Dept of COMPUTER AND INFORMATION SCIENCE

Annual research report

1987

LINKÖPING
University and
Institute of technology

LiTH-IDA-R-88-07
ISSN-0281-4250

Spring 1988

The Department of Computer and Information Science

Linköping University

Annual Research Report 1987

This report describes research on software technology and related areas within the Department of Computer and Information Science at Linköping University and Institute of Technology. Main areas of current research are programming environments, artificial intelligence, natural language processing, 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 report also presents other activities in the department, e.g. undergraduate education, the graduate study programme, knowledge transfer to industry, etc.

Mailing address:
Dept. of Computer and Information Science
Linköping University
S-581 83 Linköping
Sweden
Tel: int + 46 13 28 10 00
Telex: 8155076 LIUIDA S
Telefax: int + 46 13 14 22 31

Postadress:
Inst. för datavetenskap
Universitetet och
Tekniska Högskolan i Linköping
581 83 Linköping
Tel: 013 - 28 10 00
Telex: 8155076 LIUIDA S
Telefax: 013 - 14 22 31

CONTENTS

1.

Introduction and Overview

1.1 Research objectives

This is the January, 1988 issue of the yearly overview report of research done at the Department of Computer and Information Science (IDA) of Linköping University. This introductory chapter reviews the scope of our research programme and explains the organizional background for activities in the department.

1.1.1 Current research objectives

The scope and the objectives of our research is influenced by the following external factors:

We are located in the University's School of Engineering (Tekniska högskolan i Linköping). Knowledge areas which are significant for Swedish industry before the end of this century, should have high priority for us.

The scope of IDA's interest is partly defined by the natural borderlines to other departments in the university, notably:

Electrical engineering
Mechanical engineering
Mathematics
Physics
Business administration
Communication studies
Technology and social change

(The last two of those are in the 'Themes' research organization which has an emphasis on social sciences).

Our main source of research grants is the Swedish Board of Technical Development (STU); only about 20% of the research resources are internal university money. STU supports good research, according to the criteria that are commonly accepted in the international research community. They are eager to support the establishment and growth of "centers of excellence" in selected areas. However, it is also important for our sponsor that research results should be transferred to applications in industry, commercial users of computer systems, public administration, or in other areas of research outside computer science.

These goals are sometimes competing, and possibly even contradictory. In IDA we have tried to balance our efforts so that both the basic research goal and the applied goal should be achieved reasonably well. We also recognize the importance of continuous interaction between basic and applied research in our field.

Besides the external factors, the research direction of our department is naturally determined by the roots and the traditions that it has emerged from. Our research profile has evolved from early Swedish efforts in the following areas:

artificial intelligence
programming environments
computer architectures
administrative data processing, data base
and office systems

These areas still represent a large portion of the department's research, but they have been complemented with research also in areas such as

logic programming
complexity theory
library science

We do not wish to be a single-issue department, but at the same time we can not afford to spread out over all possible parts of computer science. The present research profile, as realized by the ten laboratories/groups described in chapters 2-11, attempts to make a reasonable trade-off between concentration and breadth.

1.1.2 Strategic Research Planning.

In early 1986, our University's School of Engineering decided to focus on *Industrial information technology* (IIT) as the primary area for new research efforts. The choice was based on the observation that information technology is both the underlying technology for the information industry (computers, software, telecommunications, electronic components), and also it is one very

important enabling technology for other industries. The term 'industrial information technology' that was adopted by the School of Engineering, refers to that second aspect of the use of information technology.

Government approval of the plan came in February, 1987, with funding being available from July 1988. During the academic year 1987/88 the effort in industrial information technology has been planned by a group with representatives from our department (Erik Sandewall) and the departments of Mechanical Engineering, Electrical Engineering, and Physics. Dr. Tore Gullstrand from Saab-Scania has also participated as a representative for industry.

The actual work in the new *Center for Industrial Information Technology, CENIIT*, will begin in mid-1988. It may eventually involve most of the departments of our school. Activities planned for our part are described in section 1.4 below.

Within our department, we presently find it more important to strengthen the existing laboratories, than to start new ones. The recruiting situation is relatively good, both for faculty and for students, and funding is therefore the primary constraint in most areas of our activities.

IDA's work in the area of administrative data processing has been plagued for many years with a number of problems. The undergraduate study-line in "computer science and business administration" ("systemvetenskapliga linjen") is badly under-funded. On the research side, there are organizational problems along the administrative borderline between 'engineering' and 'social sciences'. IDA brought out these problems for concrete discussion in 1986, and there has been a fairly lengthy debate about how to proceed for the future. Some temporary solutions have been created, but the main problems still remain to be solved.

Our main sponsor, STU, established a "ramprogram" (literally, "frame program") for research in information processing during the period 1980 to 1985. That program was very important for strengthening computer science research in Swedish universities, both because the total amount of funding increased, and because it provided fairly stable funding during a five year long period. A new frame program in our area started January 1988. Our applications for funding from that program came out very favourably, with about one third of the granted money in the first round allocated to IDA.

The STU program described above is part of the national information technology program, which also contains a substantial part aiming at industrial R&D projects. A special effort in that part will be joint activities between our university and the Defense Research Institute, where IDA participate in cooperative research on AI funded with 6 MSEK during 1987/88. Another substantial effort in this area is the PROART project, which is the AI part of the Prometheus Eureka project (future road traffic systems). PROART is coordinated by Erik Sandewall and part of the research is done in our

department.

1.2 Organization.

This annual research report is intended both for our colleagues internationally, and for Swedish readers in industry as well as in the universities. When it comes to the sections about organization and about undergraduate teaching, those two audiences have different frames of reference, and maybe also different interests. If a reader finds some parts of the following text redundant, then maybe that is a result of our attempts to cater to both groups of readers at once.

1.2.1 IDA's current research organization.

The department for computer and information science (*institutionen för datavetenskap*, IDA for short) has presently about 120 employees. This figure includes 17 researchers with a Ph.D. degree. The research in our department is organized as nine (at present) laboratories and one smaller group (for administrative data processing). Each lab consists of one or a few graduated (Ph.D.) researchers, five to ten (typically) graduate students, and some lab-specific support staff. From the department's point of view, the laboratories are the units which perform research projects, teach graduate courses, and are responsible for finding their own funding. From the graduate student's point of view, the laboratory is his or her organizational "home". The thesis project is done in one's own laboratory, but the graduate student must take courses across the range of all the laboratories.

The research program is coordinated by the research committee, headed by Erik Sandewall. The current laboratories are:

ACTLAB (Lingas) for complexity of algorithms
AIELAB (Tengvald) for artificial intelligence environments
ASLAB (Hägglund) for application systems,
CADLAB (Kuchcinski) for computer-aided design of digital systems,
LIBLAB (Hjerppe) for library and information sciences.
LOGPRO (Maluszynski) for logic programming,
NLPLAB (Ahrenberg) for natural language processing,
PELAB (Lennartsson) for programming environments,
RKLLAB (Sandewall) for representation of knowledge in logic,

The group for *administrative data processing* (Goldkuhl), although primarily a group for undergraduate teaching, also includes some research activities.

The laboratory system is an intermediate form between the "flat" university department and the "formally structured" one. In the "flat" department there is in principle no organization, just a number of professors each of which is the advisor for a number of graduate students. The laboratory structure encourages, and makes visible those cases where several professors /researchers

/advisors work jointly with their research and their students. In particular, faculty members who do a lot of work for undergraduate teaching find it convenient to be a member, but not a leader of a research laboratory. Also, a visiting scholar would be a member of an existing laboratory and would not form a new one.

The "formally structured" department is the one where the academic positions (several professorship levels, lecturer, etc.) define the hierarchical structure of the department. This has often been the traditional organization in Swedish universities. The laboratory structure at IDA is more uniform. It is also easier to change, since the department's decisions about changing laboratories (adding, deleting, splitting, or merging them) can be taken according to the needs of the research activities. The creation of a senior position does not automatically imply the creation of an organizational unit, nor the other way around either.

1.2.2 IDA:s organization in general.

IDA:s general organization is described in more detail in appendix A. The department is lead by a department board (institutionsstyrelse), whose chairman ("prefekt") is Bengt Lennartsson. The two main areas of activity are reflected in the two subordinate committees:

- the committee for undergraduate teaching, whose chairman is Anders Haraldsson;

- the research committee, whose chairman is Erik Sandewall.

The research committee equals approximately the set of laboratory leaders, and is responsible for all aspects of the department's graduate education programs and research.

The organizational groups within the department are:

- the research laboratories, headed by the lab leaders;

- two undergraduate teaching groups, one for the teaching in the School of Engineering ("tekniska högskolan"), and another for the teaching in the School of Arts and Sciences ("filosofiska fakulteten"). The teaching groups are each headed by a "studierektor", namely Anders Haraldsson and Lise-Lotte Raunio. The teaching groups report to the undergraduate teaching committee;

- a technical support and service group (TUS), which is headed by Anders Aleryd and reports directly to the department board.

The department's resources are almost consistently measured in monetary units, kronor, and not as e.g. "positions" or "slots" for teachers. For example, the School of Engineering buys a number of courses from the department, for a

price that is set in kronor. The "studierektor" uses the money partly for paying people in his own teaching group, and partly for sub- contracting research labs to do some of the courses. The laboratory leaders see a number of distinct sources of income, such as sub-contracted courses, research grants, and industry cooperation, and must make the ends meet.

Through this organization, we try to de-centralize responsibilities within the department with a minimum of bureaucracy, and without sacrificing the advantages of joint strategical planning and continuous synergy effects between the different parts of the department. The organizational and economic structure defines a small set of "rules of the game", and the task of the laboratory leaders and laboratories is to maximize the lab's performance according to the criteria that were discussed in section 1.1.1, and within the constraints of the rules.

1.3 Educational programmes

Industry representatives often point out that teaching the next generation of engineers, "knowledge engineers" and systems analysts is the most important knowledge transfer activity for a university. For IDA, it accounts for roughly 45% of the total budget, whereas knowledge transfer directly to industry accounts for about 10%, and research accounts for the other 45%.

Before giving an overview of our undergraduate educations for readers familiar with the Swedish educational system, the next section briefly reviews the system, for the benefit of the international reader, and using terms from the U.S. educational system for comparison.

Information about graduate education is given in appendix B.

1.3.1 Review of teaching organization

Students are admitted to the university after having completed senior high school with a matriculation exam, usually the year when the student is 19 years old. The student is admitted to a specific "study line", which defines what he or she will be majoring in. The school of engineering has the following study lines:

Mechanical Engineering
Electrical Engineering and Applied Physics
Engineering and Economics
Computer Engineering
Computer Science

The first four of these study lines are nominally for 4 1/2 years, in practice often more. They lead to the degree which in Swedish is termed "civilingenjör", but which is used for all branches of engineering, not only for civil engineering in the English language sense. It is comparable to a Master's degree, with the

qualification that the concluding thesis project is of moderate size - about three months' work - and is usually performed in industry and not as a research assignment. The courses in the study line are almost exclusively of a technical character; the student is assumed to have acquired the necessary knowledge of "Western Civ", foreign languages, etc. before matriculation.

For the Computer Science study line, which is a knowledge engineering type education, the nominal study time is 4 years instead of 4 1/2. Study lines in the School of Arts and Sciences are often for 3 years, and end with a Bachelor of Science degree.

In the School of Engineering, the study lines are not directly tied to departments. There is a matrix organization, where each study line buys courses from (at least potentially) all the departments. In particular, IDA sells courses to all five study lines. When we refer to courses in the "undergraduate" eduction in this volume, we really mean the courses in these study lines leading to a Master's or a Bachelor's degree.

All the students in a study line take the same courses (with minor exceptions) during the first two years, and have a free(-er) choice from the third year onwards.

The students who go to graduate school must have completed the 'undergraduate' degree with (in principle) 60 points of computer science courses. One full-time academic year is 40 points so one point is roughly one work-week. The graduate study proceeds through two successive levels, the "teknologie licentiat" degree nominally after two years, and the "teknologie doktor" (Ph.D. in engineering) nominally after two additional years. Both levels require a combination of coursework and a thesis. The "tek.lic." can therefore be seen as an advanced Master's degree with a substantial, research oriented thesis.

There are two reasons for having the licentiate degree in the system. For those students wishing to go into industry, it is a good break point in the education. Industry wants people to be as young as possible when they come, and the licentiate has already had a participation in research which is a sufficient background for many industrial jobs. Secondly, for students who contemplate whether to enter a research education at all, the Ph.D. seems a very long way off, and the licentiate is a more immediate and tangible goal. Many of the licentiates continue towards the Ph.D. but appreciate having had the breakpoint.

Students that come from the 4-year Computer science study line or from the 3-year Systems science line (Computer science and business administration) go to "filosofie licentiat" and "filosofie doktor" degrees. Three years on those lines gives 60 points in computer science, so the 2+2 years research education is counted from the end of the third year. Thus most of the courses in the fourth year of the Computers science study line are valid for graduate studies also. Students are encouraged to complete the study line, independently of any

graduate studies. Stipends for graduate studies will normally not be available until after that fourth year of study.

1.3.2 Undergraduate curricula

The Linköping University has since 1975 had a strong position in undergraduate curricula and teaching in computer science. Linköping is today the only university which offers the three main 3-4.5 years undergraduate study programs in the area of computer science and systems analysis. An increasing part is also other educational activities such as a continuing education programme in computer science for Swedish industry.

As the first institute of technology in Sweden we started 1975 the D-line (Computer Science and Technology - "Datateknik-linjen") as a four-year (now converting to 4.5 years) programme leading to a Master of Engineering. It was the first full and specialized programme in computer science, specialized on software and hardware. The programme was introduced 1982 at all other Swedish institutes of technology.

Many persons at IDA made substantial contributions during the development of the D-line and about 25% of the courses are given by IDA. The expansion of staff and graduate students at IDA during the period 1980 - 1985 is to a large extent a result from recruiting students graduated from the D-line. The number of students accepted annually to the line has grown from 30 students the first year to 120 students.

A new computer science programme, the C-line (Computer Science - "Datavetenskapliga linjen") was started in 1982. It is a four-year programme leading to a Master of Science degree. The number of students accepted annually is 30. The programme is also given at Uppsala and Umeå Universities. This new programme is at Linköping in the school of engineering, but differs from ordinary engineering curricula (such as electrical engineering, or mechanical engineering) in some significant ways:

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

= LISP as the first programming language

= relevant humanities, such as psychology and linguistics, are significant parts of the curriculum, and are introduced as basic courses during the first years

= courses in theoretical branches of computer science

= courses in AI and AI-oriented subjects

The major part of the C-line was developed by persons from IDA and most courses are given by us. It is 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.

The D-line will from last year pass through some changes. The new base with discrete mathematics, logics and to start programming with LISP instead of Pascal will be introduced there. An advantage is that the students from the C- and D-lines get the same basis and we expect a large number of students from the D-line to be better prepared to specialize their studies in both more theoretical computer science areas and in artificial intelligence. In the D-line there is also a specialization for telematics, relying partly on our research in interactive systems and office systems.

The set of courses that are available in the other programmes 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 curriculum available to other lines as well.

The mechanical engineering programme has been extended with a new specialization that combines mechanical and computer engineering. We believe that especially research in artificial intelligence will be significant within that specialization.

The School of humanities and sciences offers since 1977 a three-year programme in System analysis. The number of students accepted annually is 60. The programme is given at several other Swedish universities and colleges as well.

This programme 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. IDA is responsible for the major part of courses in the curriculum.

1.3.3 Continued education for Swedish Industry

Our programme for "continued education" of engineers in computer science is rapidly expanding. The programme was developed in cooperation with a coalition of Swedish engineering industry (Oktogonen). The aim is to renew the knowledge basis in computer science for engineers working with programming. Often they are hardware-oriented engineers. Several are leaders for groups and sections in the organization. Responsible for the programme from IDA is Anders Haraldsson.

The courses are given as academic courses and give academic credits after normal examination. They are organized for half-times studies and are given in such way that the participants are free for studies 2 days a week with one full-day teaching and one day for reading and exercises of their own.

The programme consists of a base-part Computer Science 20 points with the following courses:

= Discrete Mathematics 6 points (corresponding to 12 weeks half-time studies)
= Data Structures and Algorithms 4 points
= Principles of Programming and Programming Languages, 10 points

and two additional parts, each approx 10 points

= Programming in LISP and Prolog, 5 points
= Artificial intelligence, 5 points

= Distributed Systems, 4 points
= Computer Network, 3 points
= Operating System, 2 points
= Databases, 2 points

Thirteen course programmes, varying in size up to 28 points, have been given during 1987 with Ericsson Telecom, Ericsson Information Systems, ASEA, Saab Scania, Ellemtel and Ericsson Radio Systems as major participants.

1.3.4 Other programmes

Among other educational activities we can mention a 25 points programme given in the area of AI/expert systems in the form of a *knowledge engineering training programme*, which covers the theoretical basis needed in that area. The programme was developed 1986 in connection with the knowledge transfer program, KTP, (where also practical issues in knowledge engineering are covered). Participants have been both external from industry (Volvo, Saab Scania and Asea Atom) and KTP-members (Asea and Philips).

The programme consists of the following courses:

= Discrete mathematics
= Logics
= AI programming languages (LISP Prolog)
= AI cognitive structures
= AI knowledge representation
= Expert systems
= Project work with an expert system tool.

During 1987/88 the programme is given directly for ASEA Brown Boveri in Västerås, administred by the local college.

A Swedish International University (SIU), giving Master of Science Programmes, will start operation in 1988. Several Swedish universities will set up special Master's programmes using the English language as medium of instruction. Responsible for the initial work with the SIU is Harold Lawson at IDA, who is on leave for a position at the Swedish Board of Universities and Colleges (UHÄ).

Linköping University and IDA will offer in SIU a Master's programme in Computer Science - Knowledge Engineering. The programme will be similar to the 25 points program mentioned above. The plan is to start during 1989.

1.3.5 Organization

The undergraduate education at IDA is organized as follows:

The Committee for Undergraduate Education (IDUN - IDA's undervisningsnämnd), headed by Anders Haraldsson, is responsible for the contents of courses given by the department and the planning of teachers for the courses. There are representatives from the student unions in the board.

The department is responsible for the subject areas computer science (datalogi), telecommunication and computer systems (telesystem) and administrative data processing (administrativ databehandling).

There is a responsible director of undergraduate studies (studierektor) for each subject area. The computer science area is further divided into subareas. The directors are

Anders Haraldsson, computer science
Mikael Patel, telecommunication and computer systems, system programming
Arne Jönsson, artificial intelligence
Mats Wiren, natural language processing
Rolf Karlsson, theoretical computer science
Lise-Lotte Raunio, administrative data processing

The directors and many of the courses belong closely to a research laboratory and the division of areas of responsibilities reflects to a great extent our laboratory organization.

Appendix C lists the courses given during the academic year 1986/87, teaching personal and computer facilities for undergraduate education.

1.4 Knowledge Transfer Activities.

A research department produces and exchanges new knowledge. In order to flourish, it must itself produce new results, and also participate in the international "barter trade" for research results. The useful outcome of those

activities, from the point of view of the taxpayers and the sponsoring agencies, is when the accumulated knowledge is transferred to practical use. We use the term "knowledge transfer activities" collectively for the various ways of transferring accumulated research knowledge.

1.4.1 Knowledge Transfer Program to industry.

There is an increasing awareness in Swedish industry about the need for continuing education, and for transferring new research results directly to corporations. IDA:s activities in this area were reported in section 1.3.3.

An important task for a university department is to disseminate knowledge into the surrounding society, public sector, trade and industry. This means that the research organization should serve as a source of competence, bringing together and distributing not only its own results but also importing and collecting state-of-the-art information from the international research community.

The main channel for effectuating this task is obviously the knowledge transfer that results when people trained in undergraduate and graduate study programs enter working positions outside the university. Less efficient but equally important is the spreading of results through written reports and oral presentations by active researchers. A third way of achieving technology transfer is through cooperative work in joint projects.

In our department we have actively pursued these strategies, e.g. by issuing a special series of reports summarizing important results in central research areas specifically directed towards industry, by arranging and participating in tutorial conferences, (recent tutorials include *Artificial Intelligence, Software Development Environments, Prolog Programming Environments, AI and Expert Systems, Advanced Human-Computer Interaction*), by developing continuing education programs for industry and by direct consultations and cooperation in applied projects.

However, we felt that in many cases these methods were too slow or to restricted in order to achieve an effective technology transfer in rapidly developing areas of strategic importance for industry. Thus we initiated some years ago a discussion with industry about this problem which led to the decision to start a special *knowledge transfer program*, KTP, in 1984.

