The interest in document description and representation arises from the observation that in searching for something in a collection of representations, examining these instead of the items, it is impossible to search for that which is not explicitly described and represented. Our descriptions predetermine what we can find, and how we can find it. Representations as carriers of descriptions have each their own structural limitations, hence modulating the descriptions.

9.2.1 Project ESSCAPE

In project ESSCAPE (Expert Systems for Simple Choice of Access Points for Entries) a number of versions of small expert systems have been built using EMYCIN and Expert-Trees (a version of Expert-Ease). The results and experiences from these activities are discussed in the reports listed at the end of this chapter but some general observations can be reiterated.

Documents, which exist in a large number of forms and types, can, for each of these, be described in various ways and for various purposes. In LIBLAB we have for the moment concentrated on published documents since these are the major concern of libraries. We have furthermore delimited our studies to the problems encountered in cataloging, in which the objective is to enable access to items in a collection, as distinct from bibliography in which the the goal is the comprehensive enumeration and description of documents having something in common, and where one of the needs is to show similarities as well as facilitate distinction.

There are a number of descriptive elements that are universally used for identifying a published document. Among these are the author, the title, the date and place of publication, and the publisher. Even within these few dimensions a bewildering variety can be found, creating problems in the description and representation of documents. The problems arise because ideally one would want to have a single unique description for each item, and one which will be the same irrespective of by whom it is done, so that we can be assured that we are referring to the same item. (One way of accomplishing that is through enumeration, as in the use of the International Standard Book Number, ISBN. The problems with ISBNs are that they, naturally, do not describe the item in a way that is useful to most people, they just represent it.) Hence the need for rules for description and for cataloging.

Cataloging is traditionally described as consisting of three tasks: description, choice of headings (access points which determine under which heading the item will be put, and hence can be found, in the linear file that a catalog usually is), and choice of the form of names. Subject description in the form of classification or subject heading is usually performed separately though part of

cataloging.

In building the ESSCAPE-systems we chose to implement parts of the Anglo-American Rules for Cataloging, Second edition, (AACR2), as the knowledge base. In AACR2 are codified procedures for description of items in a collection according to a view in which authorship, or, generalizing, responsibility for the creation of the item, is the preferred first identifier, the main heading.

Some of our findings are:

The structure of AACR2, although well conceived, raises more questions than it solves, viz. accepting the model with a general frame for decription, as in chapter 1, which is then applied on different types of materials, one soon finds occasion to ask 1) how should such a frame be used?, (mainly referring to it as is now the case?), 2) what should be done in the frame and what in the application? (e.g. should the frame have provisions which are used in only two of a dozen applications?), and 3) why immediately and consistently break the pattern of the model in the applications?

Three different structures have been found for the rules, a "logical" and hierarchical, which is the one used in presenting the rules in AACR2, a "pruning", decision flow based which is used in learning the rules, and a "normal case and exceptions" which is the one used in daily practice.

The really difficult tasks, e.g. recognizing and interpreting a "title page", are not attended at all but rather taken as granted.

In description decisions are made that later influence the choices for access points.

9.2.2 Other projects

Names, of people, places, and organizations, are encountered in the description of every document. Names can be more or less complete or correct, they change, there are differing naming customs, not to mention languages and scripts. Control of names, in the form of selecting one name, and one form of it as the authoritative, preferred one, is an onerous burden that libraries have taken upon themselves in order to be able to provide access to e.g. everything published by a specific person or institution. Authority control is the designation of this activity which LIBLAB has become interested in as a general problem. A system called LINS (LIblab Name handling System) has been built by three students. In LINS those of the rules in AACR2 concerning choice of form of names that are operationalizable are used as a knowledge base that is consulted in the updating of a database of names. Rules for transcription from Russian to several languages are used to provide access to different transcription when only one has been available.

The problems encountered in authority control and in building LINS are instances of a more general problem. Thesuri and classification schedules can be regarded as two other different solutions to the same problem. In project HYPERCATalog this general class of problems will be attacked as a sub-area.

9.3 Project HYPERCATalog

Project HYPERCATalog, which is outlined below), will be the main focus of the activities at LIBLAB during the next four years, with all the staff participating. HYPERCATalog is also a joint project with the Scool of Library and Information Science at Tampere University in Finland, and Informatics Management and Engineering Ltd. in London as participants at present. Additional participants are expected from Norway and Denmark during 1986.

The objective in project HYPERCATALOG is to design and build a system for (library) catalogs that integrates the following principles:

- The catalog as a knowledge organization tool, and a private information resource handler.
- The catalog as a hypertext structure, implying navigation and browsing as the primary modes of use.
- The basic building blocks are data elements, links and a collection concept. Each data element has links, each link has data, collections have subcollections.
- Maps and graphic illustrations of structures as tools for visualization of database structure, which mirrors conceptual structures.
- Integration of text and structure editor with other functions.
- The database grows with use, enabling capitalization of the use made of it.
- Multiple views of the database and its structure.
- Private, modifiable versions of the database and the collective views.
- Different interaction modes, user models and customization needed to accomodate a wide range of users.

Some of the activities foreseen are:

- Studying mappings of database structures and conceptual structures.
- Studying and experimenting with semi-automatic methods for structuring of the database, e.g. clustering, citation networks.
- Studying problems of orientation in conceptual spaces, information and database architecture.

- Studying relations between collections of hypertexts/-media and hypercatalogs.
- Studying maintenance, updating and re-structuring of the database, and relations to private views.
- Studying problems of application to large databases, i.e. collections of millions of documents, which implies gigabyte databases.

9.4 Other activities.

Whereas the HYPERCATalog-project is a rather long-term program, and with a basic research slant, cooperation has been initiated with the Department of Medical Informatics in a joint project on paradigm integration in systems for decision support (DSSs) to General Practitioners (GPs) in primary care, also briefly described in a separate appended paper. This latter project is in part from LIBLAB's point of view regarded as a test bench and application area, but also as means for providing important feedback. It is from the point of view of the Dept. of Medical Informatics concerned with basic problems of decision making in primary care, and with tools to support the problem solving of GPs. Most of the staff in this project are hence from the Dept. of Medical Informatics, and participating GPs.

Additionally, both LIBLAB and Dept. of Medical Informatics are part of a cooporation between the University and County council in a project developing an information network for primary health care as a part of establishing a medical education program in Linköping that is called the Health University to indicate its commitment to health rather than disease, and that integrates education for people at various levels in medical and health care.

The other areas that will be investigated at LIBLAB are formalization of bibliographic description and catalogs, using ideas and formalism from SGML, Standard Generalized Markup Languages and abstract editors, and bibliometrics and citation analysis.

9.5 Personnel

LIBLAB is now fully staffed and an interesting mixture has been achieved. The personnel of LIBLAB is briefly presented below, most of them participate in all projects but with different emphases.

Roland Hjerppe, MSE, Laboratory leader, spends, apart from planning, coordination and administration etc., most of the time on the HYPERCATalog project and on the building of a model for bibliographic representation.

Lisbeth Björklund, B.Sc., library assistant in the interlending department of Linköping University Library, began her doctorate studies at LIBLAB the fall of 1985.

Bodil Gustafsson, BA, librarian, head of the cataloging department of Linköping University Library, has worked 50% of her time in LIBLAB, mostly on the project surveying the forms of representations, and on the HYPERCATalog project.

Hans Holmgren, MSE, divides his time equally between LIBLAB and teeaching at Administrative Data Processing, concentrating on bibliograhic description, and on the HYPERCATalog project.

Birgitta Olander, BA, librarian, former head of the acquisitions department, worked 50% of her time in LIBLAB on the ESSCAPE-project. BO is also pursuing doctorate studies at University of Toronto, Faculty of Library and Information Science and spent the later half of 1985 with LIBLAB in the HYPERCATalog project, before going back to finish her studies in Toronto during 1986.

Arja Vainio-Larsson, MA, former lecturer in psychology, began her doctoral studies at LIBLAB the fall of 1984 and has mostly been taking courses but participated also mainly in user modelling and in the HYPERCATalog project.

Anne-Marie Jacobson, is the part time secretary for LIBLAB.

Associated people:

The following have various associations to LIBLAB:

Kristian Wallin, student, responsible for local systems at the university library and for the NYTTFO-project in Linköping, will join LIBLAB as a doctoral student on a half-time basis after finalizing his BA-paper, and devote most of his time to the HYPERCATalog project.

Manny Jägerfeld, BA, who has a research scholarship from DFI for studying computerization in libraries, will also concentrate on HYPERCATalog and has started his doctorate studies at LIBLAB the fall of 1985.

Hans-Ove Frid, BA, who also has a research scholarship from DFI, but for studying catalog use, will be involved in the survey of representations and in the HYPERCATalog project.

9.6 List of publications

Reports:

(i.e. more extensive writings, reprints, etc.)

LiU-LIBLAB-R-1985:1
Hjerppe,R. and Olander,B: Artificial Intelligence and Cataloging: Building Expert Systems for Simple Choice of Access Points For Entries; Results and Revelations Juni 1985, 29+61p.

LiU-LIBLAB-R-1985:2
Hjerppe, R.; Olander, B. and Marklund, K.: Project ESSCAPE - Expert Systems for Simple Choice of Access Points for Entries: Applications of Artificial Intelligence in Cataloging Juni 1985,16+37p. (Presented at IFLA 51st Conference in Chicago, 18-24 Aug. 1985)

LiU-LIBLAB-R-1985:3
Hjerppe, R.: Project HYPERCATalog: Visions and preliminary conceptions of an extended and enhanced catalog Sept. 1985, 20+7p (Presented at IRFIS 6 (International research Forum in Information Science 6) Sept. 16-18, 1985, Frascati, Italy.

Working papers:

(i.e. usually preliminary, smaller papers, distributed as requested and from separate mailing list)

1. *LiU-LIBLAB-WP:27* Gustafsson, B.: En studie av bibliografiska representationer i LiUbs förvärvsregister Febr. 1985, 11p
2. *LiU-LIBLAB-WP:28* Gustafsson, B.: En fördjupad studie av bibliografiska representationer i LiUbs förvärvsregister Juni 1985, 7p.
3. *LiU-LIBLAB-WP:29* Hjerppe, R.: Vad vet datorn? Kunskapsöverföring via dator Juni 1985, 9p. (Presentation vid Tekniska Träffen i Linköping 6-7 juni 1985)
4. *LiU-LIBLAB-WP:30* Hjerppe,R.: LIBLAB. Planer för verksamheten 1986 - 1988 Oktober 1985, 8p.
5. *LiU-LIBLAB-WP:31* Hjerppe, R.: En modell för informationsresurshantering (IRM) med tillämpning på Hälsouniversitetets bibliotek Oktober 1985, 9p. (Underlag för ett inlägg vid ett av Hälsouniversitetet anordnat ideseminarium de 2 oktober 1985.)
6. *LiU-LIBLAB-WP:32* Timpka, T.; Strömberg, D.; Möller, I.; Gill, H.; Bjurulf,P.; Hjerppe,R.; Mattsson, p.; Wigertz, O.: Three Approaches to Decision Support for General Practitioners: Hypertext, Knowledge Base and Electronic Library November 1985, 15p.

10.

ADP Administrative Data Processing

Göran Goldkuhl

10.1 Administrative data processing

Including management information systems analysis and information systems analysis and design.

The subject area covered by this group deals mainly with social aspects of design and use of software for administrative applications in private companies and public services. Essential problems are the transition from natural to formal languages and vice versa together with prerequisites for, constraints on, and effects of computerized support for activities where teamwork, personal judgement and experience traditionally have been, and are expected to be, of great importance. This topic comprises systems development and tools for analysis of information requirements and tools for prototyping, the drawing up of technical requirements specifications and other kinds of user-oriented documentation and evaluation of effects caused by the use of computerized systems. It does also contain - from a general point of view - social methodology for describing administrative professional activities, for implementation, maintenance and evaluation of user-oriented computerized support.

The undergraduate study programme for Systems Analysis takes the main part of the group's teaching efforts. Beyond that we give separate single-subject courses to the level of postgraduate studies as well as courses in other study programmes.

10.2 Research activities.

Post-graduate and research activities related to the ADP undergraduate programs have previously mainly covered problems of formalization at the interface between formal logic and social science, including cognitive psychology. Through the recent arrival of Göran Goldkuhl and Annie Röstlinger from Gothenburg, we foresee a strenghtening and an expansion of research within the ADP group.

The following areas of research will be covered within the ADP group:

- Change analysis, i.e. the decision concerning computerization and/or other change actions in organizations.
- Information requirements analysis and the development of professional languages of different user groups.
- Knowledge development during information systems development with a special emphasis on critical analysis, creativity and authentic communication.
- Utilization of information systems and end users' language use and knowledge formation.
- Information systems and quality of working life.
- Qualitative research methods and humanistic foundations for information systems science.

There is currently no formal subject-oriented research organization within the humanities and social sciences faculty (research is organized into interdisciplinary "themes"). This explains the present comparatively small size of research activities within the ADP group. However a graduate study programme in administrative data processing is currently being proposed.

10.3 Personnel:

Göran Goldkuhl, PhD, senior lecturer
Eva-Chris Svensson, MSc, director of undergraduate studies
Carina Björkman, secretary
Anne-Marie Jacobson, secretary

Dahlgren, Birgitta, assistant
Hans Holmgren, MScE
Rolf Nilsson, BSc, lecturer
Tommy Ohlsson, assistant
Lise-Lotte Raunio, lecturer
Annie Röstlinger, BSc , lecturer
Dan Strömberg, MScE, lecturer (now at Foa)
Roger Zollner, BSc, assistant
Per Övernäs, BSc, lecturer

Appendix A

The Knowledge Transfer Program.

During the last years we have experienced a growing concern in industry about the rapid development in the information technology area and also a considerable increase in the interest for what is going on at the university. For instance, the following observations are made:

* Software competence is becoming an increasingly critical resource
* Software costs pose serious problems
* Computerized systems are difficult to change and maintain
* There is a fast international development with joint programs for R&D

This development has resulted in a demand for a rapid expansion of educational programs, a pressure on university staff from the labor market, requests for direct assistance in industry projects and in general in an increased volume of contacts between industry and the university.

Thus we have made conscious efforts to improve our contacts with industry, especially to make knowledge transfer more effective. The outflow of personnel from our research program to industry positions has been maintained at a reasonable level. We regret that some who leave, do it without finishing their degrees, but although the value of a licentiate or PhD is rapidly improving in industry, we still have to accept that the salary structure does not give the desired incentive to complete a degree in all cases.

One way to achieve knowledge transfer is to accept commissions from industry in research-related projects. It is our opinion that we should be restrictive about undertaking such commissions in order to maintain the fundamental goals of a research department. Again the differences in salaries are such that we can not expect our personnel to stay at the university doing similar work as they might do in industry. Such commissions are then preferably forwarded to the consulting firms, which are rapidly establishing themselves in the university environment.

For our industry knowledge transfer program (KTP) we have chosen a third alternative. We have inaugurated a joint program where a small number of large industries, which are heavily dependent on proficiency in information technology, are invited to participate in knowledge transfer activities at the department.

The goal for this program is to:

* Promote an effective use of the results from the STU program for knowledge development.

* Provide a knowledge base for industry.
* Secure the availability of qualified competence within novel information technology areas of high importance.
* Contribute to an awareness about industry needs within the university.

The fundamental assumption is that the university guarantee that research projects of a high international standard is carried out within areas of common interest. In connection with this research the university undertake to *organize projects for medium term visitors from industry with an emphasis on learning, technology evaluation and other forms of knowledge transfer.* The obligation for each participating company is to:

- assign one person full time or two persons half time working on the joint projects at the university, with the primary objective to learn and evaluate novel technologies and methods.
- Contribute 600 000 SEK a year to the KTP budget administered by the university.

The joint activities are organized in close contact with the research projects in the laboratories. Presently we have established two areas for the program:

- o **AI and expert systems,**
  including, methodology for knowledge acquisition and expert systems development, evaluation of tools, theoretical foundations and basic techniques, applications e.g. in robotics, manufacturing, technical maintenance, office systems, etc. (ASLAB, AILAB, RKLLAB)
- o **Production technology for software,**
  with an emphasis on programming environments, especially incremental tools for languages in the Algol/Pascal/Ada family. (PELAB)

We have been contaced by companies both with an interest primarily in commercial applications and companies concerned with technical systems. The program started in 1984 and at present S-E-Banken Ericsson Information Systems, ASEA and Alfa-Laval are participating. The model preferred by the companies has been to have two persons working halftime at the university. We expect to be able to accomodate a few more participants in the program within the near future and negotiations are presently carried out with interested companies.

The key ideas of the KTP effort are:

1. The university carries out research projects relevant for industry in aeras which are expected to have high future potential.
2. The program engages companies highly dependent upon advanced information processing.
3. The emphasis is on next-generation software technology.
4. Novel and advanced equipment and software tools are used in experimental settings.

5. The research content of the program should be of high international quality.
6. The ulitimate goal of joint activities is to supply participating companies with a qualified background for strategic decision making, internal use, and internal training within the information technology area.

We feel that the following are the main benefits for the participating companies:

- The immediate availability of powerful enviroments for experimentation with new software technologies.
- Support for evaluation of new trends, methodologies and products.
- Sharing of resources, especially critical-size research teams in areas where competent personnel is a scarce commodity.
- Participation in pilot projects near the edge of the research front line.
- Education of own personnel.
- Basis for recruiting students after undergraduate education.

The program presents a highly efficient way of communicating results to industry and to provide immediate access to the international research community. We have also experienced that the demonstrated industry relevance of our research program improves the possibilities to recruit the best students for graudate education.

Appendix B

Graduate Study Program.

Figure B.1 below indicates the levels of degrees in the Institutes of Technology (i.e. schools of engineering) in the Swedish university system. The figures indicate the nominal numbers of years for the studies in each step.

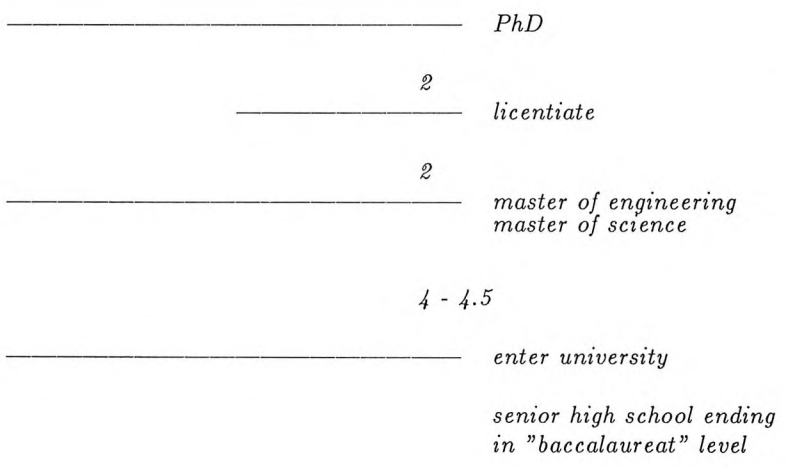


Fig B.1. *Levels of degrees*

The graduate study program provides the studies *from* the level of master of engineering, *to* the licentiate and/or PhD degrees. The courses given by our department for the undergraduate education, up to the master's degree level, are described in appendix 3.

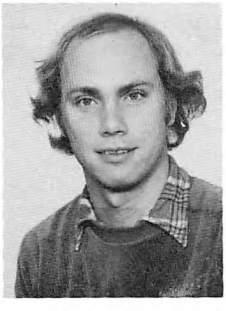
Graduate studies in the department of Computer and Information Science are organized as a program consisting of courses and project participation. The course program is planned at the department level and consists of *basic courses*, each of which is given every third year (if possible), and *occasional courses* which depend on the profile and interests of current faculty and visiting scientists. Project work is always done within one of laboratories or research groups.

Faculty engaged in graduate study program.

Douglas Busch, PhD (Rockefeller 1973, associate professor in logic and theoretical computer science. Previous affiliation Mcquarie University, Sydney, Australia). Application of theories from formal logic to problems in theoretical computer sience and artificial intelligence; algebraic specification theory, intuitionistic type theory non-monotonic logic; philosophical questions in artificial intelligence.

Wlodzimierz Drabent, PhD (Warszawa 1985, on leave from Institute of Computer Science, Polish Academy of Sciences). Logic programming, programming language semantics.

Pär Emanuelson, PhD (Linköping 1980, previous affiliation Uppsala), Senior lecturer in computer science. Now at Epitec AB. Part time thesis supervision in PELAB during 1985. Functional languages, program verification, program analysis and program manipulation, programming environments, software engineering.



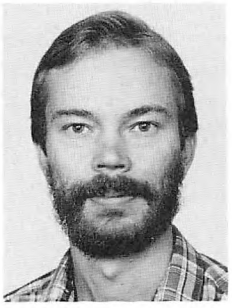
Peter Fritzson, PhD (Linköping 1984), researcher. (On leave for Sun Micro Systems 1985/86.) Thesis supervision in PELAB. Tool generation, incremental tools, programming environments.

Göran Goldkuhl, PhD (Stockholm 1980, previous affiliation Göteborg), senior lecturer. Group leader in ADP research. Information requirement analysis, behavioral aspects of information systems, research methodologies, information systems and quality of working life.

Anders Haraldsson, PhD (Linköping 1977, previous affiliation Uppsala), senoir lecturer and director of undergrade studies in computer science. Thesis supervision in PELAB. Programming languages and systems, programming methodology, program manipulation.

Roland Hjerppe, (previous affiliation KTH, DFI and expert mission Tanzania,) researcher. Group leader, LIBLAB. Library science and systems, citation analysis and bibliometrics, fact representation and information retrieval, hypertext, human-computer interaction and personal computing.

Sture Hägglund, PhD (Linköping 1980, previous affiliation Uppsala), researcher. Group leader, ASLAB. Expert systems and artificial intelligence applications, database technology, human-computer interaction.



Rolf Karlsson, PhD (Waterloo 1984, previous affiliation Lund), researcher. Data structures, algorithm analysis, computational complexity, computational geometry.