The goal of this program is to 'inject' competence derived from research into the existing industrial organization. The method is that typically two persons, located on a middle level in the organization, come to our department for a period of one or a few years, in order to learn new technology, and return to their organization after that time. The participating company also pays a yearly contribution that helps pay for researchers (particularly guest lecturers) and equipment.

Since the start in 1984 six companies have joined the programme, namely in the order of the time for their affiliation: S-E-Banken, Ericsson, ASEA, Alfa-Laval, Philips and Sandvik Coromant. All these are large multi-national manufacturing companies, with the exception of S-E-Banken, which is a major Swedish bank.

The programme has a budget of its own, but participating individuals are associated with one of the research laboratories in the department. Each participant is assigned a faculty supervisor and one graduate student, who is supposed to work closely together with the participant. Courses and other training activities are organized and coordinated at the department level.

1.4.2 KTP as a knowledge engineer training programme

The major area of interest for the companies participating in the knowledge transfer programme has been AI and expert systems. One ambition is to provide a reasonable training in knowledge engineering and expert systems development for people from industry. It is our experience that a one year combination of a half time course programme and about half time participation in applied work on expert systems development provide a good basis for continued undertakings in this area.

Our knowledge engineer training programme tries to fill the gap between commercially available tutorials including at best a course programme for a number of days or a few weeks, and the academic 4 year computer science curriculums in the area. We thus offer, in addition to the KTP project participation with apprenticeship in research, training projects and tools evaluation activities, a one year half time course programme including:

- Introduction to AI and expert systems
- Discrete mathematics
- Logic
- AI programming systems
- AI - cognitive processes
- AI - knowledge representation
- Expert systems methodology and tools

In addition to being part of KTP, this programme is also offered either in Linköping for participants from different companies (who spend 3 days in Linköping every other week) or for a full group of 20-30 participants from a single company. In this case most of the training is given on site.

1.4.3 Experiences and plans for further activities

As far as we can judge the knowledge transfer programme has become a success. However, there are also aspects which have turned out to be more problematic than we anticipated.

Before the programme was started, we analyzed the experiences of previous efforts to work together with industry trying to achieve an effective technology transfer. It seemed that a crucial problem in many cases had been the tendency for industry participants to give a higher priority to short-term obligations than to research-related work. This often gave as a result that projects did not achieve a critical mass of activity and thus were felt as a waste of time by participants from both sides.

To handle this problem we decided to concentrate our knowledge transfer activities to a small number of partners, who showed a definite commitment to the cooperation, e.g. by allocating a substantial amount of financial support to the programme. However, it still seems that the problem of allocating personnel at an appropriate competence level to programme participation delayed the start of activities in many cases.

Most companies have chosen to work with two KTP participants. Typically one of them is a more experienced person with a PhD (although not in computer science) and an established position in company. The other person is often younger and with a good background in computer science. The period of participation for a person tends to be about one year on a half time basis, visiting Linköping 2-3 days each week. Courses and other KTP activities are concentrated to certain days during a week and thus planned to meet the needs of part time visitors.

The time for active participation in Linköping varies between one and three years for participating companies. For companies that have completed the knowledge transfer programme we offer a less expensive associated membership, which does not include active project work in the department but cooperation in various forms. Similar agreements on joint efforts have also been made with Volvo, Pharmacia and Ericsson Telecom.

We are presently working on a further development of the kind of activities offered within an associated membership in KTP, for instance in connection with LAIC and CENIIT as described below. It is a general experience that a great effort has to be spent during the initial phases of cooperation between the university and a particular company. In the successful cases such a cooperation is rapidly evolving into a situation, where the infrastructure of communication and personal contacts exist, and where a mutual understanding improves the effectiveness of the joint activities. The KTP programme has served well to establish a foundation of such contacts between our department and a number of companies.

1.4.4 LAIC - Linköping AI Consortium

Current efforts also include the establishment of LAIC, Linköping AI Consortium, which incorporates the cooperative research programme in AI with the Defence Research Institute and participation in the AI subproject of the Prometheus Eureka project together with the Swedish car manufacturers

Volvo and Saab, as mentioned above.

One major theme for R&D efforts within LAIC is *plan-guided systems*, as described in chapter 10 of this report, with applications for autonomous vehicles or for automated manufacturing cells. Examples of other areas to be pursued within LAIC are geographical information systems and systems supporting strategical decision making.

1.4.5 CENIIT - The Center for Industrial Information Technology

During 1988 a special competence center in industrial information technology will be formed at Linköping University, with the intension to improve the possibilites to utilize advanced information technology in industrial processes and products. Researchers from different areas and departments will work together in projects within this center. We believe that this will provide excellent opportunities to further develop and integrate knowledge engineering methodologies with front-line domain expertise in areas such as computer-aided engineering, robotics, process control and office information systems.

The following areas is planned to be pursued as our department's contribution to CENIIT:

- *Plan-guided systems for autonomous vehicles and computer-integrated manufacturing.* This area will for the time being be covered by existing projects in IDA's laboratories and additional CENITT funding is not claimed. (Erik Sandewall)
- *Geometrical algorithms.* The purpose of this activity is to develop competence in the area *efficient algorithms for three-dimensional geometrical problems and the planning of movements*, in an active communication with applied projects in other departments. (Rolf Karlsson)
- *Engineering databases.* This area covers issues related to database technology, with a special emphasis on databases to be used in support of design, development and maintenance of (large) technical systems. Here is included design support for as well mechanical engineering as software or knowledge engineering. Of special interest are object-oriented databases, extensions to first-normal-form relational databases, and integration with information retrieval technologies for document management. Activities involve several laboratories in IDA. (Area coordinator: Sture Hägglund)
- *User Interface Management Systems.* This area covers certain aspects of human-computer interaction, with a special emphasis on tools for design and management of user-computer dialogues. During the planning period, Ralph D. Hill from ECRC in Munich has visited IDA several times for discussions. Possible activities relate to several labs in IDA. (Area coordinator: Sture Hägglund)

1.4.6 Spinoff Companies.

IDA's policy is to accept industry contracts for knowledge transfer, i.e. for work where the customer wants (his employees) to acquire knowledge in some area, but not to accept consulting jobs or other projects where the customer wants software, hardware, or designs to be delivered. In such cases we refer to existing spinoff companies, and we may also encourage IDA employees to form new spin-off companies in order to catch an opportunity.

The significance of university spinoff companies for industrial growth is well known. One part of the previous artificial intelligence laboratory split off a few years ago and formed Epitec AB. The company has presently about 25 employees and is active in development of tools for knowledge systems development, in development of applications for clients in industry, in consulting and in training of knowledge engineers.

As an even earlier spinoff, Jerker Wilander and Kenth Ericson founded the company Softlab AB in Linköping in 1981. Softlab is working in the areas of compiler design and advanced tools for software development The company is growing steadily and is also expanding its scope of applications.

Other spinoffs, primarily from CADLAB, are Grafitec AB, founded by Michael Pääbo and active in business graphics, and DIGSIM led by Bengt Magnhagen.

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) recruited heavily from our department. There are also a number of software companies founded by former students, such as e.g. Programsystem AB (programming support environments, user interface management systems, datacom), having close contacts with the department and active in transferring software originating from research projects into commercial products. The intensive communication with the many developing high tech software companies around the university is a vitalizing force for the department.

1.5 International Cooperation.

In computer science, like in most other disciplines, the most important international cooperation is the informal one. It takes place through personal contacts and visits, and at international conferences.

In addition, IDA or specific labs within IDA also participate in a number of organized international projects, namely:

- the SYDPOL programme, a Scandinavian project on SYstems Development and Profession Oriented Languages, together with the universities in Aarhus (Denmark) and Oslo (Norway). Formal activities in the programme are presently at a lower level than a few years ago, but the programme has

resulted in ongoing contacts with for instance visiting researchers and students.

- the COST-13 project, a European project on Computer Architectures for A.I., together with the Free University of Brussels (Belgium), the University of Rome 'La Sapienza' (Italy), and Delphi S.A. in Pisa (Italy).

- the PROMETHEUS project, which is part of the European EUREKA programme, and which has now gone through its planning phase. Prometheus is concerned with future traffic and automobile systems and is a joint effort by the European car manufacturers. Erik Sandewall is the European coordinator for the AI subproject, PROART.

IDA also has extensive contacts with university and industrial research laboratories, primarily in USA but also in Europe and to a certain degree also in the Far East. For instance, IDA regularly send students to the Xerox Palo Alto Research Center in USA for periods of 3-6 months.

1.6 Research Facilities.

IDA moved into a new functional building in 1985. However, one third of the department's personnel had to be left behind in another building. Some laboratory space in the new building also had to be used as offices. We foresee serious problems with office and laboratory space in the near future.

In May, 1986, IDA was awarded a grant from Rank-Xerox consisting of 18 Xerox workstations, plus additional file servers, print servers, and other computer equipment including software. This grant covered the needs expressed in an invited application and it was one of the three large grants awarded in Europe. It represents a valuable contribution to our activities, including the possibilities to provide advanced workstations for the undergraduate and masters-level education. The total number of Xerox workstations in IDA is presently 32.

During the year we have also expanded the installed base of SUN-3 workstations to 15, with a further expansion planned in the near future.

Although we have a relatively favourable situation with respect to computer equipment at present, there are still some unresolved problems. The DEC-system 2060, which we use as a 'background' computer resource and as a tool for office services, has far exceeded its technical and economical life-length and must be replaced as soon as possible.

More details on equipment are given in appendix D.

2.

ACTLAB
The Laboratory for Complexity of Algorithms

Andrzej Lingas

2.1 Introduction.

The Laboratory 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 laboratory believe that work on algorithm and data structures efficiency is no less important than the development of new programming methodologies, or new faster computers.

The laboratory is a continuation of the so called Group for Complexity of Algorithms which in turn originated from a part of the former Group for Theoretical Computer Science in the spring of 1985. Since then, the members of the laboratory have published more than twenty papers in international journals and conference proceedings on computer science. In 1987, the laboratory acquired a new graduate student, Sven Moen, whereas its first graduate student, Christos Levcopoulos, defended his Ph. D. thesis entitled *"Heuristics for Minimum Decompositions of Polygons"* .

The third year of the laboratory (group) research was mainly spent on a project called Efficient Algorithms and Data Structures for Geometric and Graph Problems funded by STUF and STU. The objectives of the project fall into three mutually interrelated categories of data structures, computational geometry and graph algorithms. They are structured as follows:

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

Computational Geometry:

Computational Geometry on a Grid
Theoretical Aspects of Robotics: Traversing a Maze with a Robot Arm
Heuristics for Minimum Decomposition of Planar Figures
Voronoi Diagrams with Barriers and Minimum Weight Triangulation
Recognizing Polygons

Data Structures:

Data Structures Requiring Minimal Changes per Update
Inherent Cost for Maintaining Data Structures
Heuristics for Optimal Search Trees

Graph Algorithms:

Fast Parallel and Sequential Algorithms for Subgraph Isomorphism and Homeomorphism
Efficient Solutions for Weighted Graph Problems

The considered problems have applications in VLSI chip design and fabrication, graphics, robotics, numerical analysis (in particular, terrain interpolation), chemistry and optimization.
In 1987, several new problems within the above project have been attacked by the laboratory members. Here we mention only the most important: the design of search trees with constant cost of structural changes per update (see Section 4.3.2), and the design of superfast parallel algorithms for subgraph homeomorphism, disjoint connecting paths and other related problems restricted to non-trivial graph families (Section 4.3.3).
The new student, Sven Moen, has started research on algorithmic aspects of text editing and maintained his interest in efficiently drawing graphs (he devoted his "examensarbete" to the latter topic). The other student, Ola Petersson, together with dr. Christos Levcopoulos has obtained interesting results on the problem of dividing a rectangle into a minimum number of squares related to electrical circuit design (see Section 4.3.1).

In addition to the research, the laboratory undertakes important consulting and educational tasks on aspects of algorithm analysis and complexity theory within the department, and is open to cooperate with other laboratories and departments. For instance, the laboratory members have recently become interested in the problem of interpolating two-dimensional surfaces in three-dimensional space with triangular nets raised by the group of dr. Fahlander at the Department of Electrical Engineering.

2.2 Laboratory Members

Laboratory leadership : Andrzej Lingas, Ph.D.

Bodil Mattsson-Kihlstrom, secr.

Supervisors:

Rolf Karlsson, Ph.D.

Christos Levcopoulos, Ph.D.

Graduate Students:

Sven Moen

Ola Petersson

2.3 Current Research

2.3.1 Computational Geometry

Computational Geometry on a Grid (Rolf Karlsson)

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 connected components, and computing all line segment intersections. The solutions are based on an efficient point location algorithm or use a new data structure, the interval trie, when sweeping the plane with a line. For problems in higher dimensions, we use a divide-and-conquer technique until the dimension is reduced to 2 (or 3) where the sweep-line algorithms apply. 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 to 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. Much of this research is joint work with Dr. Mark Overmars of Utrecht University. Some of the results have been published and some are submitted for publication.

Theoretical Aspects of Robotics (Rolf Karlsson)

Motion planning, which tries to move an object from one position to another, while avoiding obstacles, has attracted much interest in recent years. In this preliminary investigation we consider the problem of efficiently propagating a linkage through an orthogonal maze. The linkage moves strictly in two dimensions, no crossing is allowed. Given the layout of a 2-dimensional maze

with corridors of width w, we express the time complexity of moving a robot arm (linkage) of k equal-length links through the maze using a basic 2-joint motion. We give an algorithm which pushes a linkage with link length *l= 2sqrt(2) w - epsilon* around one corner in *O(log(w/epsilon))* time, combine this with the complexity of moving along the corridors, and generalize it to apply for maze routes with corridors of different widths. The derived traversal complexity we can use to efficiently find a shortest route.

Heuristics for Minimum Decompositions of Polygons
(Christos Levcopoulos)

The above heading is also the title of Levcopoulos's thesis published and defended in 1987. The following problems of minimally decomposing polygons were considered: (1) decompose a polygon into a minimum number of rectangles, (2) partition a polygon into rectangles by inserting edges of minimum total length and (3) partition a polygon into triangles by inserting a maximal set of non intersecting diagonals, such that their total length is minimized.

The first problem has an application in fabricating masks for integrated circuits. Tight upper and lower bounds were shown for the maximal number of rectangles which may be required to cover any polygon. Also, a fast heuristic which achieves these upper bounds was presented. Further refinements of the results appearing in the thesis were achieved during 1987, and were already published.

The second problem has an application in VLSI design, in dividing routing regions into channels. Several heuristics were proposed in the thesis which produce solutions within moderate constant factors from the optimum. Also, by employing an unusual divide-and-conquer method, the time performance of a known heuristic was substantially reduced.

The third problem has an application in numerical analysis and in constructing optimal search trees. Here, the contribution of the thesis concerns an analysis of the so called greedy triangulation. It is one of the most known heuristics for minimum weight (length) triangulation of planar figures (the latter has applications in interpolation of two-argument functions and finite element method). It consists in iterating the following step: insert a shortest diagonal of the input figure that does not intersect those already in the plane. In the thesis, it has been shown that there is a constant c such that for any polygon, with or without holes, with w concave vertices, the length of any greedy triangulation of the polygon is no longer than c(w+1) times the length of a minimum weight triangulation of the polygon. An independent proof of this fact for convex polygons has been given by Andrzej Lingas. On the other hand, it has been shown that for every integer n greater than 3, there exists a set S of n points in the plane such that the greedy triangulation of S is Omega (sqrt(n)) times longer than the minimum weight triangulation, improving the previously known lower bound substantially. Finally, a simple linear-time algorithm for computing the greedy triangulation of the so called semi-circular

polygons has been presented in the thesis. The above results have been partly published.

Voronoi diagrams with barriers, the shortest diagonal problem and minimum weight triangulation (Andrzej Lingas)

Planar figures which are called planar straight-line graphs (PSLG for short) are considered. A PSLG G is a pair (V,E) such that V is a set of points in the plane and E is a set of non-intersecting, open straight-line segments whose endpoints are in V. The points in V are called vertices of G, whereas the segments in E are called edges of G. If G is a simple cycle, it is just a (simple) polygon. If G has no edges, it is a planar point set. A diagonal of G is an open straight-line segment that neither intersects any edge of G nor includes any vertex of G and that has its endpoints in V. A triangulation of G is a maximal set of non-intersecting diagonals of G. The concept of the greedy triangulation (see the previous section) can be easily generalized to include any PSLG.

(1) To solve the problem of finding a shortest diagonal of a PSLG, a new generalization of Voronoi diagrams of planar finite point sets to include diagrams of PSLG's has been used. The Voronoi diagram with barriers of G (Vorb(G) for short) is a net that together with G partition the plane into the regions P(v), v in V, such that a point p is inside the region P(v) if and only if (p,v) is the shortest straight-line segment connecting p with a vertex in V that does not intersect any edge in E. Wang and Shubert independently considered such diagrams and showed that they can be constructed in time O(n log n)

A simple characterization of shortest diagonals of G in terms of Vorb(G) has been provided. This combined with the algorithm of Wang and Shubert yields O(n log n) time solution to the shortest diagonal problem.

Our method of finding a shortest diagonal of a PSLG implies that the greedy triangulation of a planar point set or any PSLG can be computed in time O(n**2 log n) and space O(n). The most efficient known algorithm for computing the greedy triangulation of a planar point set which is due to Gilbert takes O(n**2 log n)-time and O(n**2) space. Instead of computing the Voronoi diagram with barriers of the current PSLG from scratch every time after inserting a shortest diagonal, the current diagram can be adequately updated. Although this approach does not yield better worst-case complexity bounds, it is provably much more efficient in the average.

Further important applications of Voronoi diagrams with barriers can be found in planar nearest visible neighbor, shortest-path, minimum spanning tree and matching problems in the presence of barriers.

(2) Complexity analysis of another, polynomial-time heuristic for minimum weight triangulation of planar point sets has been improved. The heuristic first constructs a polygon whose vertices are all points from the input set. Next, a minimum weight triangulation of the polygon is found by dynamic programming. The union of the polygon triangulation with the polygon yields a triangulation of the input n-point set. A non-trivial upper bound on the worst case performance of the heuristic in terms of n and another parameter has been

derived. Under the assumption of uniform point distribution it has been observed that the heuristic yields a solution within the factor of O(log n) from the optimum almost certainly, and the expected length of the resulting triangulation is of the same order as that of a minimum length triangulation.

The obtained results have been already either published or submitted for publication.

Recognizing Polygons (Andrzej Lingas)

The joint work with Dr. Dean of Northern Bell Research and Dr. J. Sack of Carleton University on the so called pseudo star-shaped polygons started in 1986 has been continued. A polygon is pseudo star-shaped if there is a point from which we can see/eavesdrop its whole interior provided that it is possible to see/hear through its single edges. The class of pseudo star-shaped polygons generalizes and unifies the well known classes of convex, monotone and pseudo star-shaped polygons. An algorithm for constructing all regions from which the polygon is pseudo star-shaped running in quadratic time has been improved. On the other hand, a corresponding quadratic lower time-bound has been provided. The above results have been already accepted for publication.

2.3.2 Data Structures

Data Structures Requiring Minimal Changes per Update
(Christos Levcopoulos)

A basic set of operations on data consists of so called *dictionary* operations and *nearest neighbor* queries. These can be described as follows. Given is a set of real (or integer) keys S. The keys in S have to be stored such that for any query number q, the key in S which is closest to q can be efficiently computed. Such queries can be interleaved with operations which delete or insert keys into the set S. The aim is to minimize the time required for answering queries and performing updates (insertions and deletions), while also keeping the size of the data structure small. Another aim has been to minimize the number of changes in the data structure per insertion and deletion, i.e. the number of memory cells which have to be written on.

A reason to minimize the number of changes is that such data structures can be applied to implement so-called "finger" trees, where queries are answered very quickly in the vicinity of some specified nodes, called "fingers". We can find another application in shared environments (either with many users or with many processors). In such environments it is possible for many users (or, processors) to read the same file simultaneously, but the file can be changed by at most one user at a time. Therefore it is desirable to minimize the number of changes required.

It has been an open problem whether we can achieve a *worst-case* , constant number of changes. Together with Dr. Mark Overmars, expert in dynamic data structures, we discovered a way to show that this can be achieved. The starting idea is to use a balanced binary tree, where every node corresponds to a sequence of at most O(log n log n) keys. The proof is presently fairly complicated, concerning both implementation details and combinatorics. However, we have ideas for simplifying the method, and we hope to find simple and practical algorithms for this problem.