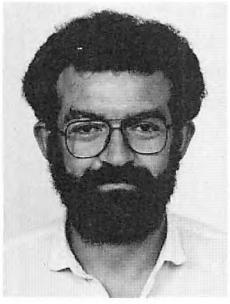
Harold W. Lawson Jr., PhD (Stockholm, several previous affiliations, also in industry), professor of telecommunication and computer systems. Computer architecture, VLSI, Computer-aided design, methodology of computer-related education and training.

Bengt Lennartsson, PhD (Göteborg 1974, previous affiliation Luleå), researcher. Group leader, PELAB. Programming environments, real-time applications, distributed systems.

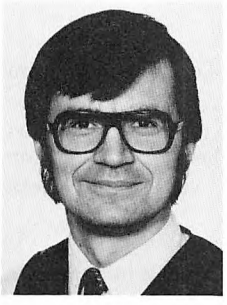
Andrzej Lingas, PhD (Linköping 1983, previous affiliation Warszawa and MIT), researcher. Group leader in geometric complexity. Complexity theory, analysis of algorithms, geometric complexity, graph algorithms, logic programming, VLSI theory.

Bryan Lyles, PhD (Rochester 1982), acting professor of telecommunications and computer systems and group leader, CADLAB, 1984-85. Now at Rochester. Computer architecture, VLSI, user interfaces, distributed systems.

Jan Maluszynski, PhD (Warszawa 1973, several previous affiliations), docent and group leader in theoretical computer science. Logic programming, software specification methods.



Kevin Ryan, PhD (Trinity College, Dublin), guest researcher in ASLAB 1985-86. Software engineering methods and environments. Educational and social issues.



Erik Sandewall, PhD (Uppsala 1969), professor of computer science. Group leader in RKLLAB. Representation of knowledge with logic, theory of information management systems, office information systems, autonomous expert systems.

Linda C. Smith, PhD (University of Illinois at Urbana-Champaign), guest researcher in LIBLAB, spring 1985. Information retrieval and artificial intelligence.

Bo Sundgren, PhD (Stockholm 1973, previous affiliation Uppsala, also at Statistics, Stockholm), adj. professor. Group leader in statistical information systems. Database design and database-oriented systemeering, conceptual modelling, statistical information systems.

Erik Tengvald, PhD (Linköping 1984), researcher. Group leader, AILAB. Artificial intelligence, knowledge representation, planning and problem solving, expert systems.

Associated faculty in other departments:

Jan-Olof Brüer, PhD, researcher in information theory. Office information systems, especially security issues.

Ingemar Ingemarsson, PhD, professor of information theory. Information theory, security and data encryption, error correction codes and data compression.

Ove Wigertz, PhD, professor of medical informatics. Medical information systems, expert systems.

Graduate Study Course Program 1984-85

Basic and Occasional Graduate Courses:

AI and Expert Systems (Erik Sandewall)

Formal Methods in the Design and Verification of Microprogrammed Hardware. (Piotr Dembinski)

Distributed Computing. (Bryan Lyles)

Parallel Machines and Message-Based Architectures (Bryan Lyles)

Introduction to Logic Programming (Jan Maluszynski)

Sequential Algorithm Analysis and Computational Complexity (Andrzej Lingas)

Attribute Grammars and Attribute Evaluators (Pierre Derensart)

Information Retrieval and Artificial Intelligence (Linda C. Smith)

Dependency-Directed Reasoning (Jim Goodwin)

Computer Related Education and Training: Content and Methodology (Harold W. Lawson)

Edinburgh LCF (Jacek Leszczylowski)

Incremental Programming Systems (Bengt Lennartsson)

Research-Related Courses and Seminars:

Program databases (Bengt Lennartsson)

AI Programming. (Erik Tengvald)

Information Retrieval and Library Systems. (Roland Hjerppe)

Expert System Tools - comparative analysis and evaluation (Sture Hägglund)

Robotics of today and the future (Peter S. Nilsson)

Special Courses for the Knowledge Transfer Program:

The Structure and Interpretation of Computer Programs (Roland Rehmnert) (Fall and spring)

Graduate Study Course Program 1985-86

(*A number of courses, previously offered as graduate courses, are now part of the undergraduate computer science curriculum and given for the first time this academic year. For that reason, the offering of specific graduate courses is somewhat more restricted this year, than what could else be expected.*)

Basic and Occasional Graduate Courses:

Theory of systems development (Göran Goldkuhl)

Non-Standard Logics for Artificial Intelligence (Erik Tengvald)

Software Engineering (Benny Odenteg, Kevin Ryan)

Search structures (Rolf Karlsson)

Computer Architecture / VLSI (Harold W Lawson)

Change analysis (Göran Goldkuhl)

Knowledge organization (Roland Hjerppe)

Analysis and Complexity of Parallel Algorithms (Andrzej Lingas)

Research-Related Courses and Seminars:

Debugging of programs with parallel processes (Bengt Lennartsson)

The AIM project (Erik Tengvald)

The HYPERCATalog project (Roland Hjerppe)

Authority control (Roland Hjerppe)

Expert System Tools - comparative analysis and evaluation (Fall.) (Sture Hägglund)

AI and software engineering (Spring) (Sture Hägglund)

Statistical Information Systems (Bo Sundgren)

RKLLAB seminars on non-standard logics, office systems and representation of knowledge about machinery. (Erik Sandewall)

Future CAD Systems (Harold W Lawson)

Logic Programming (Jan Maluszynski)

Complexity of algorithms (Andrzej Lingas)

Robotics of today and the future (Peter S. Nilsson)

Special Courses for the Knowledge Transfer Program:

Knowledge Engineering with EMYCIN. (Kristian Sandahl)

Introduction to KEE. (Roland Rehmnert)

A Selection of Seminars 1985

General seminars spring 1985

18/1 Ralph Rönnquist, IDA, RKLLAB, Relational algebra in I.M.S. Theory

29/1 John Walters, SRI International, Menlo Park: Current trends in AI and expert systems.

12/2 Örjan Ekeberg, NADA, KTH. Experiences of Smalltalk.

25/2 Rolf Karlsson, IDA. Proximity on a grid.

26/2 Harold Lawson, IDA, "Sabbatical Report"

5/3 Gunnar Carlstedt, Göteborg, "High Level CAD Tools"

11/3 Nicholas J. Belkin, The City University, London,"Using problem structures for driving human-computer dialogues"

11/3 Piotr Dembinski. Logic Programming Seminar - Intelligent Backtracking.

12/3 Sven Mattisson, Lund, CONCISE: A Concurrent Circuit Simulator

18/3 Wlodek Drabent, IDA. Logic Programming Seminar - Memory management of Prolog implementations.

19/3 Linda C. Smith, Liblab. "From Memex to Procognitive Systems to Expert Systems: Information Retrieval as an Application Domain for Artificial Intelligence".

25/3 Andrzej Ciepielewski, KTH. Or-parallel execution of logic programs

29/3 Peter Buneman, Univ of Pennsylvania. Inheritance, Data Models and Data Types

17/4 Hans Block. SCB. Redskap för expertsystem - Presentation och demonstration av SAGE

19/4 P.P. Bonissone, A.L. Brown, General Electric, Schenectady. Expanding the Horizon of Expert Systems. GE's Approach to AI.

26/4 Andri Ariste, Inst of Cybernetics, Tallinn. Activities in the Computer Field in Estonia

29/4 Andrzej Lingas, IDA. Algoritmgruppen - The Complexity of the Subgraph Isomorphism Problem

21/5 Per Gunningberg. Presentation of current work.

General seminars fall 1985

3/9 Kevin Ryan: ToolUse and other ESPRIT projects

Graduate Study Program.

17/9 Gregor Snelting, Darmstadt. The PSG - Programming System Generator

20/9 Erik Tengvald, IDA. AILAB:s AIM-projekt

23/9 Jan Maluszynski, IDA. Logic Programming Seminar. Research in Efficiency of Logic Programs

30/9 Håkan Jacobsson, LiTH. Logic Programming Seminar - AND Parallelism and Nondeterminism in Logic Programs

1/10 A. Ström, Context Vision: The programming environment for GOP-300

7/10 Wlodek Drabent, Simin Nadjm-Tehrani, IDA. Logic Programming Seminar - Classes of Logic Programs: An analyzer of logic programs ...

11/10 Johan Fagerström, IDA. OCCAM, a Concurrent Language

14/10 Yngve Bohlin. Logic Programming Seminar - Implementing prolog - compiling predicate logic programs

15/10 Svante Carlsson, Lund. Heaps of Heaps

17/10 Lars Kahn, SU-TVT Infologics AB. Teknowledge's S1 och M1

22/10 Tom Reps, University of Wisconsin, The Synthesizer Generator

24/10 Kenneth Zadeck, IBM, Yorktown Heights, Attribute Propagation by Message Passing

25/10 F. Kenneth Zadeck, IBM, Yorktown Heights, Compiler Optimization

28/10 Ivan Rankin, IDA. Logic Programming Seminar. Introduction to the YACC compiler

28/10 Carl-Wilhelm Welin, Ericsson. Erfarenheter av OPS 4/5 och derivat

29/10 Bertil Svensson, Högskolan i Halmstad. Associativa, parallella datorer

4/11 Andrzej Ciepielewski, KTH. KTH approach to OR Parallelism

25/11 Ulf Nilsson, LiTH. Logic Programming Seminar - An alternative implementation of Definite Clause Grammars

26/11 Thomas Nilsson, SOFTLAB AB: DAISY- a distributed debugger for Ericsson

27/11 ZeBo Peng, IDA. Licentiatseminarium - Steps Towards the Formalization of Designing VLSI Systems

29/11 Jan Maluszynski, IDA. Logic Programming Seminar - Report from the Swedish-Japanese Workshop

29/11 Charles Meadow, Toronto: Intelligent online assistance in information retrieval.

3/12 Kenth Ericson, SOFTLAB AB: PWS - The Parser Writing System

9/12 Toomas Timpka, IMT. Logic Programming Seminar - A demonstration of the POPLOG system

10/12 Pär Emanuelson, EPITEC AB: EPITOOL - an environment for building knowledge systems

16/12 Wlodek Grudzinski, KTH. Logic Programming Seminar - Databases and Logic Programming

17/12 Henrik Nordin, CMU: "Prodigy, a learning apprentice".

Appendix C

Undergraduate Education.

1. Undergraduate teaching in the school of engineering

The group for undergraduate teaching (the UDD-group) is responsible for courses in the two subjects *Computer Science* and *Telecommunication and Computer Systems* given in the undergraduate study programs at the Institute of Technology at Linköping. These curriculums are (figures give number of students accepted annually):

Computer Science (C) for 30 students
Computer Science and Technology (D) for 120 students
Industrial and Management Engineering (I) for 180 students
Mechanical Engineering (M) for 120 students
Applied Physics and Electrical Engineering (Y) for 180 students

These curriculums are four-year programmes and lead to a Master of Engineering or (for the C-programme) a Master of Science degree.

There are also single-subject courses from these programmes given as part-time and evening courses, and external courses given directly to companies and organizations. There has started a programme for "continuing education" of engineers in computer science. The programme has been developed by the department of computer science in cooperation with a coalition of Swedish engineering industry (Oktogonen). A first 2 year course has started during 1985 for 20 students at Ericsson, Stockholm.

Courses. During 85/86 we will give a total of approx 70 different courses. In these curriculums we give 50 courses with a total of 3500 participants, 10 single-subject courses and about 10 external courses for industry with about 400 participants.

These curriculums have at least one introductory course in computer science and programming.

In the C- and D-programmes and in the variants towards computer science in the M- and Y-programmes (which students can choose after the second year) there are courses in

- **programming methodology**
- **assembly programming**
- **data structures**

- data bases
- compiling techniques
- principles of programming languages
- concurrent programming
- operating systems
- artificial intelligence
- computer networks
- computer architecture
- computer aided design of electronics
- discrete simulation

The C-and D-programmes include two software projects. One individually the first year and one in a group during the third year. The projects require of both oral presentations and written reports.

In the C-programme we give a number of human-oriented courses:

- linguistics I and II
- psychology, introductory course
- psychology of communication
- interactive systems

We give also courses in theoretical computer science;

- logic, introductory course
- formal languages and automata theory
- programming theory
- logic programming

and courses in artificial intelligence:

- introduction to AI
- AI programming
- Knowledge representation
- Natural language processing

Computer facilities. A variety of computer systems are available to our students. Most courses use a DEC-20 computer running the TOPS-20 operating system at the Computer Centre at Linköpings University. It supports ca 60 terminals running concurrently.

The department has a PDP11/70 with UNIX for teaching purposes with 15 terminals. There are two PC laboratories with MacIntoshes and Ericsson PC's. In project courses and courses in artificial inteligence they use Xerox LISP machines.

There are 11 terminal rooms (8-9 terminals per room) and a network for connecting terminals to the various computer systems available for educational purposes.

Staff. The teaching is made by full or half time employed lecturers, by other persons with research appointment, by graduate students with teaching assistantships, and by the students themselves as part-time course assistents.

During 85/86 the staff consists of

6 full time and 2 half time senior lecturers (associate professors)
5 full time and 2 half time lecturers (assistant professors)
14 other persons, professors and research assistents
16 postgraduate students with 25% - 50% teaching assistantships
ca 4 teachers from other subjects and from industry
ca 40-50 part-time course assistents

Personnel.

Anders Haraldsson, PhD, associate professor in computer science, director of undergraduate studies
Barbara Ekman, secretary

The following persons are teaching one or more courses:

Computer Science:
Lars Ahrenberg, BSc
Rober Bilos, MSc
Nils Dahlbäck, BSc
Wlodek Drabent, PhD
Pär Emanuelsson, PhD
Christian Gustafsson,BSc
Anders Haraldsson, PhD
Sture Hägglund, PhD
Arne Jönsson, MSc
Rolf Karlsson, PhD
Andrei Lingas, PhD
Bengt Lennartsson, PhD
Jalal Maleki, MSc
Jan Maluszynski, PhD
Magnus Merkel, BSc
Benny Odenteg, BSc
Kerstin Olsson, MSc
Tommy Olsson, MSc
Mikael Patel, MSc
Roland Rehmnert, MSc
Kevin Ryan, PhD
Kristian Sandahl, MSc

Erik Sandewall, PhD
Nahid Shahmehri, MSc
Ola Strömfors, MSc
Katarina Sunnerud, MSc
Eva-Chris Svensson, BSc
Lars Wikstrand, BSc
Olle Willen, MSc
Mats Wiren, MSc
Per Övernäs, BSc

Telecommunication and computer systems:
Johan Fagerström, MSc
Björn Fjellborg, MSc
Harold W Lawson, PhD
Mikael Patel, MSc

Listing of Undergraduate Course Program 1985-86

Datateknisk översiktskurs (C1, D1)
Databaser (D3, I4)
Databaser (C3, Y3, Y4, Md4)
Programmeringsspråk (C4, D4)
Programmering i Ada(C4, D4)
Programmeringsmiljöer (C4, D4)
Systemutveckling, teroi och tillämpning (C4, D4)
AI-programmering (C4)
Logik, grundkurs (C1, D4)
Psykologi grundkurs (C2)
Naturligt språkbehandling (C4)
Operativsystem (D3tk, D4, Y4, I4)
Programmering Y, grundkurs (Y1)
Algoritm och komplexitetsteori (C4)
Programmerung Y, fortsättningskurs (Y3, Y4)
Kompilatorer och interpretatorer (Y4, I4)
Programmeringsteori II (C4)
Administrativ databehandling (Y4, I4)
Cobol (Y4, I4)
Logikprogrammering (C3, D4)
Programmeringsteori (C3)
Processprogrammering (D3pv, D4, Md4, Y4, I4)
Operativsystemteori (C3, D3pv, M4d)
Programmering och projektarbete i Pascal (C1, D1)
Lagringssstrukturer (C2, D2, Md3)
Assemblyprogrammering (C2, D2, Md3, I4)
Programutvecklingsmetodik (D2, Md3)
Programutvecklingsmetodik och programmeringsprojekt II (D3)
Programutvecklingsmetodik och programmeringsprojekt C (C3)
Data och programstrukturer D (D3)

Data och programstrukturer C (C2)
Datorspråk (C3, D3, I4)
Artificiell intelligens C (C3)
Artificiell intelligens D (D4)
Programmering Y (Y2)
Programutveckling (I1)
Datastrukturer och programutvecklingsmetodik (I2)
Datorsystem och programmering (M1)
Programmering i inkrementellt systen (C1)
Interaktiva system (C1, D3pv, D3tk)
Lingvistik I (C1)
Formella språk och automatateori (C2)
Lingvistik II (C2, C3)
Kommunikationspsykologi (C3)

Diskret simuleringsteknik (D3, Y3)
Datornät (D4, Y4, I4, Md4)
Datorarkitektur (D4, Y4)
Datorstödd elektronikkonstruktion (D4, Y4, I4)

Datalogi - baskurs (Fristående enstaka kurs Norrköping)
Programmeringsprinciper (Fristående enstaka kurs Norrköping)
Programmering i Ada (Fristående enstaka kurs Linköping)
Datakunskap (utbildningsradion)
Datalära

2. Undergraduate course program in systems analysis, in the school of humanities and sciences

The educational programme for systems analysis ranges over three years of fulltime studies. This aims at professional activities of design, evaluation and implementation of computer-based information systems. Because of that, ADP-systems analysis dominates the programme. Nevertheless great importance has been attached to other subjects in order to give the programme the necessary breadth and also to ensure that the students will become aware of the complexity of the community where computers can be used.

The first two years of the programme constitute a common core of basic studies for all students. Within the subject of ADP-systems analysis there are courses in systems development and systems theory as well as courses in programming and computer science. The courses about systems development and systems theory deal with formal methods and prototyping. For the programming courses Pascal has been chosen as the main language but, of course, other languages are taught as well. Within the field of computer science the students take courses in database design, development of interactive

systems, methodology for program development, communication, evaluation of computer systems, programming methodology, etc. Other subjects that are given within the common core of basic studies are:

- business economics and management, to get basic knowledge about the organization of corporations and public services and their "commonday" routines.
- human factors, industrial and social psychology, including ergonomics, work environment, co-determination and participative management, group dynamics etc.

There are also courses in practical swedish language for professional use, social science, matematics and statistics. The second year ends with about five months of on the job training.

During the last year the students can choose between one of the following three specializations:

- Methods for data analysis (data analysis), aimed at statistical methodology and statistical analysis methods. This specialization includes documentation and presentation of projects where storage and retrieval of data are crucial.
- Development of computer programs and program systems (program development) aimed at program development, methodology and technology. This specialization contains courses about operating systems, compilers, interpreters etc.
- Development of information systems (systemeering), aimed at methodology for design and evaluation of information systems. The program includes in-depth studies of budgeting and accounting and their relation to project management and systems budgeting.

All three specialisations end with a term-paper reporting the development and implementation of an individual project.

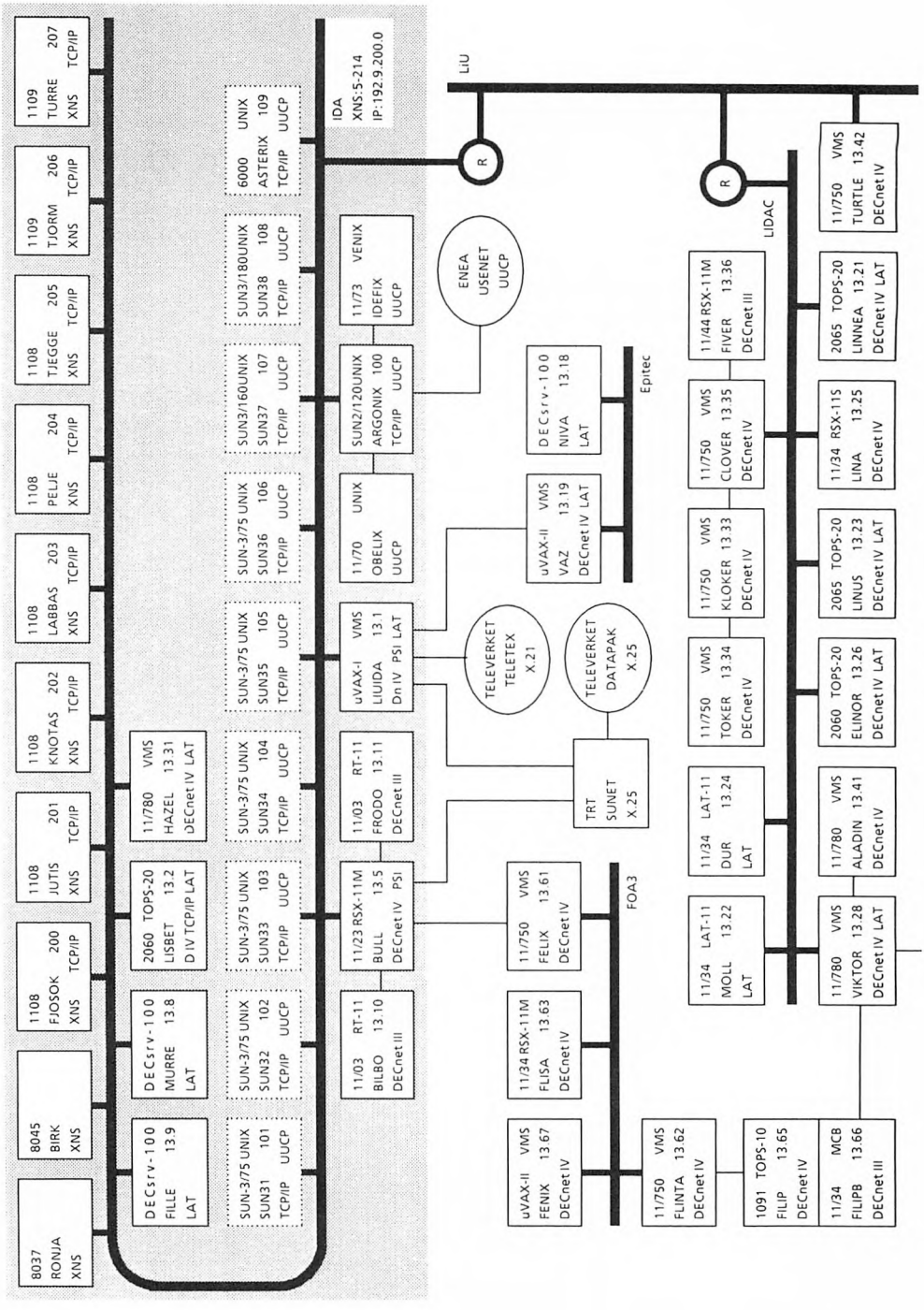
Appendix D

Computer Facilities.