Inherent Cost for Maintaining Data Structures (Rolf Karlsson)

This research is two fold. The first part looks at 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 have 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. Current research tries to further unify these problem-oriented techniques, and to make the lower bound model we have introduced more general. In part, this is joint research with Dr. Ian Munro, University of Waterloo, and Dr. Ed Robertson, Indiana University.

Another problem we study is proving an adversary-based *Omega (klogk)* lower bound for finding the *k*th smallest element in a large heap. This would then prove a known algorithm as optimal. This research is conducted together with Dr. Thomas Strothotte, Stuttgart University.

Optimal Search Trees and Optimal Partitions of Polygons

(Christos Levcopoulos, Andrzej Lingas)

The research on optimal binary search trees with zero key access probabilities (with applications eg. in point location), started in 86, has been continued. It has been shown that for an arbitrarily small positive constant *e* there exists a linear-time heuristic for such search trees, producing solutions within the factor of *1+e* from the optimum. Also, by using an interesting amortization argument, a simple and practical, linear-time implementation of a known greedy heuristic for such trees has been given. The above results have been obtained in a more general setting, namely in the context of minimum length triangulations of so-called semi-circular polygons. They have been carried over to binary search trees by proving a duality between minimum weight partitions of infinitely-flat semi-circular polygons into *m*-gons and optimal *(m-1)*-way search trees. This duality has also helped to obtain better heuristics or algorithms for minimum length partitions of polygons using known algorithms for optimal search trees. In particular, it has been shown that a minimum length partition of a simple polygon into *m*-gons can be found in time *O(n**3m**2)*, and if the polygon is convex, in time *O(n**3logm)*. The above

results have been partly obtained in cooperation with Dr. J. Sack, and they have been already published in conference proceedings (ICALP) and in a journal. An extended version of the conference paper paper has been submitted to a journal.

2.3.3 Parallel and Sequential Graph Algorithms

(Andrzej Lingas)

The problems of subgraph isomorphism and subgraph homeomorphism belong to the most general NP-complete graph problems and they have many applications in computer science. The former problem consists in determining whether a graph is isomorphic to a subgraph of another graph. The latter problem is, given two graphs, decide whether a subgraph of the second graph is homeomorphic to the first graph, i.e. whether there is a subgraph of the second graph that becomes isomorphic to the first graph after contracting some of its paths with inner vertices of degree two to single edges. If the first graph is fixed, the subgraph isomorphism can be trivially solved in polynomial time whereas the status of the fixed subgraph homeomorphism problem is open in general. The fixed subgraph homeomorphism problem is closely related to the following problem of k disjoint connecting paths: Given k pairs of vertices of a graph, find k mutually vertex disjoint paths respectively connecting paired vertices. Note that the latter problem can be seen as a generalization of several routing problems including VLSI routing. We have shown that the following restrictions of the above problems are in the class NC, i.e. can be solved by parallel algorithms running in poly-log time using a polynomial number of processors:

i) The subgraph isomorphism problem for two-connected outerplanar graphs (the first example of a restriction of subgraph isomorphism to non-trivial graph family shown to be in NC).

ii) The fixed subgraph homeomorphism problem, and the related problems of k disjoint connecting paths and of graph recognition, all restricted to to any proper sub-family of planar graphs closed under the so called operation of minor-taking.

The above results have been obtained in cooperation with Prof. A. Proskurowski of University of Oregon. On the other hand, the first known polynomial-time algorithm for subgraph isomorphism restricted to two-connected series-parallel graphs has been designed in a cooperation with Prof. M. Syslo of Wroclaw University. Series-parallel graphs are a popular model of electrical circuits. Thus, the important problem of deciding whether a circuit contains a given pattern can be modeled as the subgraph isomorphism problem for series-parallel graphs. It has been also shown that the subgraph isomorphism problem for two-connected series-parallel graphs can be solved by a random parallel algorithm running in poly-log time and using a polynomial number of processors. The discussed results have been already accepted for publication.

2.4 External contacts

Rolf Karlsson: Participated in the *3rd ACM Symposium on Computational Geometry,* Waterloo, Canada, in June. Given seminars at University of Vila Real, Portugal, and University of Bergen. Finishing two papers with Dr. Mark Overmars, Rijksuniversiteit Utrecht, The Netherlands, and one paper with Dr. Ian Munro, Waterloo, Canada. Writing a paper with Dr. Thomas Strothotte, Stuttgart University, West Germany.

Christos Levcopoulos: Written a paper with Dr. M. Overmars, Rijksuniversiteit Utrecht, The Netherlands. Presented his common paper with A. Lingas and J. Sack at the *14th International Colloquium on Automata, Languages and Programming,* Karlsruhe, Germany, and other his paper at the *5th Conference on Foundations of Software Technology and Theoretical Computer Science,* Pune, India. At the latter conference, he also presented a paper by A. Lingas and A. Proskurowski.

Andrzej Lingas: during his visit at Department of Computer and Information Science of University of Oregon, USA, in May, gave a graduate course on computational geometry and written a paper with Prof. A. Proskurowski. Gave a seminar at University of British Columbia and participated in the *3rd ACM Symposium on Computational Geometry,* Waterloo, Canada, in June. Finished a paper and started new research with Prof. J. Sack of Carleton University who visited IDA in June and August. Presented papers at the *2nd International Conference on Combinatorial Mathematics and Computing,* Canberra, Australia, in August, at the *13th IFIP Conference on System Modelling and Optimization,* Tokyo, and at *International Workshop on Computational Geometry and Discrete Algorithms,* Osaka, Japan, in September. Also, he gave a seminar at University of Fukuoka, Japan. Written a paper with Prof. M. Syslo of Wroclaw University who visited IDA in September. Acted as an opponent and gave a guest seminar at University of Turku, Finland, in November.

Sven Moen: participated in the *14th International Colloquium on Automata, Languages and Programming,* Karlsruhe, Germany.

Ola Petersson: participated in the *3rd ACM Symposium on Computational Geometry,* Waterloo, Canada, in June.

On a domestic level, the group has maintained contacts through mutual visits with an active research group (Ferenc Belik, 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 87-88:

Computational Geometry

Analysis and Complexity of Parallel Algorithms

Lower Bounds Techniques

Algorithm Analysis and Complexity Theory

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

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

Mathematical Aspects of VLSI (84)

Search Structures (85)

Analysis and Complexity of Parallel Algorithms (86)

Amortized Computational Complexity (87)

Computational Geometry (87)

3.

AIELAB
The Artificial Intelligence Environments Laboratory

Erik Tengvald

3.1 Introduction

The laboratory for artificial intelligence environments (AIElab), formed in 1986, is one of the descendants of the original artificial intelligence laboratory formed in 1981. The AIElab continues the mainstream of the knowledge representation work initiated at the artificial intelligence laboratory, viz., design of programming systems for A.I., supporting applications selected from Mechanical Engineering[1].

The research outlook and current projects of AIELAB are based in the considerable experience gained in the previous knowledge representation work in AILAB. 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 the research in knowledge representation at the AIELAB is based in the considerable experimental experience of the OBS/PAUL project. An experience of maybe highest importance for later work is:

> *Geometric reasoning steps consume a large amount of computing resources. If the geometric reasoning steps take too long, the method of explorative programming [Sandewall 78] breaks down.*

Further work by Jalal Maleki [Maleki 86], now at the RKLLAB, and the experience of other researchers, as for example [Hayes-Roth 83] and [Forbus 87], has validated this our experience.

The work in AIELAB has mainly been supported by STU, The Swedish Board for Technical Development.

Explorative programming is central for building expert systems. Consequently, our experience indicates that the possibility of constructing expert systems with a substantial content of geometric reasoning is slim indeed. That is, if you do not increase the processing power of the hardware. This is precisely the intended mode of approach chosen in the AIM-project described below.

3.2 Researchers and Project

The AIELAB research interest is on one hand directed towards geometric reasoning or more precisely on the combination of symbolic and geometric reasoning. This research interest has been formalized in the artifical Intelligence for Manufacturing (AIM) project. The objective of the AIM project is to design an AI-environment[2] able to support expertsystems performing combined symbolic and geometric reasoning. The AIM project can be seen as a new experiment in the OBS tradition.

As noted above our previous results and general experience indicate that a substantial increase of the raw processing power of the hardware is necessary to make such an AI-environment feasible and usable. Such an increase in processing power is only possible by the use of parallell programming.

Consequently, AIELAB has parallell programming as another research area. During this and probably also during the next year, parallell programming is the area in which most if not all research has and will be concentrated. The parallell programming research is currently formalized in the freeshape subproject[3].

3.2.1 Laboratory members

AILAB members working in the AIM-project (eg. Freeshape subproject) during 1986 has been:

Erik Tengvald
Bernt Nilsson
Leif Finmo
Mikael Svensson
Anders Nyberg
Jonas Wallgren

3.2.2 The AIM project

The characterising trait of the reasoning necessary to solve the OBS problem of operations planning was the numerous and complex interdependencies between the geometric and symbolic steps in the reasoning process. Indeed, this interdependency between geometric and symbolic reasoning steps is in general a characteristic of the reasoning processes typical for the manufacturing

industries. Examples are the reasoning performed in, design, process-planning, and in many cases, robot-planning, jobshop planning and materials selection.

For this reason we have found it useful to coin a new word, *geombolic* reasoning, for reasoning processes characterised by such interdependencies between geometric and symbolic reasoning steps.

> *The objective of the AIM-project is to design an AI-environment in which it is not only possible but also easy to design expert systems executing geombolic reasoning processes.*

Due to the need of sufficient processing power such an environment has to reside on a parallel computer. Its overall structure will be (fig 1).

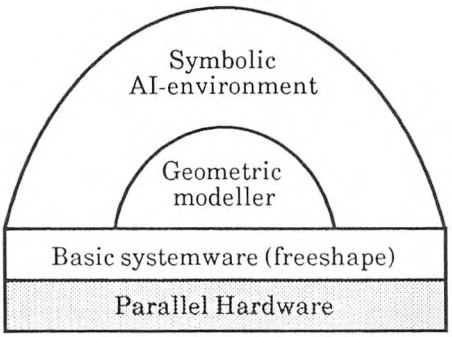


fig 1.
Overall structure of the AIM-environment

In our opinion the symbolic AI-environment is to be structured as extant AI-environments, like for example KEE, ART and Epitool.

The more programming paradigms the symbolic AI-environment subsystem supports, the better the geombolic environment will be for designing expert systems in general and geombolic expert systems in particular. The better integrated the debugging environments of the respective programming paradigms are, the better the environment will be for designing expert systems in general and geombolic expert systems in particular.

The only difference between the symbolic AI-environment subsystem and extant AI-environments is the tightly integrated geometric modeller contained. There are a number of interesting human engineering problems in this area, but as far as we can see no research issues of an implementational nature. The geometric reasoning language is at most only one more programming paradigm.

In our opinion the geometric modeller should be structured as the geometric modellers of modern CAD-systems. It must be of the solid modelling kind, it will have to support the constructive solid geometry operations of union, intersection and relative complement, it will have to support sweep operations, it must support boundary representations. The more primitive shapes the modeller subsystem make avaliable, the better the geombolic environment will be for designing geombolic expert systems.

Here we have interesting research issues of an implementational nature. The major portion of the computational resources used by an geombolic expert system will be consumed by the geometric reasoning steps. Consequently, the geometric modeller must utilise the parallel hardware efficiently.

The basic systemware should supply the implementors of the geometric modeller and the symbolic AI-environment with the right programming languages and tools, so the implementation of the higher levels can proceed speedily. We will return to this question below. For reasons of hardware economy we consider it necessary to base our software on a machine with a message passing architecture. It should have sufficient computational and graphics resources. Our current estimate is that something like an NCUBE/ten with 256 processing nodes and a graphic I/O-card [NCUBE 85], will be needed as a base for geombolic AI-environment, when used to design geombolic expert systems of practical utility for the manufacturing industries.

Today such a machine cost about 5 Mkr, by the middle of the 1990:s, when we hope to see our research results in practical use, it might cost around 100 kkr. Then it will be cost effective to place AI-environments for geombolic reasoning residing on parallel computers of said size, on most of the tables of the engineers in the manufacturing industries.

3.3 Current research: The freeshape subproject

The software requirements of the symbolic AI-environment are considerable. symbolic AI-environments are complex systems containing a plethora of detail. If we were to implement the symbolic AI-environment directly in assembler or even a systems programming language like C or Occam, our chances of meeting the project deadline would be slim. Consequently, we must introduce basic systemware defining an intermediate language higher than the systems programming languages. We call this basic systemware, the freeshape system, and the language it defines, the freeshape language.

The basic systemware is called "freeshape" since it automates the hardware related implementation tasks and thereby liberates the implementor from hardware related implementation details.

3.3.1 The freeshape Language

The freeshape language is of the "dataflow" or "forward reasoning" kind. A freeshape program can be viewed as a network of processes. The processes interact by sending values of the links which interconnect them. An example is a program for computing cos(x)*sin(x) is shown in (fig 2).

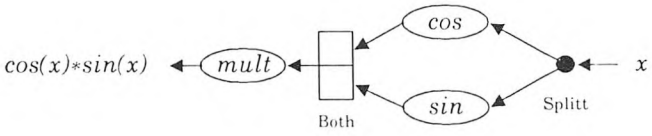


fig 2.

A splitt process copies the value recieved to it's two outputs. A both process constructs a pair (a cons) over the two values recieved. All the other primitive processes have only one input and output. Some of these expects an input in the form of a pair, or a list composed of pairs. An example is multiplication, which expect a pair as input.

The rationale for selecting the "dataflow" execution mode is simplicity. In "dataflow" execution the flows of control and of data are inseparable. Consequently, problems emanating from having the control on one processor and the corresponing data on another processor are minimized.

The freeshape system maintains a subjective time. All values created by constructs of the freeshape language are marked with the subjective timepoint at which the value are created.

The machinery for maintaining this subjective time can be implemented as an ordered sequence of process queues (fig 3).

fig 3.

The process queues are executed in subjective time order, first the t0 queue, then the t0 + 1 queue, and so on.

Within a process queue there is only a partial order over the process activations. Two activations are ordered if an only if one of the processes, directly or indirectly, send a value to the other. Otherwise the activations are

considered unordered (concurrent). One process queue correspond directly to a point in the subjective time.

The process queues and the points in the subjective time they represent are strictly ordered. This order is projected over to the process activations. Two activations residing in different process queues are always ordered.

The subjective time makes it easy to introduce a number of interesting language features. Of these, the most important is probably the resettable race-free variable.

The freeshape langage supplies the basic type "variable" and the corresponding access functions put and get (fig 4).

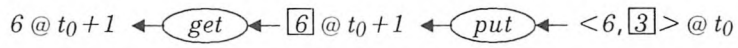


fig 4.

Put takes a pair as input. The pair is composed of a "value", *6*, and a variable set to *3*, at the time t_0. It sends the variable reset to the received value in the next point of time t_{0+1}. Get takes a variable and sends its value, *6*, in the same point of time in which it was recieved. Put takes subjective time to execute, while get takes no subjective time.

If more than one put-process tries to reset one and the same variable at one and the same point in subjective time, the variable is left unchanged and all put-processes involved will send exception values. This exception contain a list of all put-processes involved in the reset-conflict (fig 5).

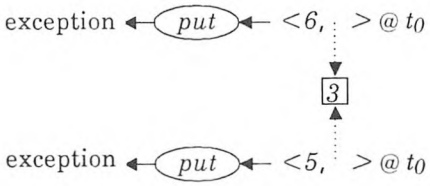


fig 5.

Two put-processes attempt to reset the same variable, at the same point in the subjective time. Both send exceptions.

Consequently, the system satisfy the following invariant: *Within one point of the subjective time a variable has one and only one value.* As the points of the subjective time are strictly ordered, the sequence of resets of the variables will be a function of the program and input at hand.

In other words, the freeshape system is race-free even though it supplies the user with resetable variables. This has been accomplished without resorting to

using explicit locks, semaphores, monitors or similar devices.

3.3.2 The freeshape system

The overall design of the freeshape system is as shown in (fig 6). The freeshape system is implemented in a number of language levels. As one progress up from the harshshape to the freeshape level, the languages takes over more and more responsibilities of the programmer.

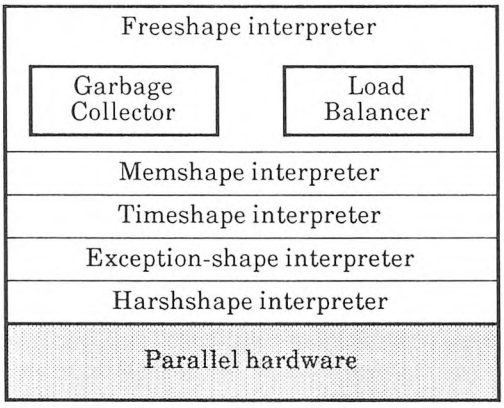


fig 6.
Overall design of the freeshape system

The most primitive of these is the harshshape language. Harshshape is nothing more than a small set of processtypes handcoded in assembler. The is a put-process but no subjective time concept. Consequently, the race-freeness of harshshape programs is the responsibility of the programmer. The harshshape system is an unfriendly programming system.

On the exception-shape level the machinery for creating exceptions are supplied. The exceptions are needed in the timeshape level where the above described subjective time machinery is introduced. At the memshape level the race-free variable and other race-free memory constructs are defined. The memshape system is an friendly programming system.

In the memshape subsystem it is possible to comparatively quickly implement the load balancer and the garbage collector which automates the hardware related implementation tasks and thereby liberates the implementor from hardware related implementation details.

A subset of all the memshape processes constitutes the primitives of the freeshape interpreter. The load balancer and the garbage collector automates the majority of hardware related implementation tasks. Consequently, we can

exclude all processes which provide explicit access to the processors, memories and communication links of the hardware, from the freeshape language. Thus, the freeshape interpreter acts as a barrier protecting the programmer from the complex structure of the parallel hardware.

3.4 Research cooperation

We are participating in the COST-13 project number 21, :Advanced Issues in Knowledge Representation:, together with universities in Brussels (Professor Luc Steels), Rome (Professor Luigia Aiello), and Pisa (Dr. Maria Simi). The RKL laboratory at our department is also participating from the Linköping side. The aim of the cooperation is to study theoretical issues relevant for massively parallel AI architectures.

We are members of the PARSYM computer mail group and regularly get the PARSYM digest via the computer mail network. Thus, we can follow what is happening in the parallel computing community in the US and indeed the world at large.

In the field of geometric modelling we have had some preliminary discussions with Professor Tom Lyche at the University of Oslo and with Peter Ta'tray at the Royal Institute of Technology in Stockholm. It is our intention to intensify this cooperation as the detailed work on the geometric modeller is undertaken. We are cooperating with Sandvik Coromant Inc. (Lennart Rohlin) in a knowledge transfer project.

We are cooperating in the use and programming of parallel hardware with the National Defence Research Institute here in Linköping and with the section for numerical analysis here at Linköping University.

The group at the National Defence Research Institute have kindly invited us to use their 16-node NCUBE system for the implementation of the freeshape system.

Notes

1. The natural language work initiated at the artificial intelligence laboratory continues in the laboratory for natural language processing, also formed in 1986. The work on non-monotonic logics initially pursued at the artificial intelligence laboratory, was transferred to the RKLlab on its formation in 1985.

2. AI-environments are also called Knowledge engineering environment, Expert system shells, et cetera.

3. The freeshape project was formerly called the hideshape project. The rationale for the freeshape system is to free the programmer from hardware related considerations. Only incidentially this implies hiding the hardware.

References

[Forbus 87] K.D. Forbus, P. Nielsen, B. Faltings, Qualitative kinematics: A Framework, IJCAI-87 pp 430-435

[Hayes-Roth 83] F. Hayes-Roth, D.A. Waterman, D.B. Lenat, Building expert Systems, Addison-Wesley Publishing Company Inc. (1983), p. 85 ,end of first paragraph.

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

[Maleki 86] J. Maleki, A Dependency Directed Deduction System Based on the Constraints Paradigm of Computation, Licentiate Thesis no. 71, Dept. of Computer and Information Science, Linköping University, 1986.

[NCUBE 85] NCUBE inc. 1815 N.W. 169th Place, Suite 2030, Beaverton, OR 97006, (503) 629-5088. Sales office: 700 E. Baseline Rd, Suite D-1, Tempe, AZ 85283, (602) 839-7545

[Sandewall 78] Erik Sandewall, Programming in an interactive environment: the "lisp" experience, Computing Surveys, Vol 10, No 1. pp 35-71, 1978

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

4.

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, in particular knowledge-based approaches, for the development and maintenance of a non-trivial range of applications software aiming at a significant increase in productivity, maintainability, understandability and user control. A central theme for our research is the integration of applied AI techniques and expert systems methodology with more traditional information technology, in particular human-computer interaction and database technology. Projects usually take an experimental approach and emphasize participation in application-oriented projects with industry and the public sector.

4.1 Summary of current research

Project activities are carried out mainly in three areas as summarized below.

Knowledge-based application systems.

This is the major activity in the lab, where we study a number of aspects of expert systems and knowledge-based techniques. In particular we take the approach that in the end a typical knowledge-based system will run in an environment where it will have to interface with traditional software, databases, conventional terminal interfaces, etc. Thus we believe that an effective strategy for productive knowledge-based application development has to start with a consideration of the ultimate delivery environment, and that we have to work backwards towards the design of facilities needed in the development environment.

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

- *Knowledge acquisition and maintenance environments.* It is generally agreed that knowledge representation and knowledge acquisition are key issues in successful expert systems development. Our approach emphasizes the importance of making knowledge re-usable, both for simplified development of generic applications and for the reuse of problem solving knowledge for e.g. teaching and training. Experimental projects have been concerned with the acquisition and representation of knowledge about technical equipment in a maintainable way. Practical applications were studied together with SATT Control (previously Alfa-Laval Automation), where we have implemented expert systems for fault diagnosis including graphical interfaces for visual interaction. Starting from these experiences current efforts are directed towards the integration of deep and shallow reasoning models, in particular "compilation" of heuristics from deep models based on qualitative reasoning. The motivation for this approach is to improve the re-usability of knowledge. Partial results have been reported in [Nordin, Weintraub 87]. A model for diagnosing multiple faults using knowledge about malfunctioning behaviour has also been developed [Hansen 88]. Current applications concern bio-chemical experiment planning (Pharmacia).

- *Knowledge-base migration and generic expert systems architectures.* Continued work has been conducted on 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. Practical cases include the migration of a knowledge base (The Antibody Analysis Advisor) from the Lisp-based EMYCIN system into the database system MUMPS. A summary of experiences in this area was among other things presented in a licentiate thesis by Sandahl 1987. A related problem was studied in a master's thesis project (Andersson), where a conventional database application (a budget quotation system) was connected to an expert system in a PC environment.

  When studying techniques for building knowledge-based system, it is also necessary to evaluate to what degree commercial tools and environments for expert systems development can be used. Our aim is to build as far as possible on existing tools and undertake tool implementation efforts only when necessary. For this purpose we are continually surveying available software on market in cooperation with the knowledge transfer program (Johansson). However, certain work on the architecture level can not be avoided and this together with our interest in efficient strategies for knowledge base migration has forced us to devote some effort to this issue (Rehmnert).

- *Intelligent human-computer interaction.* In knowledge-based systems, as in other kinds of software, the design of the human-computer interface accounts for a large part of the effort and contributes significantly to the resulting quality and usefulness of the finished system. This area has been touched upon in several of the current projects. Intelligent front-ends, i.e. systems for simplifying the interaction with complicated special-purpose

software packages, have been studied e.g. for statistical systems (Chowdhury, Sisk, Nilsson). A tool for dialogue prototyping has been developed and tested in a real application together with Philips (Löwgren). Techniques for visual interaction in trouble-shooting expert systems have been explored in cooperation with the knowledge transfer program (Hansen, Eriksson, Johansson). A main theme for our work in this area is methodology for generation of natural language text and explanations, with an emphasis on expert critiquing systems (Rankin). An preliminary experiment has been carried out on a critiquing expert system for urinary tract infections (Molin, Wiklund). Another area of great concern is knowledge-based systems for tutoring and training, with initial experiments in the economical and medical areas respectively (Hansson, Sokolnicki, Hägglund).

- *Knowledge-based approaches to systems development.* Activities in this subarea have been rather low during the year. Our long-term objective is to investigate the potential for *iterative methodologies* for systems development and how they can be supported in an effective way by knowledge-based approaches. Prototyping the human-computer interface is part this problem. Our efforts in this direction was described above and [Löwgren 88] outlines possible future investigations.

Statistical information systems.

This group, under the leadership of adj. professor Bo Sundgren, has been studying systems for processing of aggregated information concerning groups of objects in a universe of discourse. Their work in the area of intelligent aids for statistics production has involved a preliminary explorative implementation of certain aspects of a *statisticians workstation* on a Xerox Lisp machine (Chowdhury, Sisk et al.). This activity resulted in a licentiate thesis during 1987 and several publications by Chowdhury. Other investigations were performed by associated researchers in the Mathematics Department, for instance regarding the study of how administrative data could be reused for statistical purposes.

The group never reached a critical size. Since Bo Sundgren now has been granted a professorship in Stockholm, statistical information systems will no longer be a separate area of study in our group. Shamsul Chowdhury will however continue his studies in the Medical Informatics Department and contacts with that research will be maintained. We also expect that the database aspects of the area will be continued as part of our contributions to the center for industrial information technology starting summer 1988.

Knowledge transfer and other industry-related activities.

During 1987 KTP participants from ASEA, Alfa-Laval and Philips have been associated with our lab. This has involved project cooperation and course activities in expert systems and knowledge engineering. Experiences from technology transfer in the area of knowledge engineering are reviewed in [Hägglund 88].

Current involvements include cooperation with Volvo PV and with Pharmacia. We are also actively participating in planning activities for the industrial part of the Swedish Information Technology Programme, and expect that this will lead to applied projects from 1988 on. Closely related to this are the planning of joint activities with the Defense Research Institute on human-computer interaction in AI-related systems.

Another major channel for industry contacts is *Sveriges Mekanförbund*, where we participate in several study groups. We also carry out commissioned study projects, e.g. in the area of knowledge-based expert systems. These projects contribute significantly to the funding of our group, in addition to providing an effective way of communication with people from industry.

Members of the lab have also been very active in presenting our research areas and results, both on professional conferences and for industrial audiences. We also participated as one of the main organizers for the conferences on Visual Languages and on Human-Computer Interaction with 250 participants, August 1987.

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

4.2 ASLAB personnel

The following list presents persons who has participated in ASLAB project activities during 1987.

Project leadership/thesis supervision:

Sture Hägglund, PhD, acting professor
Gunilla Lingenhult, secr.

Bo Sundgren, PhD, adj. professor (until summer 1987)

Graduate students:

Shamsul Chowdhury, MSc, Tekn.Lic. (continues in medical informatics)
Henrik Eriksson, MSC
Tim Hansen, MSc
Malin Johansson, MSc, (1986/87)
Jonas Löwgren, MSc
Henrik Nordin, MSc, Tekn. Lic. (until summer 1987)
Torbjörn Näslund, BSc
Ivan Rankin, BA
Roland Rehmnert, MSc
Kristian Sandahl, MSc, Tekn. Lic. (at Epitec until summer 1987)
Tomas Sokolnicki, MSc (starting full time 1988)
Tingting Zhang, (starting late fall 1987)
Mikael Weintraub, MSc, guest researcher, (summer 1987)

Associated persons:

This list includes persons who have actively contributed to ASLAB projects during the last year, either as industry participants in the knowledge transfer program, as cooperating researchers in other departments or as undergraduate students doing their master's thesis projects in the lab. These persons are normally not paid over the lab budget.

Martin Andersson, master's thesis work
Hans Block, SCB, Stockholm
Torbjörn Eriksson, master's thesis work
Stefan Hammar, Philips Elektronikindustrier, Uppsala
Kerstin Johansson, master's thesis work
Christian Krysander, lecturer
Eva Molin, master's thesis work
Erling Nordmark, Philips Elektronikindustrier, Järfälla
Pablo Lozan-Villegas, ASEA, Västerås
Toomas Timpka, Dept of medical informatics
Ann-Charlotte Wiklund, master's thesis work

4.3 Review of major research activities.

Work in the laboratory has previously been organized in two subgroups, one for *Knowledge-based software systems*, led by Sture Hägglund and one for *Statistical information systems*, led by Bo Sundgren. During the last year activities have gradually become concentrated on the first of these areas. The main research themes are discussed below.

From 1988 on the major focus of research will be on studies of *engineering environments for generic knowledge systems.* Activities described below all contribute to these studies.

4.3.1 Knowledge acquisition and maintenance environments

Knowledge acquisition, i.e. the process of understanding, formulating and representing of the relevant knowledge for solving problems in a particular area, is generally recognized as a problem of prime importance in knowledge system development. Our studies focus on methodological support for understanding the application domain, formulating a specification of the task to be solved, developing the appropriate knowledge representation and problem solving strategies, and finally finding an efficient implementation in the delivery environment.

Our main approach is to support a two-phase development strategy, where properties of the final delivery environment is studied independently of the knowledge acquisition and development (KAD) environment. The latter should provide extensive support for formulating and understanding a given

information processing problem, including the possibility to execute the stored representation of its solution. In a second phase this solution is transformed into a production version in such a way that external constraints regarding e.g. efficiency, database size, robustness, interface to other systems, etc are observed. It is assumed that although this version of a system will allow considerably less freedom than is provided in the development environment, it will still be flexible with respect to modifiability and maintainability.

We have so far focussed our interest on knowledge-based advisory systems, in particular on what we call *initial advice consultation systems*, i.e. knowledge-based systems for supporting a non-expert user to make decisions and solve problems in a given domain. In particular it is necessary to be able to decide when a case should be handed over to a real expert. We are especially interested in diagnosis (in a wide sense), e.g. fault finding and maintenance of technical equipment. But we have also participated in and collected experiences from applications in medicine and economy.

our work on technical trouble shooting employs an approach which combines deep and shallow reasoning models. Shallow models expressing local heuristics in the form of symptom-cause rules have been successful in many cases, but suffer from the inability to provide good causal explanations or from restricted potential for reuse in similar but not identical situations. Qualitative reasoning models on the other hand have better properties from these points of view, but are harder to express and handle. We try to promote the use of qualitative models when appropriate, but with fall-back methods employing simple heuristics or user-controlled strategies as a complement. The idea is to study *families of fault diagnosis systems*, rather than starting from scratch in each new application [Nordin 87]. The use of this approach for detecting multiple faults is studied by Hansen [Hansen 1988].

In technical applications, the use of graphics and in particular the possibilities for *visual interaction* during dialogues in a consultation system for e.g. trouble shooting is of prime importance. Although we try to avoid spreading out over too many issues and thus basically intend to rely on standard graphics, we have found the design of visual interaction techniques to be so intertwined with the design of the knowledge system interface in general that it deserves further study.

Several master's thesis projects (Johansson, Eriksson, Eriksson) have been initiated in this area. Thus we have developed a graphics editor and a dialogue interface for the kind of illustrations (schemata, sketches, pictures of components, etc.) that is typically found in technical manuals. As a practical application an interface to the Alfa-Laval separator trouble shooting expert system was developed and integrated with the Epitool-based knowledge system. In an additional project this system was directly connected to a CAD system of the type used at Alfa-Laval, so that pictorial information could be directly downloaded into the trouble shooting system (Eriksson).

The visual interaction system is based on hierarchically organized composite graphic objects, where each component at a given level can be presented in different ways, *views*, e.g. as a piece of the composite object, as an icon, a (photographic) picture, or as a composite object at a lower level. The system allows the user during a dialogue to refer to a certain component by pointing as an alternative to naming, and can also highlight current components as an aid for the user to identify a specific object referenced by the system. Pictures and schemata are entered with the help of the graphics editor, the standard bitmap editor, by scanning from a paper representation, or directly from the CAD system.

4.3.2 Knowledge-base migration and generic expert systems

Historically expert systems have dealt with hard problems which could not readily be solved by conventional software development approaches. Projects were thus organized with an extreme emphasis on development support with little or none concern for conditions to be met in a regular production environment with routine users. When a successful solution was achieved, developers had to face the problem of adaption to demands regarding computational efficiency, interfacing to standard software, smoothing the user interface, technical reliability and maintainability. Sometimes these goals could be satisfactorily solved by a continued development effort. In other cases a reimplementation with a partly different technology was forced. In still other cases no fielded system was ever achieved.

Vendor of tools and environments for knowledge system development have recently recognized the importance of this problem and strive for solutions mainly along two lines. One is to reimplement the tool in a language which is easier to support in a corporate data processing center and also to interface to conventional software. It is not obvious that this approach improves the flexibility, power and usability of the development tool. The other line is to provide delivery versions of the tool, which are tuned to efficient performance in a production environment and where knowledge bases can be compiled when migrated to the delivery environment. This approach preserves the full power of the development environment, but introduces a seemingly unattractive dichotomy between the development and delivery environment.

We believe that this dichotomy should not be regarded as a drawback, but rather be envisioned as a powerful strategy which in many cases provide decisive advantages. Further we believe that time is now ripe to start from the perspective of viewing the delivery environment as the primary object and thus design the development tools starting from the anticipated spectrum of software technology indicated by application demands. With this view the reason for a separate development environment is not only to provide the best possible support in the development process, but also to avoid making premature decisions on run-time technology or particularities of a specific installation, while still designing with delivery in mind.

The first step in such a migration project has been undertaken for the Antibody Analysis Advisor (A^3) [Sandahl 1985, 87]. An EMYCIN-compatible core system was written for MUMPS (Reshagen) and a semi-automated translation system of the rule base from Lisp to MUMPS made the migration smooth [Shasavar 1985].

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. 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.

However, the routines in different blood centers differ significantly, as do their computing equipment. Thus a delivery of an A^3-like system to another blood center would presume a renewed customization of the knowledge base and the run-time environment. We believe that *customized migration of generic knowledge systems* in the long run might prove to be at least as useful as running parameterized application programs under a standard operating system.

4.3.3 Intelligent human-computer interaction

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 facilities, division of tasks between user and system respectively, etc. are primary, while syntactic details of the language used are secondary factors.

One area of particular interest to us has been to find more effective ways of producing help and explanations, in particular involving *text generation.* 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 computation 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.

Critiquing expert systems

Our experiences during the project so far has led us to concentrate our studies on *expert critiquing* as an important paradigm for knowledge-based advisory systems (Rankin 87). The critiquing approach is contrasted with the

transaction model of traditional data processing, where a predefined set of input data is required for a solution and with the expert problem-solving model, where the inferenceing of the system initiates the demand for input data. In a critiquing consultation system it is assumed that the user has to propose a solution, but that the system then produces a critical review of of this proposal.

We identify basically three subtasks in critiquing, possibly overlapping or mutually dependent, namely:

1. *knowledge generation*, where the following situations can occur:
    1. All background knowledge and all knowledge about the current situation is available before the critiquing starts;
    2. Additional knowledge is derived in order to improve the critique to be generated. For instance it may be the case that the system has to infer a 'correct' solution itself before it can produce the critique;
    3. Additional knowledge can be obtained by the user through a dialogue.
2. *focussing*, where a subset of the available or derived knowledge is selected and assigned priorities.
3. *text generation*, where the selected information is translated into an efficient presentation, presumably but not necessarily in written form. It is assumed that the presentation (or all three subtasks) is dependent upon the intended category of readers and thus that the same internal structure may result in different external texts.

We envisage at least the following cases, where all or some of the above tasks have to be solved:

1. *Expert critiquing*, e.g. commenting upon a medical diagnosis or treatment plan.
2. *Tutorial comments*, e.g. the judgements to be produced when a student has passed a computer-based training session.
3. *Problem solving explanations*, e.g. during a dialogue or after a session in an expert system.
4. *Individualized instructions*, e.g. how use or manipulate technical equipment, the functioning of which is described in a knowledge system.
5. *Data interpretation*, e.g. a weather forecast or a financial summary.

We have performed initial experiments in the first two of these areas, namely an expert critiquing system for urinary tract infections (Rankin, Molin, Wiklund) and a medical case management training system (Sokolnicki). We

intend to pursue this research further in continued projects.

Knowledge-based tutoring and training

The primary motive for development of expert systems has typically been the desire to support or automate problem-solving processes in the domain of application. However, the explicit representation of knowledge in a system can also serve the dual purpose of providing the basis also for a tutoring system, which can be used to train inexperienced personnel in decision making, especially for unfamiliar or extraordinary situations.

We believe the potential for reuse of knowledge in new applications or for different purposes, such as e.g. problem solving or training respectively, to be a core issue in expert systems technology. In order to develop a methodology and practical techniques for supporting application-oriented training in knowledge systems, we have investigated how a rule-based consultation system giving advice on economical and legal issues (LUCKY) could be reused for training (Hansson), in the same style that has previously been tried for medical decision making (MEDICS).

With these experiences as a background, we have also developed a generalized approach, which tries to incorporate support for *knowledge-based training*, in particular of *emergency procedures*, in a hybrid expert systems development environment (Hägglund 87). This approach calls for an integration of deep and shallow models for reasoning, e.g. in order to support process supervision or fault diagnosis in technical equipment based to a reasonable degree on a qualitative understanding of the corresponding processes. As mentioned before, a first study was made in the medical area. The KNOTS system (Sokolnicki) is built upon Epitool and allows the user to develop simulation problems for decision making in a structured context, in particular for medical case management.

Prototyping human-computer dialogues

An area of special importance also for the design of working knowledge-based applications is the support for a good human-computer interface. In an effort to explore these possibilities we are undertaking a study of rapid prototyping techniques and in particular the use of user interface management systems. An experimental environment for prototyping of control-panel dialogues was developed and tested both for an application at Philips and as the basis for a lab assignment in a course on interactive systems [Löwgren 87].

We expect that dialogue prototyping and knowledge-based systems can be mutually supportive, and in particular that support for local adaption of the user interface will be relevant for our work on families of customizable knowledge systems. It is also obvious that a knowledge-based system provide good support for modelling of the application functionality needed for realistic user interface prototypes [Löwgren 88].

4.3.4 Knowledge-based approaches to 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 turn out to be very important 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.

Experiences from previous projects in the area of office information systems led us to the formulation of *stepwise structuring* as a generalization of rapid prototyping, both being examples of methods for *iterative development* of software. Using a stepwise structuring approach essentially means that the degree of formalization (and thus the possibility for automated operations) of information is gradually increased during successive implementations of working prototypes or system generations. For instance, a formatted data record is conceived as a more formalized representation than a text string for a certain piece of information.

We believe that finding the right delimitation of a system's tasks and the appropriate representation of the information concerned is a crucial problem in many application areas, and that working prototypes are often effective aids in that process. Thus we think that methods for *iterative development and adaptive maintenance* are much needed and we also believe that knowledge-based techniques can contribute a lot to this end.

One useful strategy employed in knowledge systems for finding the right concepts and decision rules is *(inductive) learning.* A system for learning rules from examples was previously implemented as an experiment (Moen) and we are now looking into an approach suggested by Borgida at Rutgers regarding how to "learn" concepts and which specific attributes characterize these concepts. The idea is is to be very restrictive when the properties of vague concepts are initially specified, but periodically review all exceptions that have been encountered and when necessary relaxing the constraints or reconsidering the concept hierarchy.

This approach presumes an ability to handle exceptions in a knowledge system, which is more flexible than simply rejecting transactions that violate constraints. This is essential for systems concerned with information systems dealing with "natural" objects. Such objects are typically vague to a certain degree and it is often very hard to specify integrity constraints which prevent faulty data to enter the system, while allowing correct but unusual variations to pass. Certain (declared) types of violations of constraints applying to objects

in the knowledge base should thus not force a fatal error, but result in "exception objects" which are allowed to persist in the knowledge base.

The activities in the area of knowledge-based approaches to information systems development has been carried out at a relatively low level during 1987, mainly by associated people from the ADP group (Näslund et al.).

4.3.5 Statistical information systems.

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

The main area of study for this group has been *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.

The Statistician's Workstation.

The main effort has been 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.

The goal was 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 tabular, graphical and verbal presentation of abstracted information.

As part of the activities an experimental implementation of a *statistician's workstation* was carried out in Lisp on a Xerox Lispmachine (Sisk). This implementation primarily supports the use of a statistical program package MINITAB, which is actually run on a different computer via the local area network. The idea is to demonstrate a situation where different tools for data analysis and interpretation are available and where an intelligent front-end system helps the user to select the appropriate tool, connect to the recommended computer, initialize the processing of data and finally assists in the presentation and interpretation of the results.

In the current implementation the system basically assists in multivariate table analysis, carrying out a dialogue with the user and initiating different analyses to be done by MINITAB. Built-in statistical expertise is needed in order to structure the analysis in successive steps and guide in the choice between alternatives. The system illustrates possible relationships between variables (indicating potential dependencies) using the graphics of the Lisp Machine.