The department has a policy of giving high priority to the supply appropriate computing resources for research and education. We have also during the years been able to modernize and keep in pace with the rapid development in the area, e.g. regarding the emergence of powerful workstations with high-resolution graphics and high-performance CPU. Our orientation towards experimental computer science makes such a policy especially important and we believe that adequate computer equipment is essential for the quality of research and education.

Our main computer resources for research are a DECsystem-2060 (there are additional systems for undergraduate education), a VAX 780 (which is shared with the Physics department) and a Xerox Ethernet with eight 1108/1109 Lisp Machines, file server and laser printer. (We expect to add an additional number of Xerox 1186 workstations in the near future.) We have also recently acquired a few SUN workstations. In addition there are lots of smaller computers (MicroVax, PDP-11:s, MacIntosh and other PC:s of various kinds.) There is also special purpose equipment, especially for text processing.

The schematic picture on the next page shows the local network and the accessible computer systems.


Network visible from IDA LiU 860110
8037 RONJA XNS
8045 BIRK XNS
1108 FJOSOK 200 XNS TCP/IP
1108 JUTIS 201 XNS TCP/IP
1108 KNOTAS 202 XNS TCP/IP
1108 LABBAS 203 XNS TCP/IP
1108 PELJE 204 XNS TCP/IP
1108 TJEGGE 205 XNS TCP/IP
1109 TJORM 206 XNS TCP/IP
1109 TURRE 207 XNS TCP/IP
DECsrv-100 FILLE 13.9 LAT
DECsrv-100 MURRE 13.8 LAT
2060 TOPS-20 LISBET 13.2 DIV TCP/IP LAT
11/780 VMS HAZEL 13.31 DECnet IV LAT
SUN-3/75 UNIX SUN31 101 TCP/IP UUCP
SUN-3/75 UNIX SUN32 102 TCP/IP UUCP
SUN-3/75 UNIX SUN33 103 TCP/IP UUCP
SUN-3/75 UNIX SUN34 104 TCP/IP UUCP
SUN-3/75 UNIX SUN35 105 TCP/IP UUCP
SUN-3/75 UNIX SUN36 106 TCP/IP UUCP
SUN-3/75 UNIX SUN37 107 TCP/IP UUCP
SUN3/160UNIX SUN38 108 TCP/IP UUCP
SUN3/180UNIX SUN39 108 TCP/IP UUCP
6000 UNIX ASTERIX 109 TCP/IP UUCP
IDA XNS:5-214 IP:192.9.200.0
11/03 RT-11 BILBO 13.10 DECnet III
11/23 RSX-11M BULL 13.5 DECnet IV PSI
11/03 RT-11 FRODO 13.11 DECnet III
uVAX-I VMS LIUIDA 13.1 DnIV PSI LAT
11/70 UNIX OBELIX UUCP
SUN2/120UNIX ARGONIX 100 TCP/IP UUCP
11/73 VENIX IDEFIX UUCP
LiU
R
ENEA USENET UUCP
TELEVERKET TELETEX X.21
TELEVERKET DATAPAK X.25
TRT SUNET X.25
FOA3
Epitec
uVAX-II VMS VAZ 13.19 DECnet IV LAT
DECsrv-100 NIVA 13.18 LAT
11/34 RSX-11M FLISA 13.63 DECnet IV
11/750 VMS FELIX 13.61 DECnet IV
uVAX-II VMS FENIX 13.67 DECnet IV
11/750 VMS FLINTA 13.62 DECnet IV
11/34 LAT-11 DUR 13.24 LAT
11/750 VMS TOKER 13.34 DECnet IV
11/750 VMS KLOKER 13.33 DECnet IV
11/750 VMS CLOVER 13.35 DECnet IV
11/44 RSX-11M FIVER 13.36 DECnet III
11/34 LAT-11 MOLL 13.22 LAT
1091 TOPS-10 FILIP 13.65 DECnet IV
11/780 VMS VIKTOR 13.28 DECnet IV LAT
11/780 VMS ALADIN 13.41 DECnet IV
2060 TOPS-20 ELINOR 13.26 DECnet IV LAT
2065 TOPS-20 LINUS 13.23 DECnet IV LAT
11/34 RSX-11S LINA 13.25 DECnet IV
2065 TOPS-20 LINNEA 13.21 DECnet IV LAT
LIDAC
11/34 MCB FILIPB 13.66 DECnet III
11/750 VMS TURTLE 13.42 DECnet IV

Appendix E

Publications since 1980.

DISSERTATIONS:

(*Linköping Studies in Science and Technology. Dissertations.*)

No 51 **Erland Jungert:** Synthesizing Database Structures from a User Oriented Data Model, 1980.

No 54 **Sture Hägglund:** Contributions to the Development of Methods and Tools for Interactive Design of Applications Software, 1980.

No 55 **Pär Emanuelson:** Performance Enhancement in a Well-Structured Pattern Matcher through Partial Evaluation, 1980.

No 58 **Bengt Johnsson, Bertil Andersson:** The Human-Computer Interface in Commercial Systems, 1981.

No 69 **H. Jan Komorowski:** A Specification of an Abstract Prolog Machine and its Application to Partial Evaluation, 1981.

No 71 **René Reboh:** Knowledge Engineering Techniques and Tools for Expert Systems, 1981.

No 77 **Östen Oskarsson:** Mechanisms of Modifiability in Large Software Systems, 1982.

No 94 **Hans Lunell:** Code Generator Writing Systems, 1983.

No 97 **Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983.

No 109 **Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation, 1984.

No 111 **Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984.

LICENTIATE THESES:

(*Linköping Studies in Science and Technology. Theses.*)

No 17 **Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. 1983.

No 28 **Arne Jönsson, Mickael Patel:** An Interactive Technique for Communicating and Realizing Algorithms. 1984.

No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator. 1984.

No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching. 1985.

No 52 **Zebo Peng:** Steps towards the Formalization of VLSI Design Systems, 1985.

No 60 **Johan Fagerström:** Simulation and Evaluation of an Architecture based on Asynchronous Processes. 1986.

EXTERNAL PUBLICATIONS.

(*Papers published in books, journals or international conference proceedings.*)

1. **Piotr Dembinski, Jan Maluszynski:** And-Parallelism with Intelligent Backtracking for Annotated Logic Programs, in *Proc of the IEEE Symposium on Logic Programming*, pp 29-38, Boston 1985.

2. **Pierre Deransart, Jan Maluszynski:** Relating Logic Programs and Attribute Grammars. *Journal of Logic Programming, vol 3*, No. 2, pp 119-158, 1985.

3. **Dimiter Driankov**, An outline of a fuzzy sets approach to decision-making with interdependent goals, *Proc. of the First IFSA Congress,* Palma de Mallorca July, 1985. To appear in *Fuzzy Sets and Systems, An Int. Journal.*

4. **Dimiter Driankov**, Inference with single fuzzy conditional proposition, to appear in *Fuzzy Sets and Systems*, (1985).

5. **Dimiter Driankov**, A calculus for belief-intervals- representation of uncertainty, to appear in *Proc of the Int. Conf. on Information Processing and Management of Uncertainty*, Paris, June 1986.

6. **Johan Elfström, Jan Gillqvist, Hans Holmgren, Sture Hägglund, Olle Rosin, Ove Wigertz:** A Customized Programming Environment for Patient Management Simulations. *Proc. of the 3rd World Conf. on Medical Informatics*, Tokyo, 1980.

7. **Pär Emanuelson, Anders Haraldsson:** On Compiling Embedded Languages in Lisp. *Proc. of the 1980 LISP Conf.*, Stanford, Calif, 1980.

8. **Pär Emanuelson:** From Abstract Model to Efficient Compilation of Patterns. *Proc. of the 5th Int. Conf. on Programming*, Turin, 1982. Revised version to appear in *Science of Computer Programming.*

9. **Johan Fagerström:** Experiences with Occam: A Simulator for Asynchronous Processes. In *Proc. of the Hawaii Int. Conf. on System Sciences, HICSS-19*, 1986.

10. **Peter Fritzson:** A Systematic Approach to Advanced Debugging through Incremental Compilation. *Proc of the ACM SIGSOFT/SIGPLAN Symposium on High-Level Debugging,* Pacific Grove, CA., March 1983.

11. **Peter Fritzson:** Symbolic Debugging through Incremental Compilation in an Integrated Environment. *The Journal of Systems and Software* 3, 285-294, (1983).

12. **Peter Fritzson:** Preliminary Experience from the DICE System - A Distributed Incremental Compiling Environment. *Proc. of the ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments,* Pittsburgh, PA. April 1984.

13. **Peter Fritzson:** The Architecture of an Incremental Programming Environment and some Notions of Consistency. In *Proc. of the GTE Workshop on Software Engineering Environments for Programming-in-the-large*, Harwichport, MA. June 10-12, 1985.

14. **James W. Goodwin:** Why Programming Environments Need Dynamic Data Types. *IEEE Trans. Software Eng., vol SE-7,* no 5, 1981. Also in Barstow et al. (eds.) *Interactive Programming Environments*, McGraw-Hill, 1984.

15. **James W. Goodwin and Uwe Hein:** Artificial Intelligence and the Study of Language. *Journal of Pragmatics, 6*, pp 241-280, North-Holland, 1982.

16. **James W. Goodwin:** WATSON - A Dependency Directed Inference System, In *Proc. of the AAAI Workshop on Non-Monotonic Reasoning*, New Palz, NY, 1984.

17. **James W. Goodwin:** A Process Theory of Non-Monotonic Inference. in *Proc. of*

the Int. Joint Conf. on Artificial Intelligence, IJCAI, 1985.

18. **Sture Hägglund:** Dialogue Models for Human-Computer Communication. A Practitioner's View. in *Proc. of the Workshop on Models of Dialogue: Theory and Application.* Linköping 1981.

19. **Sture Hägglund, Johan Elfström, Hans Holmgren, Olle Rosin, Ove Wigertz:** Specifying Control and Data in the Design of Educational Software. *Computers & Education, vol 6,* no 1, 1982.

20. **Sture Hägglund, and Roland Tibell:** Multi-Style Dialogues and Control Independence in Interactive Software. In Green et al. (eds.) *The Psychology of Computer Use*, Academic Press, 1983. Previous version in *Proc. of the 1st European Conf. on Cognitive Engineering*, Amsterdam, 1982.

21. **Sture Hägglund:** On the Design of a Query Environment for Office Use. *Proc. of the 2nd Scandinavian Seminar on Information Modelling and Database Management*, Tampere, 1983.

22. **Uwe Hein:** Interruptions in Dialogue. Also in D. Metzing (ed), *Dialogmuster und Dialogprozesse.* Hamburg, Buske, 1981.

23. **Uwe Hein:** Natural and Artificial Communications. - Some Reflections -. in *Proc. of the Workshop on Models of Dialogue: Theory and Application.* Linköping 1981.

24. **Uwe Hein:** Constraints and Event Sequences. *Proc of the NATO symp. on Artificial Intelligence*, Lawrence Erlbaum, 1982.

25. **Uwe Hein:** PAUL - A Programming Language for Knowledge Engineering Applications. *Proc. of the International Conference on Artificial Intelligence*, Leningrad, October 1983.

26. **Roland Hjerppe:** What artificial intelligence can, could, and can't, do for libraries and information services. *Proc. 7th IOLIM,* Learned Information Ltd. London. December 1983.

27. **Hans Karlsson, Roland Lindvall, Olle Rosin, Erik Sandewall, Henrik Sörensen and Ove Wigertz:** Experience from Computer Supported Prototyping for Information Flow in Hospitals. *Proc. of the ACM SIGSOFT Second Software Engineering Symposium: Workshop on Rapid Prototyping,* Columbia, Maryland, April 19-21, 1982.

28. **Hans (Karlsson) Gill, Bertil Kågedahl, Erik Sandewall, Henrik Sörensen, Lennart Tegler, and Ove Wigertz:** A Notation for Information Flow Models Supporting Interactive System Development. *Proc of the 6th Annual Symposium on Computer Applications in Medical Care*, Washington DC, nov 1982.

29. **Rolf Karlsson, Ian Munro**, Proximity on a Grid, in the *Proc. of 2nd Symposium on Theoretical Aspects of Computer Science* (1985), *Springer-Verlag Lecture Notes on Computer Science 182,* 187-196

30. **Rolf Karlsson, Ian Munro, Ed Robertson,** The Nearest Neighbor Problem on Bounded Domains, in the *Proc. of 12th Int. Colloquium on Automata, Languages and Programming* (1985), *Springer-Verlag Lecture Notes on Computer Science 194,* pp 318-327.

31. **H. Jan Komorowski:** QLOG - The Software for Prolog and the Logic Programming. *Proceedings of the Logic Programming Workshop*, Debrecen, Hungary, 1980. Also in Clark, Tärnlund (eds.) *Logic Programming*, Academic Press, 1982.

32. **H. Jan Komorowski:** Partial evaluation as a means for inferencing data structures in an applicative language: a theory and implementation in the case of Prolog. *Proc of the Symp. on Principles of Programming Languages*, Albuquerque, 1982.

33. **H. Jan Komorowski:** An Abstract Prolog Machine. *Proc. of the European Conf. on Integrated Interactive Computing Systems*, Stresa, 1982.

34. **H. Jan Komorowski:** A Prototype Compiler for Prolog. Poster version presented at *the 6th Int. Conf. on Software Engineering*, Tokyo, 1982.

35. **Harold W. Lawson Jr.**: New Directions in Micro- and System Architecture in the 1980's, in *Proc. of the National Computer Conference, NCC-81,* Chicago, Ill., 1981.

36. **Harold W. Lawson Jr.**: An Approach to Improving Computer Literacy, in *Teaching Informatics Courses: Guidelines for Trainers and Educationalists*, (ed. by A.L.W. Jackson), North-Holland, 1982.

37. **Harold W. Lawson Jr.**: The Holistic Approach in Introducing Computer Systems, in *The Computing Teacher, vol 10*, no 7, October 1982. Also in Japanese translation in Nikkei-Computer, Niekkei-McGraw-Hill, Tokyo, 1982.

38. **Harold W. Lawson Jr.**: An Architecture-Based Strategy for Improving Computer Education, in *Proc. of the Euromicro 82 Symposium*, Brussels, September 1982.

39. **Harold W. Lawson Jr.**: Some Consequences of Tomorrows Electronics CAD Systems, in *Proc. of Mantech 83, Discoveries Int. Symp.*, London, 1983.

40. **Harold W. Lawson Jr.**: Computer architecture education, a chapter in Tiberghien (Ed.): *New Computer Architectures*, pp 224-285, Academic Press, 1984.

41. **Harold W. Lawson Jr.**: Impact of CAD and Integrated Circuit Developments on Telecommunication. *Proc. of the EUTECO Conference,* Oct 1983, Varese, Italy.

42. **Harold W. Lawson Jr.**: Ingrediants and Implications of Tomorrows CAD Systems. *Integrated Circuit Seminar,* July 18-22 1983, Singapore.

43. **Harold W. Lawson Jr.**: Architecting VLSI Systems. *Integrated Circuit Seminar,* July 18-22 1983, Singapore.

44. **Harold W. Lawson, Jr.:** Addressing Fundamental Problems in Computer Related Education and Training. In *Proc. of the 4th World Conf. on Computers in Education*, Norfolk, 1985.

45. **Harold W. Lawson Jr., Bryan Lyles:** An Architecturial Strategy for Asynchronous Processing. in *Concurrent Languages in Distributed Systems: Hardware-Supported Implementation*, (ed. by Reijnsand, Dagless), North-Holland, 1985.

46. **Christos Levcopoulos, Andrzej Lingas:** Covering Polygons with Minimum Number of Rectangles, *Proceedings of the STACS Symposium, Paris (1984), Lectures Notes in Computer Science, vol 166*, Springer Verlag.

47. **Christos Levcopoulos:** On Covering Regions with Minimum Number of Rectangles, *Proceedings of the Workshop on Parallel Computing and VLSI*, Amalfi, Italy, (1984) North-Holland Publ. Co.

48. **Christos Levcopoulos, Andrzej Lingas:** Bounds on the Length of Convex Partitions of Polygons, in the *Proc. of the 4th FST-TCS Conference*, Bangalore, India, (1984), *Lectures Notes in Computer Science, vol 181*, Springer Verlag.

49. **Christos Levcopoulos,** Minimum Length and "Thickest-First" Rectangular Partitions of Polygons, in the *Proc. of the 23rd Allerton Conf. on Comm., Control and Computing,* Illinois, October 1985.

50. **Christos Levcopoulos,** A Fast Heuristic for Covering Polygons with Rectangles, in the *Proc. of 5th Int. Conf. on Foundations of Computation Theory*, GDR, (1985), *Lectures Notes in Computer Science, vol 199*, Springer Verlag.

51. **Christos Levcopoulos:** Fast Heuristics for Minimum Length Rectangular Partitions of Polygons, to appear in *Proc of the 2nd ACM Symposium in Computational Geometry*, Yorktown Heights, June 1986.

52. **Andrzej Lingas:** Heuristics for Minimum Edge Length Rectangular Partitions of Rectangular Partitions of Rectilinear Figures, *Proceedings of 6th GI Conference on*

Theoretical Computer Science, Dortmund (1983), *Lectures Notes in Computer Science*, Springer Verlag.

53. **Andrzej Lingas:** An Application of Maximum Bipartite C-Matching to Subtree Isomorphism, *Proceedings of the 8th Colloquium on Trees in Algebra and Programming*, L'Aquila (1983). *Lectures Notes in Computer Science, vol 159*, Springer Verlag.

54. **Andrzej Lingas:** A Note on Complexity of Logic Programs, *Proceedings of the Logic Programming Workshop*, Aldeia das Acoteias, Portugal (1983).

55. **Andrzej Lingas:** The Greedy and Delauney Triangulations are not bad in the average case and Minimum Weight Geometric Triangulation of Multi-Connected Polygons is NP-complete, *Proceedings of the International Conference on Foundations of Computation Theory*, Borgholm (1983), *Lecture Notes in Computer Science, vol 158*, Springer Verlag. See also *Information Processing Letters, vol 22*, pp 25-31, (1986).

56. **Andrzej Lingas:** A Linear-Time Heuristic for Minimum Weight Triangulation of Convex Polygons. *Proc. of the Allerton Conference on Communication, Control, and Computing*, Urbana, Illinois 1985.

57. **Andrzej Lingas:** Subgraph Isomorphism for Easily Separable Graphs of Bounded Valence. *Proc. of the 11th Int. Workshop on Graphtheoretic Concepts in Computer Science*, Castle Schwanberg, Wuerzburg, Germany, June, 1985.

58. **Andrzej Lingas:** On Partitioning Polygons. *Proc of the 1st ACM Symposium on Computational Geometry*, Baltimore, Maryland, June 1985.

59. **Andrzej Lingas,** Subgraph Isomorphism for Biconnected Outerplanar Graphs in Cubic Time, in the *Proc. of the 3rd Symposium on Theoretical Aspects of Computer Science,* January, 1986, Orsay, France, *Lecture Notes in Computer Science, vol 210*, Springer Verlag.

60. **Andrzej Lingas:** On Approximation Behavior and Implementation of the Greedy Triangulation for Convex Planar Point Sets, to appear in *Proc of the 2nd ACM Symposium in Computational Geometry*, Yorktown Heights, June 1986.

61. **J. Bryan Lyles:** CAD Approaches for an Asynchronous Architecture. In *Proc. of the Nordic Symposium on VLSI in Computers and Communications*, June 13-15, 1984, Tampere, Finland.

62. **J. Bryan Lyles, Zebo Peng, Johan Fagerström:** Naming Services in a Distributed Computer Architecture. In *Proc. of the Nordic Symposium on VLSI in Computers and Communications,* June 13-15 1984, Tampere, Finland.

63. **Jan Maluszynski, Jorgen Fischer Nilsson:** A Comparison of the Logic Programming Language Prolog with Two-Level Grammars. *Proc. of the 1st Logic Programming Conference*, Marseille-Luminy, 1982.

64. **Jan Maluszynski, Jorgen Fischer Nilsson:** A version of Prolog based on the notion of two-level grammar. *Proc. of the Prolog Programming Environments Workshop*, Linköping, 1982.

65. **Jan Maluszynski, Jorgen Fischer Nilsson:** Grammatical Unification. *Information Processing Letters, vol 15*, pp 150-158, (1982).

66. **Jan Maluszynski:** Towards a Programming Language based on the Notion of Two-Level Grammar. *Theoretical Computer Science, vol 28*, pp 13-43, North-Holland (1984).

67. **Jan Maluszynski, H. Jan Komorowski:** Unification-Free Execution of Logic Programs, in *Proc of the IEEE Symposium on Logic Programming*, Boston 1985.

68. **Henrik Nordin:** Using Typical Cases for Knowledge-Based Consultation and Teaching. To appear in *Proc of the 3rd Annual Conf. on Applications of Expert*

Systems, Orlando, Fla., 1986.

69. **Ludmila Ohlsson:** A Computer Model for Domain Dependent Systems. *Proc of 7th Int. ALLC Symp. on Computers in Literary and Linguistic Research*, Pisa, 1982 (North-Holland).

70. **Mikael Patel, Arne Jönsson:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, in *Proc of the 19th Annual Hawaii Int. Conf. on System Sciences, HICSS-19, 1986.*

71. **Zebo Peng:** A Unified Approach to Design Representation and Synthesis of VLSI Systems. In *Proc. of the 19th Annual Hawaii Int. Conf. on System Science, HICSS-19,* Jan, 1986.

72. **Zebo Peng:** Synthesis of VLSI Systems With The CAMAD Design Aid, to appear in *Proc. of the 23rd ACM/IEEE Design Automation Conference*, Las Vegas, June 1986.

73. **Günter Riedewald, Jan Maluszynski, Piotr Dembinski:** *Formale Beschreibung von Programmiersprachen*, R. Oldenburg Verlag, Munchen, Wien, (1983).

74. **Roland Rehmnert, Kristian Sandahl:** Knowledge Organization in an Expert System for Spot-Welding Robot Configuration. In *Proc. of the 5th Int. Workshop on Expert Systems and Their Applications,* Avignon, 1985.