This effort is part of a research project where the possibilities of utilizing expert systems techniques in statistical information systems are studied [Chowdhury 1986]. The first comprehensive result of this investigation was published as a licentiate thesis by Shamsul Chowdhury during the spring 1987.

Current status

The group for statistical information systems has been lead on a part time basis by Bo Sundgren, who holds a position at Statistics Sweden in Stockholm. During 1987 Bo Sundgren was awarded an adjunct professorship at the School of Economics in Stockholm and left our group. Considering the circumstances we then decided not to continue research in the area, but to concentrate on the other topics studied in the lab.

Shamsul Chowdhury will continue certain aspects of his work on statistical expert systems as a graduate student in medical informatics, a group with which we have close cooperation. The *Statistician's workstation* project is however not carried any further in its current form, although we still regard research on intelligent front ends as an important topic for studies.

4.4 External cooperation.

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

1. Department of Medical Informatics. Cooperation in several areas, e.g. medical expert systems, migration techniques, expert critiquing applications and knowledge-based training systems (Timpka, Reshagen).
2. National Bureau of Statistics. Study of the design of statistical information systems. (Sundgren.)
3. ASEA. Previous cooperation on a consultation system for robot configuration has been followed by Knowledge Transfer Program activities (Lozan-Villegas).
4. Philips Elektronikindustrier AB. Cooperation in the Knowledge Transfer Program with an emphasis on knowledge-based techniques for supporting routine operators in real-time systems, e.g. in military applications, including tools for dialogue modelling (Hammar, Nordmark).
5. Volvo PV, Gothenburg. Joint activities on expert systems is starting in

December 1987 with a special emphasis on systems for fault diagnosis and technical maintenance.

6. Pharmacia, Uppsala. Cooperation on knowledge-based systems in support of experiment planning in bio-technology is starting during the spring, 1988.

During 1987/88 discussions and planning of joint projects in the area of human-computer interaction in intelligent system started with the Defense Research Institute in Linköping (Hans Marmolin) and with Department of Psychology, Stockholm University (Yvonne Waern).

4.5 Publications

For a more extensive listing of published papers, including departmental reports, see appendix E. Below a list of publications since 1986 by lab members is given for an easy reference.

External publications:

1. **Shamsul Chowdhury,** et al: Expert System Aid in Statistical Analysis and Interpretation of Data. In *Proc. of the Society of Reliability Engineers*, Outaniemi, 1986.
2. **Shamsul Chowdhury:** A Survey of Statistical Expert Systems. In *Proc. of the Conf. on Expert Systems and their Applications*, Avignon, 1987.
3. **Shamsul Chowdhury, Ove Wigertz,** Microcomputer Oriented Knowledge-Based System for Health Care Improvement in the Developing World, in *Proc. of the IEEE/EMBS 9th Ann. Conf.,* Boston, 1987.
4. **Nils Dahlbäck, Sture Hägglund,** Människa och datorsystem i samverkan. Arbetsmiljöfondens rapporter: MDA-rapport 1987:16. 1987.
5. **Sture Hägglund,** Kunskapsbaserade expertsystem, rapport 86001, Sv. Mekanförbund, 1986.
6. **Sture Hägglund,** Redskap för expertsystem, i *Proc. NordDATA -86*, och i *Nordisk DATAnytt*, no 8, 1986.
7. **Sture Hägglund,** Kunskapsbaserade expertsystem i administrativa tillämpningar, i *Nordisk DATAnytt*, no 6, 1986.
8. **Sture Hägglund,** Verktyg, arbetsformer och yrkes-orienterade språk, i *Arbejdsformer sat på dagsordenen*, SYDPOL Rapport no 1, 1987.
9. **Sture Hägglund,** Datorstödda Kunskapssystem i framtidens kontor, TELDOK rapport no 26, Televerket, 1986.
10. **Sture Hägglund, Christer Hansson and Tomas Sokolnicki:** Knowledge-Based Training of Case Management Routines and Emergency Procedures. In *Proc. of the 3rd Int. Conf. on Expert Systems,* London, 1987.
11. **Sture Hägglund,** Kunskapsbaserade expertsystem, del II, rapport 87004, Sv. Mekanförbund, 1987.
12. **Sture Hägglund,** Emerging Systems for Computer-Based Knowledge Processing in Office Work. Accepted for publication in *Office: Technology and People,* 1988.
13. **Sture Hägglund,** The Linköping Approach to Technology Transfer in Knowledge Engineering. To appear in *The Knowledge Engineering Review vol 2*, no 3, Cambridge University Press, 1988.
14. **Tim Hansen,** Diagnosing Multiple Fault Using Knowledge about Malfunctioning Behaviour, to appear in *Proc. of the 1st Int. Conf. on Industrial and Engineering Applications of AI and Expert Systems*, Tullahoma, 1988.
15. **Minton, Carbonell, Knoblock, Kuokka and Nordin,** Improving the

Effectiveness of Explanation-based Learning, in *Proc. of the Workshop on Knowledge Compilation*, Sept. 24-26, Oregon State University, 1986.

16. **Henrik Nordin:** Using Typical Cases for Knowledge-Based Consultation and Teaching. In *Proc. of the 3rd Annual Conf. on Applications of Expert Systems*, Orlando, Fla., 1986.
17. **Ivan Rankin,** On the Implementation of Hellberg's Morphology System. *Proc. of the Fifth Meeting of Nordic Computational Linguists,* Helsinki 1986.
18. **Roland Rehmnert:** Programmering som intellektuell process. Erfarenheter av vidareutbildning av industriverksamma. In *Proc NordDATA 87,* Trondheim, 1987.
19. **Kevin Ryan,** et al., Surveying Software Tools for a Method Driven Environment, *Proc IFIP-86*, Dublin, 1986.
20. **Kevin Ryan,** The Value of Mixed Metaphors in Computer Education, *Proc. Nat. Computer Education Conf.*, San Diego, 1986.
21. **Kristian Sandahl:** The Migration of Expert Systems into Production Environments. *Proc. Nord-Info Seminar on Knowledge Engineering*, Köpenhamn, 1986.
22. **Pål Sørgaard,** Evaluating Expert Systems Prototypes. Presented at *The 9th Scand. Sem. on Use and Development of Information Systems,* Båstad 1986.

Licentiate theses

1. **Shamsul I. Chowdhury,** Statistical Expert Systems - a special application area for knowledge-based computer methodology, 1987.
2. **Kristian Sandahl,** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.

Additional reports

1. **Martin Andersson:** A Prototype of an Automatic Budget Quotation Generator, Master's thesis, Linköping University, 1987.
2. **Torbjörn Eriksson, Kerstin Johansson:** Ett grafiskt gränssnitt för konsultationssystem inom reparation och underhåll, Master's thesis, Linköping University, 1987.
3. **Eva Molin, Ann-Charlotte Wiklund** Ett kommenterande expertsystem för konsultation vid behandling av urinvägssjukdomar, Master's thesis, Linköping University, 1987.
4. **Roland Nilsson:** A Survey of Planning Techniques for Intelligent Front Ends. Master's thesis, Linköping University, 1987.
5. **Jonas Löwgren,** Applying a Rapid Prototyping System to Control Panel Dialogues. Report LiTH-IDA-R-87-26.
6. **Henrik Nordin:** Reuse and Maintenance Techniques in Knowledge-Based Systems. Report LiTH-IDA-R-87-18.
7. **Henrik Nordin, Mike Wientraub:** Developing Families of Fault Diagnosis Systems, report in preparation, 1987.
8. **Kristian Sandahl:** Creating an Antibody Analysis Advisor as an Exploratory investigation into Expert System Development. LiTH-IDA-R-85-20
9. **Henrik Eriksson:** Using CAD Information in Expert Systems, IDA Systems Doc. 32.1, Linköping University, 1987.

ASLAB Memo series 1987-

87-01 **Rankin, I,** An Orientation Study in How Language is Used.
87-02 **Löwgren, J,** Användargränssnitt - specifikation och design. Master's thesis.
87-03 **Eriksson, H,** Överföring av CAD-information till expertsystem. Master's thesis.
87-04 **Sokolnicki, T,** Knowledge-Based Support and Training of Decision Making Situations. Master's thesis.
87-05 **Rankin, I., Hägglund, S.. Molin, E., Wiklund, A.C.** Implementing CRIME - a Critiquing Commentary System.
87-06 **Sandahl, S.:** The Migration of Expert Systems into Production Environments.

87-07 **Hägglund, S.**, Emerging Systems for Computer-Based Knowledge Processing in Office Work.

87-08 **Nordin, H., Weintraub, M.**, Developing Families of Fault Diagnosis Systems.

88-01 **Löwgren, J.**, Outline of a Portable Display Manager for Direct-Manipulation User Interfaces to Knowledge-Based Systems.

88-02 **Hägglund, S., Rankin, I.**, Investigating the Usability of Expert Critiquing in Knowledge-Based Consultation Systems

88-03 **Rankin, I.**, Towards Effective Text Generation in Critiquing Expert Systems

5.

CADLAB
The Laboratory for Computer-Aided Design of Digital Systems

Krzysztof Kuchcinski

5.1 Introduction

The laboratory for Computer Aided Design of Digital Systems, CADLAB, is concerned with the computer aided synthesis and verification of digital systems, especially those involving very large scale integrated circuits (VLSI). The major effort of our research work concentrates on the behavioral and structural aspects of digital system specification, design, simulation, optimization, partitioning, synthesis and formal verification methods.

CADLAB was formed from Professor Harold Lawson's Telesystem group when Telesystem, Datalogi and ADB merged to form IDA in 1983. The first years of CADLAB were devoted to building competence in the area and establish the research profile of the group. They formed the basis for our present activity.

During 1987 CADLAB is able to report on some significant progress. This progress will be highlighted in detail in later sections. In the reported time CADLAB was broadly concerned with several aspects of the hardware design problem, especially the process of translating a behavioral description of a digital system into VLSI implementations. Such a transformation should be carried out so as to preserve semantics of the algorithm and at the same time meeting certain cost/performance constraints. Therefore it is a quite complex problem and in practice exhaustive searching for optimal implementation is impossible. To address the complexity problem, several methods are proposed. First, the design space can be cut if we assume a given class of system

---

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

architectures. Second, the intermediate representation of the design can be introduced to form a base for different optimization strategies. And finally, the stepwise refinement method can be utilized.

Our research activities concentrate mainly on different methodologies and computer-aided design tools which constitute a complete hardware design environment. The main field of interest include synthesis and verification problems. As a result of this work, we have also implemented, among others, the ASL language and the CAMAD system.

5.2 Current Work

CADLAB is currently engaged in a set of research projects, collectively called the ASAP-project (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. This assumption has an evident implication on the CAD tools. First, the tools should allow the designer to explore the design space starting from the architectural level rather than a lower level like the logic level. This provides a possibility to choose a better design at an early stage of the design process. Second, it supports the design of special purpose systems which are embedded in a wide variety of products (sometimes called ASIC, i.e., Application Specific Integrated Circuits) such as telecommunication systems, electronic and biological instruments, robots and automatic control systems. For 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 potentially quite high. In the next section we now identify the major premises of the project.

5.3 Asynchronous Architectures

In the project we argue strongly for the asynchronous architecture implementation of embedded (special purpose) systems. The facts that support our assumption are as follows:

- Today, VLSI technology ensures a high performance of digital circuits. 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 0.5 mm as it does to go along a coaxial cable 75 cm long."
- There are problems to implement a single, global clock for large synchronous systems (for example, two-dimensional systolic arrays). 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. To synchronize independent isochronous regions we can

use a self-timed discipline which leads us to asynchronous systems.

- Embedded systems are inherently parallel. Typical tasks to be performed in embedded systems are data capture, processing, control signal generation, display maintenance and possibly statistics gathering. They are usually described as a set of heterogeneous tasks which are called *processes*. The process can be view as an independent program which communicates with other processes and the external environment to perform some system activities. To implement the process abstraction, we map processes into the set of *processors*. This also leads to the asynchronous architecture with the communication protocol between processors.
- The asynchronous architecture approach forms a background for a "flexible" design style. This implies that a system may be easily extended using well defined components. One can think also about specialization of the system components (e.g. processors) to obtain an efficient realization, well suited for the problem. Trade-offs to obtain a balance between performance and cost-effective implementation are also possible.
- This approach, where timing is handled by the asynchronous strategy, provides a basis to design *systems* rather than single chips. In order to find "optimal" system solution, we need also CAD tools capable of making hardware-software trade-offs. These CAD tools must provide various forms of synthesis algorithms and support the evaluation of the designs in order to compare the performance. This helps a designer to explore the design space so as to reach an optimized solution.

We have now identified the major advantages of using asynchronous architecture approach at different levels starting at the technology level with assumptions about delays and clocks and ending at the system level with assumption of design style. To fully utilize the proposed architecture benefits there must exist, however, a CAD system to support this approach. The CAD system must conveniently provide many different tools that utilize an efficient graphical man-machine interface as well as a design database which possess a knowledge about various design views. Typically it should contain tools like synthesizers, simulators, verifiers as well as other specialized programs for test pattern generator, rule checkers and enforcer, etc.

5.4 Ongoing CADLAB Projects

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

System Specification and Verification

To support architectural specification, synthesis and analysis, a specification language has been proposed in a licentiate thesis by Tony Larsson. The language supports synthesis, analysis and simulation tools. A set of calculus and algebraic manipulation rules related to ASL descriptions are investigated.

These include rules for abstraction, binding, and event reduction and they are intended to form a framework for the design of higher level verification and synthesis tools. Enabling semantic preserving syntactic transformations, the rules support algebraic verification methods; however, exhaustive verification methods (based on simulation) is also supported and can be made more tractable (by improved simulation speed) if abstraction and binding rules are used to prune a design description.

This work continues with extended studies of the semantic base for the algebraic manipulations, e.g including a boolean ring structure that provides the framework for propositional and predicate logic reasoning. Other directions studied includes arithmetic and temporal reasoning.

Schematic Entry System

An attempt to study and evaluate a schematic entry system was made as examination work by an undergraduate student, Mats Larsson. The reason for this was twofold, first to analyze the extent to which such a system could be integrated with tools developed at CADLAB and second, to develop a tool consisting of the schematic entry system and a simulator, Adapt, developed by Tony Larsson to be used in teaching undergraduate students.

Synthesis from Behavioral Descriptions

The hardware synthesis project aims at the development of a formal design methodology and an integrated set of automatic as well as computer aided design tools for the synthesis problem. We are particularly interested in the synthesis of VLSI systems from their high level behavioral descriptions. Four major tasks of the synthesis process have been treated in this research: first the automatic transformation of a high level behavioral description which specifies only what the system should be able to do into a structural description which specifies the physical components and their connections; second the partitioning of a structural description into a set of modules so that each module can be implemented independently and operated asynchronously; third the optimization of the system implementation in terms of cost and performance; finally, the automatic generation of microprograms to implement the control structures of VLSI circuits.

To address these four synthesis problems, a formal design representation model, the extended timed Petri net (ETPN), has been developed by Zebo Peng. This design representation consists of separate but related models of control and data path. It can be used to capture both the structures and behaviors of VLSI systems as well as the intermediate results of the synthesis process. As such, the synthesis tasks can be carried out by a sequence of small step transformations. The selection of these transformations is guided by an optimization algorithm which makes design decisions concerning operation scheduling, data path allocation, and control allocation simultaneously. This

integrated approach results in a better chance to reach the globally optimal solution.

Besides the four basic synthesis problems, several related issues of VLSI system design have also been studied in this project. The architectural considerations for the implementation structure is concerned with the selection of the target architectures for the VLSI implementations. Automatic extraction of parallelism from a sequential program deals with the problem of how to extract parallelism from PASCAL programs. Finally, the problem of integrating a design verification scheme into the synthesis process is also investigated. These issues have been studied by Zebo Peng and Krzysztof Kuchcinski.

Synthesis of Pipeline Structures

Pipelining is a fundamental technique in computer design, but since it is today usually a manual process, it is prone to design errors. This project, conducted by Björn Fjellborg, aims at finding a way to describe systems so that extraction and synthesis of "pipelinable" parts can be done automatically. "Extraction" is taken to be different from synthesis in that it only aims at determining whether pipelining is at all feasible. If so, a complete synthesis procedure follows. However, as the extraction requires an evaluation of potential pipeline structures in the design, it is in fact a partial synthesis. An important question is exactly what parts of the synthesis must be performed to obtain enough results for an evaluation. The criteria that has to be tested include not only static information (can the system be described as a sequence of elementary operations), but also the dynamics of the system, i.e., to what extent a pipeline will be used.

A pipeline can be described in terms of processes and processors, yielding two fundamental views:

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

The first view facilitates description of communication between stages. Both capture the state of the pipeline; the second view in particular connects each piece of state to the associated computation. The second view can also be seen as higher-level description of the system that is independent of particular implementations. Those are then described by the first view. Formalisms for process definition in this context need be developed, as processes may interact and spawn into subprocesses.

This project tries to combine these features into a single view, by using a formalism that describes each excitation of the pipeline as a separate process, and letting these processes be interrelated by the constraints given by the data path (pipeline stages). This way, one can achieve a description that allows expression of scheduling constraints as well as error handling, and thus can be used as the basis for a CAD tool for designing pipelined systems.

In the ASAP context, synthesizing pipelines can be interpreted as the dynamic run-time interconnection (by virtual circuits) of a given set of function modules into temporary pipeline configurations. Parallelisms can be drawn between this "dynamic pipelining" and the more conventional "static pipelining", but now the cost for run-time extraction/synthesis must be included.

Embedded Architectures

During the present research year, Mikael Patel has been investigating embedding higher level processing elements into the proposed asynchronous architecture, ASAP. Previous examples, simulations, and models of computation within ASAP have mainly utilized fine-grain parallelism such as multiple parallel executing arithmetical units.

To allow the interconnected processing elements to be programmable and flexible we are interested in processor structures which allow a high level of flexibility and performance to a low level of microprogramming. This has initiated a study of architecture concepts developed within the Reduced Instruction Set Computer (RISC) research community.

The choice of a reduced instruction set and application specific function support contributes to a higher level of flexibility. The asynchronous port mechanism as proposed within the ASAP project combined with high-level computation, i.e. processes, reduces the high overhead associated with asynchronous communication and achieve yet a possible level of parallelism into the overall architecture. Processing and communication may thus better be balanced.

The choice of programmable processing elements within the ASAP architecture is also motivated by 1) the reduction of data paths and the possibility of communication, i.e., parallel or serial, between functional units, and 2) the overall architecture may be tailored of an application area instead of a specific application and allow evolution of the embedded computer system.

Design Database

A group of students under the supervision of CADLAB have defined and implemented an object oriented database to allow a common core for data representation and manipulation within a computer aided design environment.

Data within the experimental design database is modeled as persistent associations between objects and values. The overall idea behind the database is to represent data as simple object-attribute-value-triples. All manipulation of data is performed on these triples through a library of access functions. The database closely corresponds to a fine-grain segmented virtual memory.

During the summer of 1987 Göran Rydquist evaluated and redesigned the

design database as part of his Master Thesis. The experimental database access library is currently implemented in C and runs on a SUN-3 under UNIX.

Now that we have considered the general course of research activities, let us highlight the progress made during 1987.

5.5 Progress During 1987

The CADLAB group has made several important advances during 1987. This has resulted in the completion of one Ph.D. dissertation and the publication of several papers in international journals and international conference proceedings.

We feel that we have made progress in all of the ongoing projects mentioned in the previous section. In particular, based upon the published papers and the Ph.D. dissertation, we can identify the following specific progress.

In the area of automated hardware synthesis, the definition and implementation of a CAD system, called CAMAD (Computer Aided Modelling, Analysis, and Design of VLSI Systems), has been presented in Zebo Peng's Ph.D. dissertation. The dissertation proposes a formal model for VLSI design representation and a set of optimization, partitioning and synthesis algorithms. Additionally this work has been described in many papers published in different journals and international conferences proceedings. The highlights of this work are presented in section 5.6.

The work on specification and verification of digital systems showed also a significant progress during 1987. Based on the language for a specification of asynchronous architectures the symbolic manipulation set of hardware oriented rules forms a basis for the research in the field of hardware verification and synthesis. The highlights of this work are presented in section 5.7.

5.6 Automated Synthesis of VLSI Systems

Zebo Peng's dissertation describes the development of a VLSI automated synthesis system. The input to the synthesis system is high level behavioral descriptions of VLSI systems and the output is their implementation structures at the register transfer level.

The major contribution of this dissertation is the development of a formal design methodology and a set of design algorithms for the automated synthesis of VLSI systems which includes the following components:

- A formal design representation model, the extended timed Petri net (ETPN), which can be used to represent a design with various degrees of abstraction during the synthesis process.

- The formal notation of semantics and semantic equivalence of the ETPN design representation, which leads to the formal verification that the implementation produced by the synthesis algorithm satisfies the behavioral specification of the designed system.
- A set of transformation rules for the ETPN model and an optimization strategy which selects the transformation rules and applies them to a particular design. The transformation of behavioral description into structural description is formulated as a optimization problem and a horizontal optimization algorithm is developed.
- A procedure for translating a PASCAL program into the ETPN design representation. Extraction of parallelism from the algorithms in terms of PASCAL programs is performed during this procedure.
- A partitioning algorithm which transforms an ETPN design into several modules such that each can be later implemented independently and operated asynchronously.
- A microprogram generation algorithm which generates microprograms to implement VLSI control structures represented as timed Petri nets. The algorithm also analyzes some properties of the given Petri net and performs microprogram reduction and optimization.
- An algorithm for translating the ETPN design representation into a register transfer level implementation with some physical parameters which can later be used to drive a layout generation tool.

One of the basic issues of a synthesis system is the design representation model used to represent the design during the synthesis process. The ETPN model we have developed is a *unified* design representation which can be used to represent the design with several levels of detail. That is, it allows the designers to specify the abstract information of the systems being designed so as to give the designer freedom in the implementation phase to make trade-offs, or to make use of some automatic tools, to reach an optimized solution. At the same time, it also provides the capability for the designers to predefine lower level details when the designer feels it is necessary to freeze some implementation decision in order to cut down the design search space or to make use of some standard components stored in a module library.

The ETPN design representation consists of separate but related models of control and data path. The data path of the design representation is represented as a directed graph with nodes and arcs. The nodes are used to capture data manipulation units, which prescribe behaviors of the data manipulation units as mappings from a set of inputs to a set of outputs without giving the details of how the system performs the mappings. The arcs represent the connections of the nodes. The control part of the design representation, on the other hand, is captured as a timed Petri net with restricted transition firing rules. These two parts are related by the control signals coming from the control part to the data path and the conditional signal traveling in the opposite direction. One small example of the ETPN

design representation is illustrated in Figure 5.1.

As Petri nets are inherently concurrent, the ETPN design representation can deal with the design of distributed control for multiple processes executing concurrently. In a Petri net model, there can exist more than one event at a time as indicated by several S-elements holding tokens simultaneously. More importantly, the progression of these events may occur independently; there is no need to synchronize these events unless it is required. When synchronization

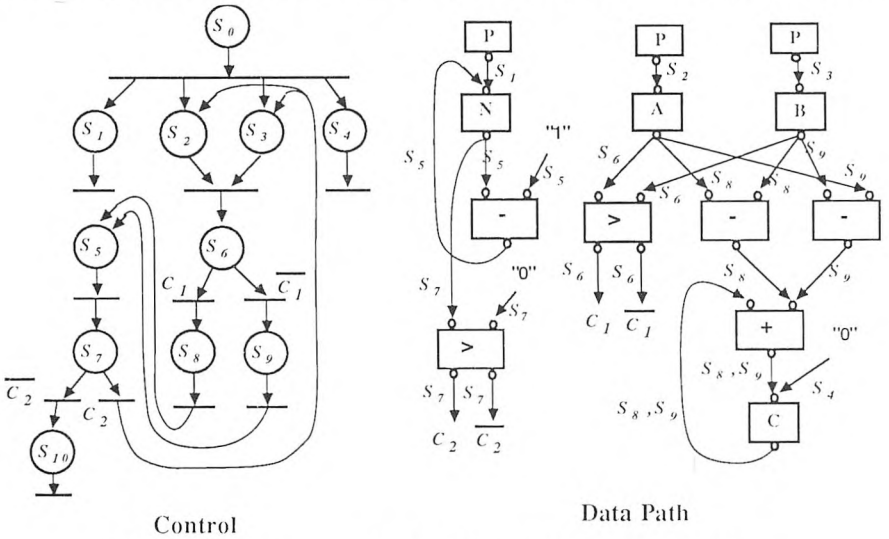


Figure 5.1 An example of the ETPN design representation

is needed, it is also very easy to model it. Another advantage of a Petri net is its asynchronous property. There is no inherent implication of a clock mechanism for the firing operations. Only the partial ordering of the occurrence of events is specified.

The main feature of the ETPN design representation is its ability to capture the intermediate result of a design explicitly so as to allow the design algorithm to make accurate design decisions. For example, given an intermediate result represented in the ETPN form, an algorithm can be used to calculate its implementation cost, check whether that satisfies the design constraints, and automatically choose a transformation to apply to the design which produces another intermediate result with improved cost.

The formulation of the unified design representation model also provides a framework to incorporate a set of different design methodologies and tools in a coherent way. In our approach, different design tools all interact with the ETPN design representation which functions as a centralized design data base. Therefore they can freely communicate with each other and the effect of one design algorithm can immediately be visible to the others.

With the formulation of such a formal design representation, it is also possible

to perform verification of digital hardware at different levels, i.e. to prove that two designs which may be at different levels are equivalent to each other in respects to some axioms.

We have also formalized the process of transforming a behavioral description into structure description as an optimization problem. An optimization strategy is then developed to guide the transformation process so as to reach an optimal design. The basic idea is that the ETPN description model allows us to view a behavioral description as a *primitive* structural description which is of course very crude, i.e., if we implement it directly, we get a very expensive design. But once we have a structural description, we can make improvements on it to produce a better one; moreover these improvements can be done step by step until a satisfactory result is reached.

The synthesis process starts with a PASCAL description of the designed system. The first step of the synthesis process is to translate the PASCAL program into its equivalent ETPN description which then serves as the intermediate representation of the design. The second step is to perform operation scheduling, data path allocation, and control allocation integratively. The third step is to partition the design into a set of asynchronous modules. The final step is to translate the ETPN design representation into the register transfer level design with control implementation.

The formalization of the ETPN design representation model and the synthesis process have led to the efficient use of CAD and automatic tools in the synthesis process. An integrated design environment, the CAMAD design aid system, has been developed based on the ETPN model and the proposed synthesis methodology. The overview of the CAMAD design aid system is illustrated in Figure 5.2. CAMAD is written in PASCAL and runs on a VAX 11/780 machine under the VMS operating system.

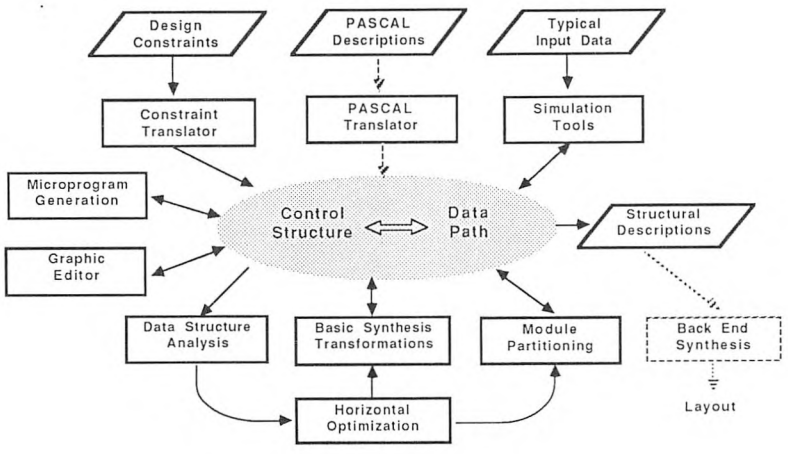


Fig. 5.2 Overview of the CAMAD design aid system

5.7 Specification and Verification of VLSI Systems

In this project a method for specification and verification of hardware systems, based on a language called ASL (Action Specification Language), is studied by Tony Larsson. The work concentrates on conceptual and semantic issues related to the specification and verification of hardware systems behavior and corresponding structural implementation. Behavior includes both actional behavior aspects, time independent functional aspects, and data type aspects. Actional behavior includes interface issues, synchronous and asynchronous timing strategies, as well as other timing properties. ASL supports specification of hardware systems including their actional behavior, functional transformations, and structural decomposition. It is an open language in the sense that it can be adapted to a library of functions, actions, and module primitives. This will support reusability of functions, interfaces (communication actions related to ports) as well as complete modules. A simple LISP like syntax is used to provide a symbolism for the semantic definition and conceptual ideas. A set of (action) calculus and reduction rules are intended to be used as a framework for a set of verification tools.

An important distinction is made between actional behavior and time independent functional transformations. The aim is to support reusability of functions as well as interfaces. An interface is defined as a composition of (temporal) actions involving a set of ports. In contrast to actions, the evaluation of functions is assumed to be performed without direct temporal effect. Functions, primitive or composed, may in this sense be thought of as a tool for temporal abstraction. Function abstractions can either be primitives inherited from a library or be defined in a purely functional style. Later on, in the implementation phase, a function abstraction may be partitioned and described as a structure of data driven modules.

The semantics of ASL is defined by help of a semantic relation. A calculus (semantic preserving syntactic transformation rules), event and port reduction rules, and partial evaluation (binding) rules are proposed. These rules are intended to support semantic preserving syntactic transformations which enable verification of equality of a design specification and its implementation. Port binding and partial evaluation may also be used to prune a design specification and/or implementation (in order to reduce combinatorial and sequential complexity) so that both deductive (symbolic) and exhaustive verification techniques are made tractable.

Currently the symbolic manipulation of logic and arithmetic expressions embedded in ASL descriptions are studied. The symbolic manipulation is to a large extent based on well known algebraic systems such as the boolean ring structure for logic expressions and integer rings and fields for arithmetic expressions. Further the coupling of these systems with conditional and temporal expressions are studied. The research issue includes how the knowledge base represented by these algebraic systems can be combined and used in hardware verification applications. These applications includes a wide spectrum of circuits at transistor-, gate-, register-, and architecture- levels.

5.8 Related Activities

CADLAB is involved in the graduate course program of IDA. During the autumn of 1987 the course "Introduction to Petri Nets" was given by K. Kuchcinski together with J. Maluszynski (LOGPRO). Additionally, a series of research seminaries on digital systems design automation are given every week by members of CADLAB.

During 1987 the work of the group was presented on the following international conferences:

The 8th International Conference on Computer Hardware Description Languages and their Applications, Amsterdam, April, 1987.

The 13th EUROMICRO Symposium on Microprocessing and Microprogramming, Portsmouth, September 1987.

We have contacts with other groups working in the area of digital systems design automation. During his visit in CADLAB in November 1987, Dr. Hristo N. Djidjev form Bulgarian Academy of Science expressed his interest in further contacts with our group in the subject of design automation of systolic arrays. We are also planning to keep closer contacts with Dr. Peter Marwedel's group at Kiel University, West Germany. Dr. Marwedel visited CADLAB in December 1987 as the opponent of Zebo's Ph.D. dissertation.

5.9 Personnel

Krzysztof Kuchcinski, Ph.D.
Prof. Harold W. Lawson Jr., Ph.D. (on leave from CADLAB)
Britt-Marie Ahlenback, secr.
Mikhail Ferapontous, Ph.D. (guest researcher from Sept. 1987)
Björn Fjellborg, MSE
Tony Larsson, Tech.Lic
Fredrik Lindström, MSE (from July 1987)
Mikael Patel, Tech.Lic.
Zebo Peng, Ph.D.

Professor Harold W. Lawson Jr. is on leave from the university due to his work on organizing the Swedish International University. During this time Dr. Krzysztof Kuchcinski acts as a laboratory leader.

5.10 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.

Tony Larsson, On the Specification and Verification of VLSI Systems.

5.11 Ph.D. Theses

Zebo Peng, A Formal Methodology for Automated Synthesis of VLSI Systems.

Piotr Siemienski, Specialized Database for Computer-Aided Design of VLSI Integrated Circuits. (In Polish.)

The major part of Siemienski's thesis work was done at CADLAB during 1983-85. However, the thesis was defended in the Institute of Electron Technology CEMI, Warsaw, Poland.

5.12 References

The following are the CADLAB publications for the year 1987 that are referenced in the text. For the full list of publications, please refer to the appendix.

1. K. Kuchcinski and Z. Peng, *Microprogramming Implementation of Timed Petri Nets*, INTEGRATION, the VLSI journal 5 (1987), pp. 133-144.

2. K.Kuchcinski and Z. Peng, *Parallelism Extraction from Sequential Programs for VLSI Applications*, Report LiTH-IDA-R-87-20, also to appear in Microprocessing and Microprogramming, the Euromicro Journal

3. T. Larsson, *Specification and Verification of VLSI Systems Actional Behavior*, The 8th International Conference on Computer Hardware Description Languages and their Applications, Amsterdam, April, 1987.

4. T. Larsson, *Semantics of a Hardware Specification Language and Related Transformation Rules*, INTEGRATION, the VLSI journal 5 (1987), pp. 145-158.

5. Z. Peng, *A Horizontal Optimization Algorithm for Data Path/Control Synthesis*, to appear in Proc. of the IEEE International Symposium on Circuits and Systems, Espoo, Finland

6.

LIBLAB

The Library and Information Science Research Laboratory

Roland Hjerppe

6.1 Introduction

LIBLAB, a joint project of the Department of Computer and Information Science and the University Library has been funded by the Delegation for Scientific and Technical Information (DFI) since it started in 1983. In the first research program for LIBLAB of 1982 two major research themes are specified: *Document Description and Representation,* and *Users and (Library) Systems.* For each of them there are also two subthemes defined. One minor theme, *Networking*, is also specified.

1983-1985 research efforts were concentrated mainly on the first theme: Document description and representation. The "Anglo-American Cataloging Rules. Second Edition." (AACR2) was studied by building a a number of small knowledge based systems with the cataloging rules as a knowledge domain. During 1985 the HYPERCATalog-project emerged as a project that relates to all the themes. This project has been the main focus of LIBLAB since then.

Research activities in the HYPERCATalog project have focused on HYPERKITtens, i e implementations of hypertext software for use in bibliographic systems.

In early 1987 it was decided in Parliament that DFI should cease existence as of July 1, 1988. As a consequence of this decision the future direction of LIBLAB has been reconsidered and discussed throughout the year. Much effort

---

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

has been used in planning sessions, attempts to find new sources of funding, and a critical assessment of our own work.

At the instigation of DFI an assessment of LIBLAB and its work has been carried out by Dr Philip Bryant at the Centre for Catalogue Research, University of Bath, England, one of the leading experts in the field of bibliographic research. This assessment was thoroughly reviewed by the members of LIBLAB. One result of this was a new research program for LIBLAB.

6.1.1 Assessment of LIBLAB

LIBLAB spent two days in an internal workshop in October discussing the present situation and the future. The LIBLAB research program and the assessment by Philip Bryant were reviewed by the lab members. The workshop resulted in a plan to rewrite the research program in order to reflect the present research orientation as well as taking heed of comments and advice from Philip Bryant's report.

In his assessment Bryant concludes that:

- the potential value of LIBLAB's work is considerable;
- IDA provides a base for library research that is unique in Europe if not in the world;
- too wide a variety of research interests has hampered progress;
- LIBLAB enjoys a much higher profile overseas than in Sweden - dissemination of results and ideas to the Swedish library community should be improved.

6.2 A new research program for LIBLAB

One result of the internal and external assessments of LIBLAB was that the necessity of a new research program became clear. All of the laboratory participated in formulating the new program during the last quarter of the year.

The new program outlines two major themes:

1. Users and information systems, especially bibliographic systems
2. Document description and representation

These themes are mainly the same as before. Differences appear in the subthemes:

1.1 **User participation and user behavior.**
Active participation by the users in the conceptual modelling of the system is encouraged. The conceptual model will be the basis for the user interface. Individual user profiles are necessary for the adaption of the system to the users' needs.

1.2 **Orientation in the database: Maps and other tools.**
The user should be able to browse and navigate in the database. In order to do this guides, maps and other tools for orientation are required.

2.1 **Description and representation of collections and elements.**
Which modes of representation are adequate for the structures of knowledge organization in bibliographic databases?

2.2 **Structures of collections, literatures and domains.**
There are three different types of structures in document collections. Generating the structures and their relationships is the focus of this subtheme.

2.3 **Hypertext and hypermedia.**
Hypertext and hypermedia are of general interest for LIBLAB.

The HYPERCAT project is still the main project that brings together experiences and knowledge gained in the different projects pursued within the research themes.

6.3 Project HYPERCATalog

The HYPERCATalog project was described in the previous Annual Research Report and fuller descriptions are available in LIBLAB's report series. The goal is to build and implement a library catalog that differs in most ways from today's catalogs. The most important desirable features are:

- The catalog as a hypertext structure, implying navigation and browsing as the primary modes of use.
- 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.

In 1987 the HYPERCATalog project has focused on model construction, where the models have been given the name HYPERKITtens.

6.3.1 HYPERKITtens

HYPERKITtens are small prototypes - incomplete hypercatalogs - based on existing hypertext software. By creating HYPERKITtens it is possible to test preliminary ideas of the HYPERCATalog and get suggestions for modifications and improvements. Presently LIBLAB has access to hypertext software for Macintosh computers such as Guide, Storyspace, and HyperCard, in addition to Notecards, which is designed for Xerox workstations.

The work on HYPERKITtens is carried out with the same database implemented on the various systems in order to make possible comparisons between them. Furthermore, different applications are used in each system in order to take advantage of the specific features of the software.

HYPERLIB is a project assignment carried out by Kerstin Andreasson, a systems analysis student. The project has produced two HYPERKITtens, one based on Guide and the other on HyperCard, demonstrating the use of geographical maps and floor plans for orientation and navigation in the library and its collections.

HYPERCLASS is a project that demonstrates the use of hypertext softwares for presentation of classification structures. In this system the classification scheme (SAB) serves as the main access point to the library, whereas HYPERLIB offers access by physical location. Three HYPERKITtens, one based on Guide, one on HyperCard and one on Storyspace are being developed.

HYPERREF is a system where some 200 bibliographic references have been linked using the Guide software. References and links make up trails. A sequence of links forms a path, and with a set of paths and trails it is possible to navigate in the collection or parts of it. The same database is being used in

another HYPERKITten based on Notecards.

The HYPERKITtens described here could be amalgamated into a fuller HYPERCATalog prototype demonstrating many of the desired features listed above.

6.4 Other projects and activities

Preparatory work on combinations of hypertext and formal bibliographic description, using ideas and formalism from SGML, Standard Generalized Markup Languages and other structure editors has also been done.

Two doctoral projects are being carried out within the research theme *Users and (information) systems*:

The project "Cognitive systems models and user interface construction" focuses on conceptual design, user interface construction, and user participation in system design.

The project "Personal information management in computer science" is a series of case studies of computer scientists with the purpose of identifying characteristics of personal bibliographic information management in a research environment.

Manny Jägerfeld has been engaged by the WHO and the Center for Medical Technology Assessment in Linköping to build a database for a European clearinghouse on assessment of health technology.

Birgitta Olander has been engaged by UHÄ in the work on a national strategy for information services for higher education and research.

In December LIBLAB hosted a seminar called "Swedish Research in Library Science after DFI", with invited speakers from Inforsk (Umeå Univ.), the Center for Library Research (Gothenburg Univ.), and DFI. Over 50 people from all over the country participated in the seminar.

LIBLAB was also represented at IFLA (Int. Fed. of Library Associations) 53rd General Conference, August 16-21, in Brighton, England, where Roland Hjerppe presented two short invited papers.

Roland Hjerppe also attended HYPERTEXT'87, November 13-15, in Chapel Hill, North Carolina. This was the first conference dedicated to hypertext, and it attracted some 200 participants.

Arja Vainio-Larsson attended ACM CHI+GI 1987 (Human Factors in Computing Systems and Graphics Interface), April 5-9, in Toronto, Canada;

and Interact'87 (2nd IFIP Conf.), September 1-4, in Stuttgart.

LIBLAB has also participated in a project at the Dept. of Medical Informatics aimed at building decision support systems for general practitioners in primary care.

6.5 Laboratory members

Roland Hjerppe, MSc.
Laboratory leader: planning, coordination and administration.
Research areas: the HYPERCATalog project; Hypermedia; SGML and other structure editors.

Birgitta Olander, BA, MLS.
Research assistant. Doctoral student at the Faculty of Library and Information Science, Univ. of Toronto, Canada, since 1984.
Research area: Personal information management; the HYPERCAT project.