75. **Olle Rosin, Hans Holmgren, Sture Hägglund,** Implementing Tuning and Feedback Facilities in a System for Patient Management Simulations, *Proc. 3rd Congress on Medical Informatics Europe*, Toulouse, 1981.

76. **Piotr Rudnicki, Wlodzimierz Drabent:** Proving Properties of Pascal Programs in MIZAR 2, *Acta Informatica, vol 22,* pp 311-331, 1985.

77. **Kristian Sandahl, Sture Hägglund, Jan-Olof Hildén, Roland Rehmnert, Lars Reshagen:** The Antibody Analysis Advisor and its Migration into a Production Environment. To appear in *Proc. of the 1st Int. Conf. on Expert Systems*, London 1985.

78. **Erik Sandewall et al:** Provisions for Flexibility in the Linköping Office Information System, *Proc. of the National Comp. Conf.*, Los Angeles, 1980.

79. **Erik Sandewall, Claes Strömberg, Henrik Sörensen:** Software Architecture Based on Communicating Residential Environments. *Proc. of the 5th Int. Conf. on Software Engineering*, San Diego, 1981. Also in Barstow et al. (eds.) *Interactive Programming Environments*, McGraw-Hill, 1984.

80. **Erik Sandewall, Henrik Sörensen, Claes Strömberg:** A System of Communicating Residential Environments. *Proc. of the 1980 LISP Conf.*, Stanford, Calif, 1980

81. **Erik Sandewall:** Unified Dialogue Management in the Carousel System. *Proc. of the ACM Conference on Principles of Programming Languages*, Albuquerque, NM, 1982. Appeared in print in *N. Naffah (ed.) Office Information Systems,* North Holland, 1982.

82. **Erik Sandewall:** An Environment for Development and Use of Executable Application Models. Presented at the *seminar "Software factory experiences",* Capri, May 3-7, 1982.

83. **Erik Sandewall, Sture Hägglund, Christian Gustafsson, Lennart Jonesjö, Ola Strömfors:** Stepwise Structuring - A Style of Life for Flexible Software. *Proc. of the National Computer Conference,* Anaheim, 1983.

84. **Erik Sandewall:** Formal Specification and Implementation of Operations in Information Management Systems. In: Jan Heering and Paul Klint (eds.), *Colloquium Programmeeromgevingen*, MC Syllabus, Mathematisch Centrum, Amsterdam 1983.

85. **Erik Sandewall:** A Functional Approach to Non-Monotonic Logic. in *Proc of the*

Int. Joint Conf. on Artificial Intelligence, IJCAI, 1985 and *Computational Intelligence, vol 1*, no 2, pp 80-87, 1985.

86. **Erik Sandewall:** Non Monotonic Inference Rules for Inheritance with Exception. To appear in *Proceedings of the IEEE, Special Issue on Knowledge Representation.*

87. **Erik Sandewall:** Specification Environments for Information Management Systems. Panel position paper to appear in *Proc. IFIP Congress 1986.*

88. **Piotr Siemienski:** A specialized VLSI CAD DATABASE. *Nordic Symposium on VLSI in Computers and Communications*, June 13-15 1984, Tampere, Finland.

89. **Dan Strömberg, Peter Fritzon:** Transfer of Programs from Development to Runtime Environments. *BIT, vol 20*, no 4, 1980.

90. **Ola Strömfors, Lennart Jonesjö:** The Implementation and Experiences of a Structure-Oriented Text Editor. *Proc of the ACM SIGPLAN/SIGOA Symposium on Text Manipulation*, Portland, Oregon, June 8-10, (SIGPLAN NOTICES, vol 16, no 6) 1981.

91. **Ola Strömfors,** Editing Large Programs Using a Structure-Oriented Text Editor. To appear in *Proc. of the Int. Workshop on Advanced Programming Environments.* Trondheim, Norway. June 1986.

92. **Bo Sundgren:** How to Satisfy a Statistical Agency's Need for General Survey Processing Programs. *Proc. of the 45th Session of the International Statistical Institute*, Amsterdam, Aug 12-22, 1985.

93. **Erik Tengvald**, Reducing Design Complexity, or Why does AI Work, *Proc. of the AIMSA-84 Conf.*, Varna, Bulgaria, (1984).

94. **Ove Wigertz, Johan Elfström, Sture Hägglund and Olle Rosin:** Computer-Assisted Training in Patient Management and Clinical Decision Making. in Pages et al. (eds.) *Meeting the Challenge: Informatics and Medical Education*, North-Holland, 1983.

95. **Jerker Wilander:** An interactive programming system for Pascal. *BIT, vol 20*, 2, 1980. Also in Barstow et al. (eds.) *Interactive Programming Environments*, McGraw-Hill, 1984.

96. **Yoshikazu Yamamoto, Mats Lenngren:** Graphic Model Building System, in *Proc of the 16th Annual Simulation Symposium,* Tamppa, Fla., 1983, and in *Proc of the IMACS Symp. on Simulation in Engineering Sciences,* North-Holland, Nantes, 1983.

OTHER RESEARCH REPORTS:

(*Departmental reports, contributions to nordic conferences, and papers awaiting external publication.*)

97. **Pierre Deransart, Jan Maluszynski:** Modelling Data Dependencies in Logic Programs by Attribute Schemata, INRIA Report RR323, and LiTH-IDA-R-84-08 (1984).

98. **Wlodzimierz Drabent:** An Experiment with Domain Construction for Denotational Semantics. LiTH-IDA-R-85-17

99. **Dimiter Driankov**, Inference with consistent probabilities in Expert Systems, submitted for publication 1985.

100. **Kenth Ericson, Hans Lunell:** Redskap för kompilatorframställning LiTH-MAT-R-80-39

101. **Peter Fritzson:** Distribuerad PATHCAL: Förslag till ett distribuerat interaktivt

programmeringssystem för PASCAL. LiTH-MAT-R-81-05

102. **Peter Fritzson:** Fine-Grained Incremental Compilation for Pascal-Like Languages. LiTH-MAT-R-82-15

103. **Peter Fritzson:** Adaptive Prettyprinting of Abstract Syntax applied to ADA and PASCAL. LiTH-IDA-R-83-08

104. **James W. Goodwin:** An Improved Algorithm for Non-monotonic Dependency Net Update. LiTH-MAT-R-82-23

105. **Hans Grunditz, Uwe Hein, Erik Tengvald:** Artificiell intelligens i framtidens CAD/CAM system. LiTH-MAT-R-82-30

106. **Sture Hägglund:** Towards Control Abstractions for Interactive Software. A Case Study. LiTH-MAT-R-80-37

107. **Sture Hägglund** et al: 80-talets elektroniska kontor: Erfarenheter från LOIS-projektet. LiTH-MAT-R-81-04

108. **Sture Hägglund:** Informationshantering i det elektroniska kontoret. *Proc. of the 5th Nordic conf. on Information and Documentation*, Trondheim, 1982. LiTH-MAT-R-82-27.

109. **Sture Hägglund:** En analys av interaktiva frågesystem för relationsdatabaser. *Proc. NordData 83, Oslo, 1983.*

110. **Sture Hägglund:** Mot femte generationens programvara i samarbete högskola - näringsliv. In *Proc NordDATA 85,* Copenhagen, 1985.

111. **Anders Haraldsson:** Experiences from a Program Manipulation System. LiTH-MAT-R-80-24

112. **Anders Haraldsson:** INTERLISP - en avancerad integrerad programmerings-omgivning för LISP-språket. *Proc. Nord-Data 82*, Göteborg, 1982. LiTH-MAT-R-82-29.

113. **Kristo Ivanov:** From Computers to Information and Systems Science. LiU-MAT-ADB-R-80-3

114. **Kristo Ivanov:** Teologisk logik och systemteori. LiU-MAT-R-81-2

115. **Kristo Ivanov:** Sekundära sannolikheter i beslutsfattande. LiU-MAT-R-81-3

116. **Kristo Ivanov:** An elementary data-structure for data processing systems. LiU-MAT-R-82-2

117. **Kristo Ivanov:** Presuppositions of formal methods for development of computer systems. LiU-IDA-R-83-1

118. **Kristo Ivanov:** Computer applications and organizational disease. LiU-IDA-R-83-2

119. **Kristo Ivanov:** Systemutveckling och ADB-ämnets utveckling. LiU-IDA-R-84-1

120. **Erland Jungert:** Deriving a Database Schema from an Application Model Based on User-defined Forms. LiTH-MAT-R-80-35

121. **H. Jan Komorowski, James W. Goodwin:** Embedding Prolog in Lisp: An Example of a Lisp Craft Technique. LiTH-MAT-R-81-02

122. **Krzysztof Kuchcinski, Zebo Peng:** Microprogramming Implementation of Timed Petri Nets. LiTH-IDA-R-85-19

123. **Harold W. Lawson Jr.**: Introducing Computer Concepts and Terminology, in *Proc. NordDATA 81, Copenhagen, 1981.*

124. **Harold W. Lawson Jr., Arne Jönsson:** Algoritmbeskrivning och Dimensional Flowcharts, in *Proc. NordDATA 82, Göteborg, 1982.*

125. **Harold W. Lawson Jr.**: Tools for Tomorrows Integrated Hardware/Software

Development, in *Proc. NordDATA 83, Oslo, 1983.*

126. **Harold W. Lawson, Jr.:** Sabbatical Report. LiTH-IDA-R-85-05

127. **Harold W. Lawson Jr.:** Swedish Participation in the Malaysian National Microelectronics Programme, LiTH-IDA-R-85-11

128. **Bengt Lennartsson, Ola Strömfors:** DICE - en portabel integrerad programmeringsomgivning. In *Proc NordDATA 85,* Copenhagen, 1985.

129. **Mats Lenngren, Yoshikazu Yamamoto:** GMBS - Graphic Model Building System, in *Proc NordDATA 83, Oslo, 1983.*

130. **Hans Lunell:** Some notes on the terminology for Compiler-Writing Tools LiTH-MAT-R-80-41

131. **Hans Lunell:** En konceptuell maskin för Pascal. (Preliminär version). LiTH-MAT-R-82-09

132. **Henrik Nordin:** Kunskapsbaserade stödsystem i framtidens bankarbete. In *Proc. NordDATA 85,* Copenhagen, 1985.

133. **Alexander Ollongren:** On the Implementation of Parts of Meta-IV in Lisp. LiTH-MAT-R-81-07

134. **Östen Oskarsson:** Construction of Customized Programming Languages. LiTH-MAT-R-81-10

135. **Östen Oskarsson, Henrik Sörensen:** Integrating Documentation and Program Code LiTH-MAT-R-81-01

136. **Michael Pääbo:** CAD-elektronik idag och i framtiden. LiTH-IDA-R-84-03.

137. **Ralph Rönnquist, Erik Sandewall:** The Relationship between Ordered and Unordered Trees in I.M.S. Theory. LiTH-IDA-R-84-04

138. **Ralph Rönnquist:** Relational Algebra in I.M.S. Theory. LiTH-IDA-R-85-06.

139. **Ralph Rönnquist:**The Information Lattice of Networks Used for Knowledge Representation. LiTH-IDA-R-86-02

140. **Kristan Sandahl, Roland Rehmnert:** Expertsystem i verkstadsindustrin. In *Proc NordDATA 85,* Copenhagen, 1985.