Arja Vainio-Larsson, MA.
Doctoral student since 1984.
Research area: Cognitive systems models and user interface construction; the HYPERCAT project.

Lisbeth Björklund, BSc.
Doctoral student since 1985.
Research area: Structures of systems for knowledge organization and representation; the HYPERCAT project.

Manny Jägerfeld, BA.
Doctoral student since 1985.
Research area: (working on the WHO/LINFO project mentioned above).

Siv Söderlund, BA.
Administrative assistant since 1986.

Associated people:

Toomas Timpka, M.D. Doctoral student, Dept. of Medical Informatics, principal investigator in the LIMEDS project, a cooperative venture between LIBLAB and Dept. of Medical Informatics.

6.6 Publications in 1987

Reports:

LiU-LIBLAB-R-1987:1
Hjerppe, R.: LINS - LIBLAB's Name Handling System. A knowledge-based system for authority control of personal names according to AACR2, Ch.22, Headings for persons. June 1987, 8p.
In: C Bossmeyer, ed. The Library of the Future. Proc. ELAG (European Library Automation Group) 11th Library Systems Seminar, Frankfurt, April 1-3 1987. Deutsche Bibliothek, Frankfurt aM. 1987. ISBN: 3-922051-19-7. pp.67-80.

LiTH-IDA-R-87-04
Vainio-Larsson, A.: Datavetenskap : Teknik och Vetenskap. February 1987, 13p.

Working papers:

LiU-LIBLAB-WP:39
Hjerppe, R.: LIBLAB. Plan för verksamheten under första halvåret 1988. Oktober 1987, 4p.

LiU-LIBLAB-WP:40
Olander, B.: HYPERCAT notes. March 1987, 7p.

External papers:

Papers presented at IFLA 53rd General Conference, August 16-21 1987, Brighton, England :

Hjerppe, R.: Computer Networks as a Publication Medium? Implications for Libraries. August 1987, 4p.

Hjerppe, R.: IFLA and Professional Communication. August 1987, 3p.

7.

LOGPRO
The Logic Programming Laboratory

Jan Maluszynski
Professor of programming theory

7.1 Introduction

The Laboratory for Logic Programming was formally created in spring 1985 though research activities in logic programming at the department started much earlier. 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 laboratory 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.

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

Jan Maluszynski , Ph.D. *group leader, acting professor*
Douglas Busch, Ph. D. *visiting researcher, associate professor*
Wlodzimierz Drabent, Ph. D. *visiting researcher* (left in Jan. 1987)
Staffan Bonnier, *graduate student*
Simin Nadjm-Tehrani, *graduate student*
Ulf Nilsson, *graduate student*
Torbjörn Näslund, *graduate student*

Some of the research has been carried out in external cooperation with researchers at INRIA, France, Aiken Computation Lab., Harvard, USA and Institute of Computer Science of the Polish Academy of Sciences.

The main research activity has been concentrated around the project

The work in the Logic Programming Group is mainly supported by STU, The Swedish Board for Technical Development and by NFR, the Swedish Natural Science Research Council.

"Research in Efficiency of Logic Programming" funded by the National Swedish Board for Technical Development (grants STU-F 85-3166 and STU 86-3372). These grants expired June 1987. In spring 1987 a new project entitled "Logic Programming with External Procedures" was proposed. In fall 1987 the Swedish National Board for Technical Development decided to fund the project during the three years period (grant STU 87-2926).

The present research is partially based on the previous results:
- a study of two-level grammars as a logic programming language [Mal 84]
- a formal comparison of logic programs and attribute grammars [DeMa 85].

The following notions resulted from these comparisons:

7.2 The Objectives of the Present Research

It is often claimed that logic programming has great potential for reducing the cost of software development. One of the reasons supporting this claim is the declarative nature of logic programming. Since the control information need not be specified, the size of the code is often dramatically reduced in comparison with the size of algorithmic programs. However, the cost of software development also depends heavily on the possibility of re-using of existing modules, and on the methodology of programming. Thus, the problem how to combine logic programs with existing software is of great practical importance, but unfortunately its "ad hoc" solution may destroy declarative reading of programs. This in turn may create serious problems in reasoning about such programs and decrease their reliability.

The present research has two objectives:
- to develop a theoretical basis for a solution of the problem of re-usability of existing software in logic programming preserving the declarative nature of logic programs;
- to contribute to the methodology of development of correct logic programs with external procedures.

7.3 The Research Topics

7.3.1 Amalgamation of Logic Programs with Functional Procedures

The objective is to develop a systematic approach to writing logic programs which use external procedures. The motivation for this is twofold:
- to allow for re-using of existing (possibly imperative) software while still preserving the declarative nature of the top-level logic programs;
- though the Horn clause logic provides a universal computational paradigm it seems often quite unnatural to express functions in the relational formalism.

In recent years there have been a number of suggestions concerning combination of functional and logic programming in a single framework (see

e.g. [GrLi 86] or [BeLe 86] for a survey).
The approaches can be classified as:

(1) integrating existing programming languages and logic programs (well-known examples of this type are LOGLISP, QLOG, POPLOG and APPLOG);

(2) construction of new languages which allow one to define functions and relations and to combine functional and relational definitions. (well-known examples are EQLOG, LEAF and FUNLOG).

The main objective within the first approach is often to give access from logic programs to specific features of the underlying programming language, or programming environment. This aspect is usually more important than concern about the declarative semantics of the amalgamation. It may be rather difficult to give such a semantics if low-level features of the underlying system are accessible in the resulting language. On the other hand, many of the languages defined within the second approach have both declarative and operational semantics and some completeness results are also presented.

Our general perspective is different: we assume that we have given two arbitrary languages - one (not necessarily functional!) with functional procedures and the other one being a logic programming language. (As a matter of fact the suggestion presented below can be also applied to non-functional procedures if their interface with the environment is properly defined). It is our belief that a good combination of these two should result in a language which is as "conservative" as possible. Old programs (both their evaluation and meanings) should not change. It will allow us not only to think in a well established manner about things which have not changed but also to save a lot of work because we do not have to rewrite old programs in a new language. On the other hand we want to be able to build new logic programs employing calls to functional procedures and integrate them with the old ones. A basis for construction of an interface between the two different systems is the assumption that terms are their common data structures. A call of a functional procedure is itself a term. Since its execution is assumed to return a term we can view the underlying programming system as a term rewriting system. This permits the use of the theory of logic programming with equality (see e.g. [JLM 84]) to give a clean declarative semantics of the amalgamated language, and for application of E-unification in its interpreter. However, since we are not specific about the language of the functional procedures we have no access to the rewrite rules used by the system and we cannot use them for construction of E-unifiers. In [LeMa 86] we suggest a way for overcoming this difficulty. It is a new unification algorithm, called S-unification, which is a special incomplete case of E-unification. Let us compare our suggestion with the two kinds of approaches mentioned above.

The distinction between our work and the first approach is that the underlying programming system is considered as a black box: low-level features can be used in the underlying programs but not on the logical level. This makes it possible to give a relatively simple declarative semantics of the amalgamation. The difference between our suggestion and the second approach is that we are primarily interested in re-using functional procedures written in other languages, regardless of the type of the language (be it a pure functional

language, or an algorithmic language which admits functional procedures, like Fortran, Pascal or Ada) in a logic programming environment. In contrast to the systems of this category we assume existence of the term machine and we are able to use it in the top-level computational mechanism without being specific about its construction.

7.4 Methodology of Amalgamated Programming

We are searching for concepts and methods that facilitate development of correct logic programs including calls to external procedures. This work is based on existing ideas concerning logic programming.

Correctness of programs is usually defined with respect to some initial formal specification. Often it is suggested to develop correct programs from this specification by some formal transformations. In case of logic programs this approach has been studied in many papers (see for example [HaT 82], [Cla 81], [Hog 81]). A (pure) logic program is a logic formula and due to its declarative reading may be sometimes considered a specification of the problem. Therefore it is sometimes suggested to organize development of logic programs as construction of a derivation in the first order logic, where the initial formula is a complete specification of the problem and the final one is a logically equivalent logic program. However, a general difficulty with complete specifications is that they often do not properly reflect user's intuition and must be subject to changes. Therefore, testing of the program is necessary not to discover bugs, since the ideal development technique would guarantee correctness of the program with respect to the specification, but rather to confront user's informal understanding of the problem with the formal specification. The aspect of programmer's intuition is reflected in Shapiro's work on algorithmic debugging [Sha 83]. The debugging process is controlled by an "oracle" which answers the questions generated by the debugging system. These answers are statements concerning properties of the intended model. They should be satisfied by the program under development.

In our present research attention is focused on assertions which can be used to describe expected properties of a program being developed. The programmer is expected to selectively write assertions concerning those properties which she finds important. These assertions may play role of formalized comments. We plan also to continue our work [DrMa 87] on formal proof techniques for assertions. Some of the assertions, like type declarations and annotations may be used for automatic compile-time consistency checks during development of the program. For the others a run-time checker can be a useful tool in the debugging process. It may disclose improper behaviour of the program before obtaining an answer and it may be used by an algorithmic debugger as (a part of) an "oracle".

Development of a logic program can now be seen as updating of a database of clauses and assertions. A large number of updates may finally give a satisfactory resulting program with assertions either describing its "critical" properties, or providing a complete specification, if desired. At every step of this process the clauses should be consistent with the assertions. This will

improve reliability of the programs by elimination of some bugs. We are interested in studying formal methods for proving this consistency. These methods will contribute to everyday practice of informal reasoning and testing and to development of tools supporting these activities.

According to [DrMa 87] an assertion is viewed as a formula of a first order language. It specifies a relation on not necessarily ground terms. With each predicate (i.e. procedure) p of the program two formulas are being associated. One of them, the *precondition* characterizes the expected form of the arguments of p whenever p is called during execution of the program. The other one, the *postcondition* describes the expected instantiation of the arguments upon the success in relation to their form in the call. Thus, the preconditions are connected with the operational aspect of the program since they depend on the computation rule and on the search strategy. On the other hand, the postconditions can be related to both operational semantics and declarative semantics since they describe properties of the computed answer substitutions, which in correct implementations are also correct answer substitutions.

This notion of assertion can be used as a formal framework for investigation of the concepts of types and annotations appearing in the literature.

7.5 The Results

The following results were published in 1987:

7.5.1 A Restricted Class of Logic Programs [Dr 87]

We defined a class of logic programs which do not employ logical variable, work on ground terms and have particularly simple data flow. These are called simple logic programs. The aim of the experiment was to check how often logic programs belong to the class and to analyze the programming techniques which result in the programs which are not in the class. For this an analyzing program has been written which checks whether a given program is in the class. The sample of analyzed programs includes programs of different size. The results show that simple programs are used quite often and give insight into some techniques of using logical variables. Simple programs can be implemented in a very efficient way. In particular they allow to compile-out unification. However, this topic was not further developed.

7.5.2 Context-free types in logic programming ([KoMa 87], [Nä 87])

Our previous study of relations between logic programs and two-level grammars resulted in a notion of type for logic programs. The idea is to use context-free grammars for specifying data domains of logic programs. This amounts to considering many-sorted term algebras instead of one-sorted Herbrand universes. The nonterminals of the context-free grammars play the

role of the sorts.

In [KoMa 87] we illustrated on example the usefulness of this idea for development of logic programs.

Logic programs with data-domains specified by context-free grammars can be seen as a special class of two-level grammars. Using Cornell editor synthesiser we developed an experimental implementation of two-level grammars as a logic programming languge [Näs 87]. The system is an incremental compiler of two-level grammars into Prolog. Its intended use is to support development of typed logic programs and typed Definite Clause Grammars as well as to demonstrate the concept of two-level grammar at work.

7.5.3 A method for proving run-time properties of logic programs [DrMa 87]

Certain properties of logic programs are inexpressible in terms of their declarative semantics. One example of such properties would be the actual form of procedure calls and successes which occur during computations of a program. They are often used by programmers in their informal reasoning.

We introduced and proved sound an inductive assertion method for proving partial correctness of logic programs. The method formalizes common ways of reasoning about logic programs and makes it possible to formulate and prove properties which are inexpressible in terms of the declarative semantics. An execution mechanism using the Prolog computation rule and arbitrary search strategy (eg. OR-parallelism or Prolog backtracking) is assumed. The method may be also used to specify the semantics of some extra-logical built-in procedures for which the declarative semantics is not applicable.

7.5.4 S-unification: a basis for amalgamated programming [LBM 87]

Following the suggestion of [LeMa 86] an incomplete algorithm of E-unification, called S-unification was formally defined. It has been proved that if S-unification succeeds the result is a complete set of E-unifiers of the arguments (which due to our restrictions is a singleton). S-unification may also fail or report that it is not able to solve the problem of E-unification for given arguments. It has been proved that if it fails the actual arguments have no E-unifier. S-unification is a modification of Robinson's unification algorithm which allows the functors in the unified terms to be names of external procedures. Some unification steps may require reduction of the subterms which are procedure calls. However, it may happen that the arguments of the call are not fully instantiated, in which case the algorithm reports that it is not able to solve the unification problem.

7.6 References

References to LOGPRO papers

(These are only the papers quoted above, all LOGPRO publications can be found in the list of IDA publications in the Appendix).

[DeMa 85] P. Deransart and J. Maluszynski, Logic programs and attribute grammars, J. Logic Programming 2 (1985) 119-155.

[Dra 87] W.Drabent, Do Logic Programs Resemble Programs in Conventional Languages, LiTH-IDA-R-87-01, 1987, also in Proc. of IEEE SLP'87, San Francisco, September 1987.

[DrMa 87] W.Drabent and J. Maluszynski , Proving runtime properties of logic programs, Proc. of TAPSOFT'87, Pisa 1987, LNCS 250, 167-181.

[KoMa 87] J.Komorowski, J.Maluszynski , Logic programming and rapid prototyping, Science of Computer Programming 9 (1987), 179-205

[LBM 87] J. Leszczylowski, S.Bonnier, J.Maluszynski, Logic programming with external procedures: Introducing S-Unification, Revised Version, Information Processing Letters (to appear)

[LeMa 86] J. Leszczylowski, J.Maluszynski, Logic programming with external procedures: Introducing S-Unification, Rep. TR-86-21, Dept. of Computer Sc., Iowa State University, Ames 1986

[Mal 84] Maluszynski,J., Towards a programming language based on the notion of two-level grammar, *Theoretical Computer Science* 28 (1984), 13-43;

[Näs 87] T. Näslund, An experimental implementation of a compiler for two-level grammars, Proc. Int. Symposium on Methodologies for Intelligent Systems, Charlotte, NC Oct. 1987, North-Holland 1987.

Other References

[BeLe 86] M. Bellia, G. Levi, The Relation between Logic and Functional Languages: A Survey, J.Logic Programming, 1986:3:217-236

[Cla 81] K.L.Clark, The Synthesis and Verification of Logic Programs, Research Report 81-36, Dept. of Computing, Imperial College, London 1981.

[GrLi 86] Logic programming, functions, relations and equations, DeGroot D. Lindstrom G., Eds. Prentice-Hall, 1986.

[HaT 82] Å. Hansson and S.-Å. Tärnlund Program transformation by data structure mapping,
in: K.L. Clark and S.-Å. Tärnlund Logic Programming (Academic Press, London, 1982) 117-122.

[Hog 81] C.J. Hogger, Derivation of Logic Programs, JACM 28, 2, 372-392.

[JLM 84] J.Jaffar, J.L.Lassez, M.J.Maher, A theory of complete logic programs with equality, J.Logic Programming 1984:3:211-223

[Sha 83] E.Y. Shapiro, Algorithmic Program Debugging, MIT Press , 1983.

8.

NLPLAB
The Laboratory for Natural Language Processing

Lars Ahrenberg

8.1 Introduction

The interests of NLPLAB cover most aspects of the fields of Natural Language Processing and Computational Linguistics. Our theoretical research interests are primarily in the following areas: (i) parsing techniques for constraint-based formalisms, (ii) knowledge representation, including discourse representation, for natural language understanding, and (iii) the characteristics of man-machine interaction in natural language. One application area is presently of special interest to us, namely the construction and use of natural language interfaces (NLIs) to computer software. Research in this area is carried out within the project "Analysis and Generation of Natural Language Texts", financed by STU. The goal of this project is to develop a general-purpose NLI with ability to communicate in Swedish and English.

8.2 NLPLAB Personnel

Lars Ahrenberg, Ph.D., lab leader
Britt-Marie Ahlenbäck, secretary
Nils Dahlbäck, B.A.
Arne Jönsson, Tech.Lic., B.A.
Magnus Merkel, B.A.
Bernt Nilsson, research engineer
Mats Wirén, M.Sc., B.A.

The work in the Laboratory for Natural Language Processing is mainly supported by STU, The Swedish Board for Technical Development.

8.3 A Short Overview of Current Research

8.3.1 Parsing techniques for constraint-based grammars

An absolute requirement on a general-purpose NLI is that it can handle a fairly large number of the grammatical constructions of the languages that it communicates in. Ideally, the constructions that have been used in the construction of users' inputs should be recognizable in real time. For this to be possible we require a grammatical formalism which is both powerful enough to express the complexity of natural language constructions, yet sufficiently restricted so as to allow recognition and parsing by fast algorithms. Our strategy is to use declarative grammar formalisms, such as Lexical-Functional Grammar and PATR and find flexible and efficient parsers for such formalisms.

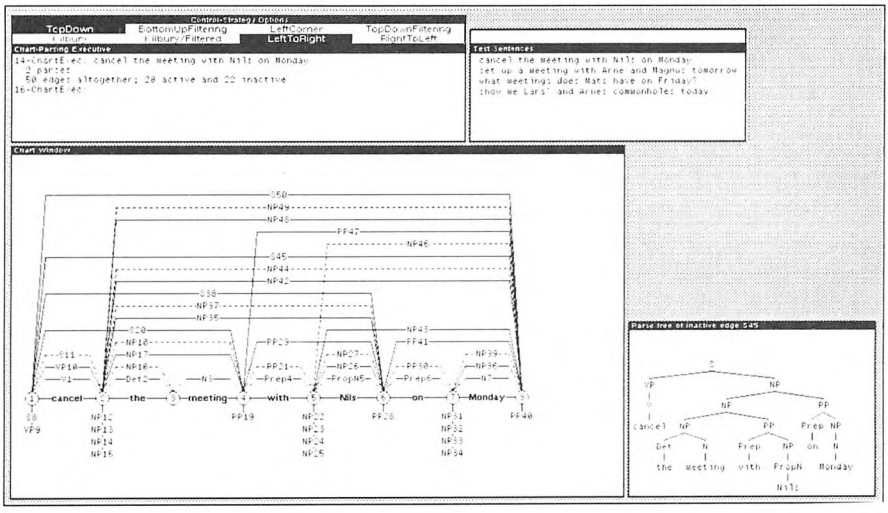


Figure 1: *Interface to a chart parser with open-ended control structure.*

Mats Wirén has studied the effects of utilizing open-ended control structures, where different choices in the control-strategy space can be tested (see fig. 1). In the first part of this work (reported in Wirén, 1987) different rule-invocation strategies in chart parsing were surveyed and tested, the conclusion being that significant gains in efficiency can be obtained by fine-tuning this strategy.

Further continuation and generalization of this work includes an implemented control-strategy independent PATR system (Wirén, 1988a), and an application of the PATR system to *on-line parsing*, a type of application which seems to profit significantly from an open-ended control structure (Wirén, 1988b).

8.3.2 LINLIN — architecture for a natural language interface

The Linköping Natural Language Interface (LINLIN) is being designed so as to be able to participate in a restricted dialogue with the user, not merely to interpret questions. The knowledge required to engage in a dialogue is varied and complex. It does not only comprise linguistic knowledge in a narrow sense (morphology, syntax, semantics) but also knowledge of the world (the domain of the background system), knowledge about how to participate in a dialogue, e.g. about how to refer to an object depending on its saliency in the discourse, or about what kinds of response are possible after a certain type of initiative from another dialogue participant. Moreover, these latter tasks presuppose that the agent is familiar with what may be termed the current *discourse state*, i.e. the objects that are talked about or otherwise salient and the current goals and direction of the dialogue. It is a difficult issue, not only to track down and give appropriate representation of this knowledge, but even more so to integrate it and make optimal use of it in a working system.

Our approach to this problem is to rely on a uniform representation format for object descriptions (both linguistic objects and domain objects), to use simple and general rule formalisms such as context free grammars with constraints to express linguistic knowledge and to employ constraint-propagation as the main processing formula.