141. **Kristian Sandahl:** Creating an Antibody Analysis Advisor as an Exploratory investigation into Expert System Development. LiTH-IDA-R-85-20

142. **Erik Sandewall:** An Approach to Information Management Systems. LiTH-MAT-R-82-19

143. **Erik Sandewall:** Ny teknologi i kontorsdatasystem. *Proc. Nord-Data 82,* Göteborg 1982. LiTH-MAT-R-82-17.

144. **Erik Sandewall:** Partial Models, Attribute Propagation Systems and Non-Monotonic Semantics. LiTH-IDA-R-83-01

145. **Erik Sandewall:** Theory of Information Management Systems. LiTH-IDA-R-83-03

146. **Erik Sandewall:** Fjärde generationens programvaruutbildning. LiTH-IDA-R-84-13.

147. **Piotr Siemienski:** Towards an Integrated VLSI CAD System, the Database and its Implementation. LiTH-IDA-R-85-04.

148. **Anders Ström:** DSS - ett datalagringssystem. *Proc. Nord-Data 82,* Göteborg 1982.

149. **Dan Strömberg:** Text editing and incremental parsing. LiTH-MAT-R-82-34

150. **Erik Tengvald:** En Intuitiv Förklaring till Kildalls Algoritm LiTH-MAT-R-80-27

151. **Erik Tengvald:** A Note Comparing Two Formalizations of Dataflow Algorithms. LiTH-MAT-R-80-28

152. **Erik Tengvald:** AI an Emerging Science. LiTH-IDA-R-84-11

153. **Lars Wikstrand, Sture Hägglund:** A System for Program Analysis and its Application as a Tool for Software Development and Program Transfer. LiTH-MAT-R-80-30

154. **Jerker Wilander:** Felkorrigering i inkrementella programmeringsomgivningar. *Proc. Nord-Data 82*, Göteborg 1982.

GENERAL:

155. **Anders Beckman**: Varför jag inte kan vara datalog: en diskussion av värderingar. LiTH-MAT-R-80-40

156. **J-O Brüer, S. Chowdhury, A. Fäldt, H. Gill and R.Rönnquist:** Office Models. ASLAB Memo 84-01.

157. **Andrzej Blikle:** Notes on the Mathematical Semantics of Programming Languages. (Lecture notes.) LiTH-MAT-R-81-19

158. **Pär Emanuelson:** Programtransformationer. LiTH-IDA-R-83-06

159. **Sture Hägglund, Jon-Erik Nordstrand** (eds.): A Study on Directions for Research and Development of Scientific and Technical Information Systems. (With contributions from **Hein, Hägglund** and **Sandewall.**)

160. **Sture Hägglund:** Datorstödda Informationssystem i ett regionalpolitiskt perspektiv. I Snickars (ed.) *Beslut för regional förnyelse*, Publica 1984.

161. **Sture Hägglund:** Kunskapsbaserade expertsystem. Ny teknik för applikationsutveckling i nästa generations programvarusystem. LiTH-IDA-R-83-07

162. **Uwe Hein**: A Proposal for an Artificial Intelligence Laboratory at SSRC, Linköping, 1980.

163. **Uwe Hein:** Vad är artificiell intelligens? LiTH-MAT-R-81-13 och tidskriften DATA, Köpenhamn, 1982.

164. **Uwe Hein:** Kunskapsteori: Representation, Manipulation och Organisation av kunskap - Del 1 - den teoretiska ramen. (Lecture notes). LiTH-MAT-R-82-04

165. **Uwe Hein:** Kunskapsteori: Representation, Manipulation och Organisation av kunskap - Del 2 - associativa nätverk. (Lecture notes). LiTH-MAT-R-82-07

166. **Uwe Hein:** Kunskapsteori: Representation, Manipulation och Organisation av Kunskap - Del 3 - Lingvistiskt orienterade representationssystem. (Lecture Notes). LiTH-MAT-R-83-08

167. **Uwe Hein, Sture Hägglund** (eds.): Proceedings of the Workshop on Models of Dialogue. Theory and Application, Linköping, 1981. (With contributions from **Hein** and **Hägglund.**)

168. **Kristo Ivanov:** Forskningsanknytning av Universitetets grundutbildning. LiU-MAT-ADB-R-80-1

169. **Kristo Ivanov:** Systemvetenskap och fragmentering av kunskap. LiU-MAT-ADB-R-80-2

170. **Kristo Ivanov:** Mot ett ingenjörsvetenskapligt universitet. LiU-IDA-R-84-2

171. **Kristo Ivanov:** Några policy-riktlinjer för ämnet ADB. LiU-MAT-R-83-3

172. **H. Jan Komorowski** (Ed.): Proceedings of the Symposium on Prolog Programming Environments, Linköping, 1982.

173. **Bengt Lennartsson:** Programvarumiljöer. Produktionsteknik för programvara i

Ada och andra språk. LiTH-IDA-R-84-01

174. **Mats Lenngren, Thomas Pettersson, Michael Pääbo, Ewa Attebo:** Datorgrafikdagar 7-9 juni 1983.

175. **Hans Lunell:** Tre skisser om datalogi som vetenskap LiTH-MAT-R-81-16

176. **Michael Pääbo:** Introduktion till datorgrafik, LiTH-IDA-R-83-02

177. **Erik Sandewall:** Datavetenskaplig utvecklingsmiljö och kunskapsöverförings-program. LiTH-IDA-R-83-10

178. **Dan Strömberg:** Datorn - hjälpreda eller hot i det lilla företaget? (LiTH-MAT-R-81-06)

179. **Dan Strömberg:** Gränserna för artificiell intelligence - en reseskildring (LiTH-MAT-R-82-46)

180. **Dan Strömberg:** Ett kritiskt perspektiv på artificiell intelligens forskning. LiU-MAT-R-82-1

181. **Bo Sundgren:** Conceptual Design of Databases and Information Systems. LiTH-IDA-R-84-09 (Lecture Notes)

SYSTEMS DOCUMENTATIONS:

182. MEDICS - Systemdokumentation och användarhandledning. **Hans Holmgren, Sture Hägglund, Olle Rosin.** (SSRC Systemdok. 17)

183. MEDICS - Författarmanual - Preliminär version. **Hans Holmgren, Sture Hägglund, Olle Rosin.** (SSRC Systemdok. 18)

184. MINISCOPE - Användarhandledning. **Lars Wikstrand.** (SSRC Systemdok. 19)

185. IDECS3 Reference Manual. **Sture Hägglund.** (SSRC Systemdok. 20)

186. ED3 - User's Guide **Ola Strömfors.** (SSRC Systemdok. 21)

187. SCREBAS - Provisional Reference Manual. **Erik Sandewall.** (SSRC Systemdok. 22)

188. Ett gränssnitt mellan LISP 1.6 och MIMER på DEC-10 vid LIDAC. Användarhandledning. **Hans Holmgren.** (SSRC Systemdok. 23)

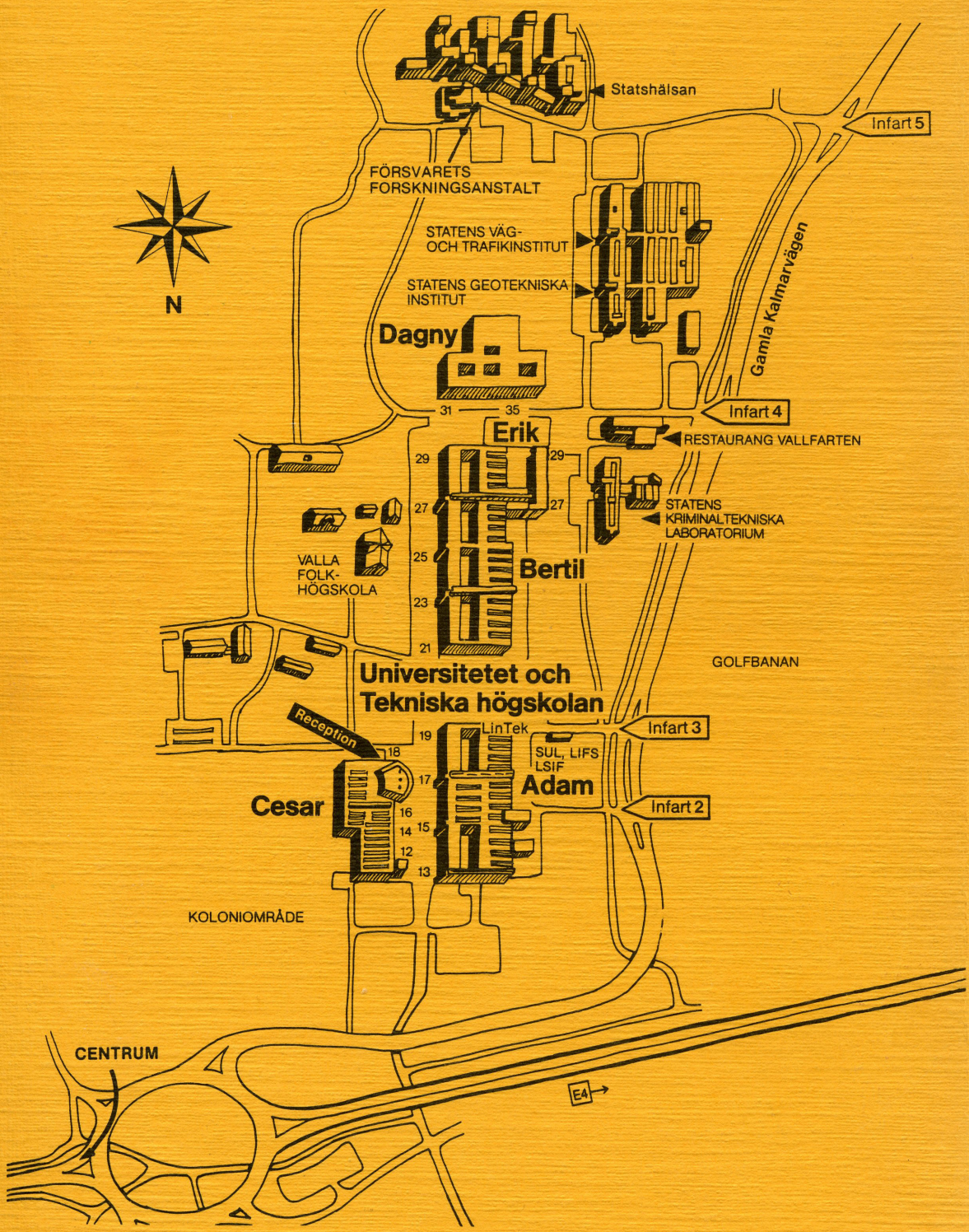
189. ALGOL68C - Release 1.271, Users Guide. **Arne Fäldt.** (SSRC Systemdok. 24)

190. Handledning i användande av PIG. **Olle Willen.** (SSRC Systemdok. 25)

191. The Lois manager. **Erik Sandewall** (SSRC Systemdok. 26)

192. AFORM User's Guide. **Arne Fäldt.** (SSRC Systemdok. 27)

Universitetsområdet, Valla



Klintland Grafiska, Linköping 1986