In the pilot version of LINLIN, called FALIN, we are running an LFG chart parser in conjunction with a semantic network and a simple module for identifying and classifying objects on the basis of intrinsic properties as well as properties concerning their discourse status. While FALIN itself was not running during 1987, the major components (dictionary, morphological processor, grammar, parser, semantic network) have been completed. The system is expected to run in the spring of 1988.

The output of FALIN is a pair of a grammatical description (functional structure) and a content description. These two structures have the same format but contain different kinds of information. The functional structure describes the input utterance and its constituents as syntactic objects employing attributes for grammatical relations and morphosyntactic features. The content structure describes the utterance as a message with attributes representing basic object types, inherent properties, relations between objects in the domain as well as "discourse relations" between linguistic and non-linguistic objects (such as being the speaker, or the topic, of an utterance) (cf. Ahrenberg, 1987b).

The descriptors of the content structure are determined on the basis of (a) linguistic information, (b) semantic information (encoded in definitions for object types and attributes) and (c) pragmatic information encoded in the current dialogue state, which is treated as an object on a par with other objects.

8.3.3 Discourse representation

Discourse representation has a double interest to us. On the one hand we are interested in the computational modeling of a discourse state as explained in the previous section. Work in that direction has only just begun, however. On the other hand we have an interest in cognitive models of discourse.

The notion of *discourse representation* has come more and more into focus in cognitive science research on text understanding. There exist many approaches to the problem of discourse representation, and a number of theoretical notions such as frame, script, context space, focus space, discourse model have been put forward.

Nils Dahlbäck, in his thesis work, starts out from Johnson-Laird's theory of *mental models* and evaluates it as a theory of discourse representation both theoretically and empirically. The empirical studies comprise two experiments. The first experiment had subjects reading texts with referential discontinuities and investigated the role of background information in interpreting such texts. An interesting phenomenon showed up, namely that there seemed to be individual differences in cognitive strategies between subjects. These two strategies can be described as propositional vs. spatial. Background knowledge seemed more available for subjects using a spatial strategy. A second experiment was then designed to further investigate these individual differences. The data analysis for this experiment has just started.

The theory of mental models is also the starting point for an analysis of the concepts of 'symbol' and 'knowledge' as used in AI and cognitve psychology. It is argued that the word 'symbol' is used in two different senses, and that the confusions arising from this is one of the causes for the controversy on the symbolic nature of the mind and of the possible limitations of AI-systems (Newell, Dreyfus, Winograd).

8.3.4 The study of dialogues between human users and NLIs

A natural language dialogue between a computer and a human user by means of a natural language interface differs in important respects from human dialogues, spoken as well as written. To some extent these differences are known, but there is a need for empirical studies aiming at uncovering the similarities and differences between these types of dialogues. Among the questions we want answers to are: (i) What linguistic coverage do we need in a natural language interface? (ii) How much of the language is specific for different domains? (iii) What restrictions and limitations can we impose on the dialogue without it causing problems for the user?

To perform such studies we have developed tools for the simulation of NLIs as

well as for the analysis of dialogue. The simulation NLI (Dahlbäck & Jönsson, 1986) has been used in experiments to collect an initial set of 20 dialogues where the subjects were given different tasks to solve by means of a system which they believed were a database or advice system equipped with an NLI. Preliminary results of these experiments indicate that anaphora and ellipsis are common phenomena, that the linguistic variation is limited but that individual differences in style are important (Dahlbäck & Jönsson, 1988a, b).

For the analysis of the dialogues we have designed an interactive tagging system called DagTag (Ahrenberg & Jönsson, 1987). DagTag uses descriptors (attribute-value pairs) as tag units and is more general than most existing tagging systems in its ability to deal with constituents above (and between) word and sentence level. The different kinds of constituents recognized by DagTag are Dialogue, Sequence, Utterance, Clause/Move and Phrasal constituent. The relation between units on different levels is one of constituency, i.e. the system assumes that a Dialogue can be analyzed into a sequence of Sequences, which in turn can be analyzed into a sequence of Utterances, and so on. The category of Phrasal constituent is recursive. Thus, a tree structure is imposed on the dialogue during the analysis.

The system can be used for finding out quite subtle information from the analyzed dialogue, e.g. correlations between grammatical and functional properties. A set of dialogues can be searched by means of *a model dag*, a structure combining any kind of information on any set of the five structural levels of the system, and create statistics as well as concordances from the search.

Our future plans for DagTag include its extension into a semi-automatic system by the incorporation of a parser (presently, all tagging is done by the use of menues), and into a learning system whereby the analyses inputted manually are used by the system to extend its grammar and lexicon.

```
Interpret: i dag kl kvart över 10 på förmiddagen
(1988 1 15 2 FRI fm 10 15)
Interpret: om 2 år i mars
(1990 3 NIL NIL NIL NIL NIL NIL)
Interpret: på fredagen i vecka 24 nästa år
(1989 6 16 24 FRI NIL NIL NIL)
Interpret: om 2 timmar
(1988 1 15 2 FRI em 15 59)
Interpret: för 5 dagar sedan
(1988 1 10 1 NIL NIL NIL NIL)
Interpret: den 4/4 i år
(1988 4 4 14 MON NIL NIL NIL)
Interpret: i morgon kl 8
morgon eller kväll? kväll
(1988 1 16 2 SAT em 20 NIL)
```

Figure 2: *Sample interpretations from ClockWise.*

8.3.5 CLOCKWISE — a system that interprets temporal expressions

The language of an NLI naturally differs from one domain to another. But certain sublanguages, such as the language of times and temporal relations, recur from one application to another.

In natural language temporal information is expressed in a number of different ways. To refer to a particular day, for instance, one might provide information about a date, month and year, such as *January 1st, 1987.* This piece of information could be given in other forms: *the first of January, 1988, 01/01/1988, 1988-01-01, the first day of 1988.* Provided that speakers use the same calendar the above expressions will pick out the intended time point regardless of when the expression is uttered. Other expressions, such as *next week* or *tomorrow*, pick out different time points depending upon the time of speech, that is, the time referred to is determined by speech-time and the expression. These expressions are said to be deictic or indexical. Speakers also exploit background and contextual knowledge to a large extent. A phrase like *in December* will refer to a particular December of a certain year although a year is not specified in the same utterance. A detailed account of the structure of temporal adverbials is given in Merkel (1987).

We have constructed a system, ClockWise, that is capable of mapping temporal expressions in natural language into an absolute time location on a time axis. When ClockWise is fed with an expression such as *next week on Friday morning at six o'clock* the output would be a frame where values have been specified for periods such as year, month, day and hour. The values for the first three of these are calculated from the expression *tomorrow* starting from speech time which is taken from the computer's internal clock. ClockWise will also deduce what day of week the expression refers to and what week this day belongs to, though this is not part of the explicit information in the expression itself.

The syntactic variation for temporal expressions is large, which is indicated by the following examples:

On Monday next week at six o'clock.
Next week on Monday at six o'clock.
On Monday at six o'clock next week.
At six o'clock on Monday next week.
Next week at six o'clock on Monday.
At six o'clock next week on Monday.

ClockWise interprets all of these expressions as having the same temporal reference.

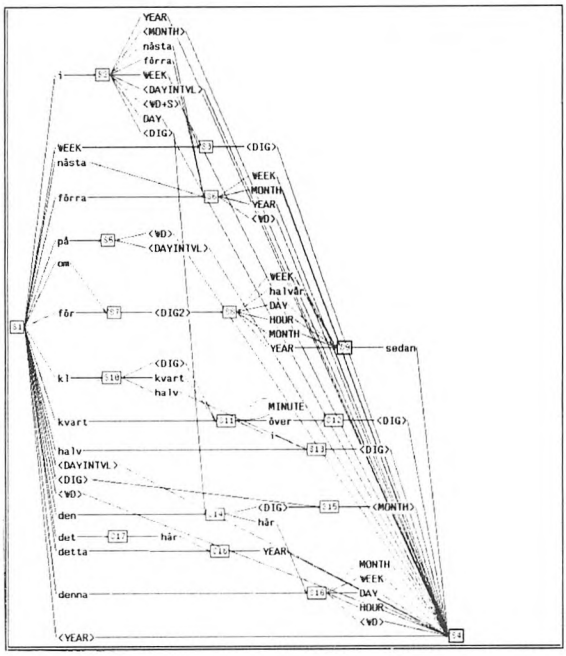


Figure 3: *The transition net used in ClockWise.*

The first version of ClockWise consists of a parser, based on a finite-state machinery, and a temporal expert which exploit its knowledge about temporal entities and relations and infers implicit temporal information. During the parsing process temporal data is stored in a notepad. When the notepad is filled the temporal expert will evaluate the data and output an absolute time.

A second version of ClockWise is under construction. This is based on an LFG-type grammar to allow for more compact expression and integration in a larger grammar.

8.4 External contacts and major events of 1987

NLPLAB likes to maintain contacts both with the research community of our field and with interested parties in other academic fields as well as in industry and society at large. We regularly participate in the meetings of SAIS (The Swedish AI Society), OFTI (The Area Group for Resaerch into Speech and Interaction) and the Nordic meetings of Computational Linguistics ("Nordiska datalingvistikdagarna").

In March IDA and NLPLAB hosted a symposium on Text and Courses of Events sponsored by the Humanistisk-Samhällsvetenskapliga Forskningsrådet (HSFR). The symposium was part of the planning phase of a new HSFR research programme, *Språk, Kommunikation och Teknologi* (SKOT) and focused on the various relations between textual structure and event structure. Invited speakers were *Jerry Hobbs*, Stanford, *Robin Cooper*, Edinburgh and *Bonnie Webber*, Philadelphia.

In August we took part in the symposium on "Effektiv Människa-Dator Interaktion" (Efficient Human-Computer Interaction) in Linköping with participants from Swedish industry and authorities where Arne Jönsson gave the speech on natural language interfaces.

Nils Dahlbäck and Sture Hägglund of ASLAB in cooperation have written a report for the MDA (Humans, Computers and Working Life) research program (Dahlbäck & Hägglund 1987).

In February *Lars Ahrenberg* defended his doctoral dissertation (Ahrenberg, 1987a). This happy event took place in Uppsala.

8.5 List of publications

The following are the publications referenced in the text above. For a full set of NLPLAB publications, please see the appendix at the end of the annual report.

Ahrenberg, Lars (1987a). *Interrogative Structures of Swedish. Aspects of the Relation between Grammar and Speech Acts.* Doctoral dissertation, Reports from Uppsala University department of Linguistics **15**, Uppsala 1987.

Ahrenberg, Lars (1987b). *Parsing into discourse object descriptions.* In *Proceedings of the 3rd Conference of the European Chapter of the Association of Computational Linguistics, Copenhagen April 1-3 1987,* pp. 140-47.

Ahrenberg, Lars and Jönsson, Arne (1987): *An interactive system for tagging dialogues.* Paper presented at the 14th international conference of the association for literary and linguistic computing, Gothenburg, June 1-5, 1987. Also as Research Report LiTH-IDA-R-87-22, Inst f datavetenskap, Linköpings universitet.

Dahlbäck, Nils & Hägglund, Sture (1987). *Människa datorsystem i samverkan: Probleminventering och förslag till forskningsinsatser.* Rapport framtagen för forskningsprogrammet Människa-Dator-Arbetsliv (MDA).

Dahlbäck, Nils & Jönsson, Arne, (1986). *A system for studying human computer dialogues in natural language*, Research Report, LiTH-IDA-R-86-42, 1986.

Dahlbäck, Nils & Jönsson, Arne (1988a). *Talking to a computer is not like talking to your best friend.* To be presented at The first Scandinivian Conference on Artificial Intelligence, Tromsö, Norway, March 9-11, 1988.

Dahlbäck, Nils & Jönsson, Arne (1988b). *Analyzing human-computer dialogues in natural language.* To be presented at The 3rd IFAC/IFIP/IEA/IFORS Conference on Man-Machine Systems, Analysis, Design and Evaluation. Oulu, Finland. June 14-16 1988.

Merkel, Magnus (1987). *The Interpretation of Swedish Temporal Frame-Adverbial Phrases.* Paper presented at the 11th Scandinavian Conference on Linguistics, Bergen 10-13 June 1987 and to appear in the proceedings.

Wirén, Mats (1987). A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing. *Proceedings of The Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark. Also as research report LiTH-IDA-R-87-13, Department of Computer and Information Science, Linköping University, Linköping, Sweden.

Wirén, Mats (1988a). A Control-Strategy-Independent Implementation of PATR. Paper accepted for presentation at The First Scandinavian Conference on Artificial Intelligence, Tromsö, Norway.

Wirén, Mats (1988b). On-Line Parsing with a Control-Strategy-Independent PATR System. Paper submitted to The 12th International Conference on Computational Linguistics, Budapest 22-27 August 1988.

9.

PELAB
The Programming Environments Laboratory

Bengt Lennartsson

The research in PELAB is currently concentrated on two areas:

- support for handling distributed software, and
- program transformations in the context of very high level languages

In this chapter we first give a general introduction to *Programming Environments* as a research area. Then there is a list of the current members of the PELAB group. After that we describe our general model, overlapping kernel projects, for the organization of our research. Finally there are presentations of the activities in three such kernel projects: a summary of a completed kernel project, DICE, Distributed Incremental Compiling Environment, a more exhaustive presentation of PEPSy, Programming Environments for Parallel Systems, in the middle of its lifetime, and finally some of the ideas behind MLSA, Multi-Level Software Architectures to be born. A reader who is mainly interested in our results during 1987 can skip to Section 11.3.

Programming Environments is a research area aiming at better understanding of and support for the software specialist's work on system design and implementation. In the academic environment new ideas are being tested, and new methods and tools are developed. A number of the research results have been used in production quality tools and systems now available on the market. For instance, the development of integrated incremental environments in the research community is the basis for commercial products like the Ada environment from Rational, Inc.

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

Programming environment research covers a wide spectrum of aspects, each needing its own research method:

- development of new incremental algorithms for program analysis, synthesis and transformation
- development of an appropriate representation of software systems and their components including representation of dependency relations among objects of different versions
- experimental development of the overall structure and architecture of integrated programming support environments
- experimental development of user interfaces including selection of functions that really support the software specialist and increase software quality and productivity

Development of efficient algorithms is done primarily through formal theoretical work. Design of an object management system has to be based on sound formal concepts, but also on experiences from tool integration and interaction in actual environments. The same applies to the overall structure and to the architecture of programming environments. The last aspect, the selection of functionality and the design of user interface, however, has to be investigated mainly experimentally. Prototypes have to be implemented, sufficiently robust and complete for evaluation.

The evaluation of prototype implementations, however, generally need to be on a small scale for two reasons. Firstly, in the academic environment we don't have very big systems and projects to play with. It is not realistic to organize projects with hundreds of people producing hundreds of thousands lines of code just for tool and method evaluation. Secondly, response time is a very essential parameter for user satisfaction in general and also for the tuning of functionality and of dialogue design. The optimal border line between what should be done by the system and what should be done by its user is a trade-off depending on performance of both parties. Programming environment research is aiming at investigating and developing methods, tools, and systems appropriate for industrial use several years ahead in the future. At that time the computers used for industrial software development will be some order of magnitude more powerful than what is available in the academic environment today. Hence, to have realistic response times in the evaluation of academic prototype environments, the size of the application has, in general, to be of moderate size.

The rapidly decreasing price/performance ratio for hardware, processing power as well as memory, continuously moves the border-line between what is possible and impossible. This applies both to the experimental components of programming environment research and to the industrial software development environments.

9.1 PELAB Personnel 1987

The members of PELAB share their time between undergraduate education and research. The research part is 20 to 80 per cent of full time. It varies from person to person, and also from one year to the other.

Bengt Lennartsson, PhD 1974, lab. leader
Gunilla Lingenhult, secretary

Supervisors:

Peter Fritzson, PhD 1984
Anders Haraldsson, PhD 1977

Employed graduate students:

Rober Bilos, licentiate 1987
Johan Fagerström, licentiate 1986
Bengt Karlstrand, BSc
Mariam Kamkar, licentiate 1987
Yngve Larsson, MSE
Nahid Shahmehri, licentiate 1987
Lars Strömberg, MSE
Ola Strömfors, licentiate 1986

Associated persons:

Ulf Cederling, University of Växjö
Pär Emanuelson, Epitec AB
Kristina Ernstsson, lecturer IDA
Azadeh Ghaemi, Programsystem AB
Tommy Olsson, lecturer IDA
Kjell Post, undergraduate student
Tom Rindborg, Softlab AB
Dick Schefström, TeleLogic AB
Dan Strömberg, Swedish Defence Research Institute
Jerker Wilander, Softlab AB

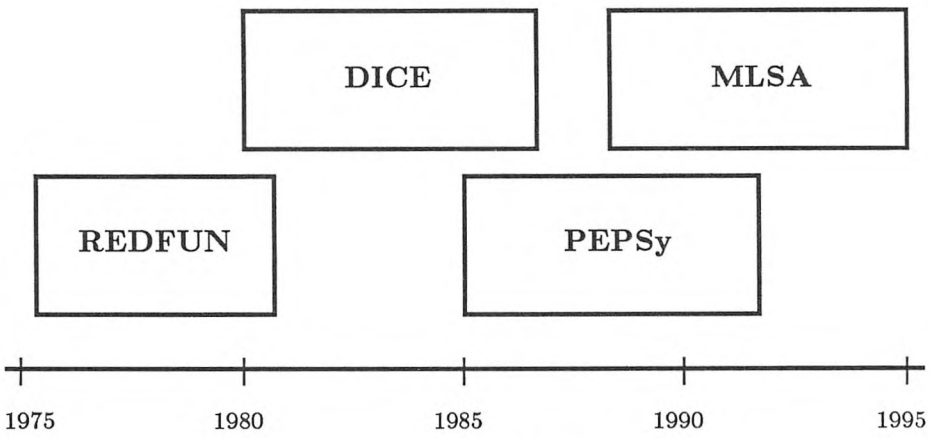
9.2 Overlapping Kernel Projects

In programming environment research we have to change our focus of attention as hardware development rapidly moves the border between what is possible and not. In PELAB we do this, not continuously, but stepwise every three to five years. Our organization is based on *kernel projects*. The philosophy behind kernel projects is to let the students work on individual but related problems, and that their work should be based upon and contribute to the accumulated knowledge and experience in the group.

Before we start a new kernel project we spend a considerable time on defining it in terms of an appropriate area for research, a set of problems or aspects we consider relevant. The lifetime of a kernel project is about five years. In the

first phase, two to three years, emphasis is on developing concepts and models, often by a sequence of iterations where some of the ideas are tested in prototype implementations. However, in this early phase it is typically a job only for one or two persons, frequently discussing with the rest of the group.

In the PELAB history there have been two successfully completed kernel projects, REDFUN and DICE initiated by Anders Haraldsson and Peter Fritzson, respectively. There is one in progress, Programming Environments for Parallel Systems, PEPSy, in the transition from its first to its second phase. There is also a new one to be started, currently labeled MLSA for Multilevel Software Architectures. It is instructive to look at the kernel projects and to study when they have resulted in publications.



Overlapping Kernel Projects

In the early phase of each kernel project external publications have occurred sparsely. After a solid basis, in the form of a PhD thesis work, however, it has turned out to be easy for others to join, to do interesting work, and to publish results. Anders Haraldsson's work on partial evaluation in the REDFUN project inspired Pär Emanuelson and Jan Komorowski to apply partial evaluation and the REDFUN implementation to pattern matching and to logic programming, respectively. In the same way Peter Fritzson's work on the Distributed Incremental Compiling Environment, the DICE project, generated thesis work for Johnny Eckerland on retargeting the incremental code generator, for Rober Bilos on token-based program representation, and for Mariam Kamkar and Nahid Shahmehri on program flow analysis. PEPSy has just now after two years evolved to a state where the ideas and experiences can be presented at international conferences.

We have found that in a group of the size of PELAB, it is convenient to have overlapping kernel projects. In order to be able to annually show results to our sponsors and to smoothly accept new graduate students, it seems to be optimal

to set up a new kernel project when the current one starts to generate publications.

We present the idea of overlapping kernel projects not only to explain the work in PELAB but also as a possible more general model for combined experimental and theoretical research in rapidly evolving areas.

9.3 Research Projects

The role of the kernel project is mainly to form a conceptual and social environment for the researchers and graduate students. Ideas and experiences are inherited and propagated across project borders. For instance, results in program manipulation in the REDFUN project, 1975-1981, will be highly relevant for the MLSA project we are planning to start.

9.3.1 The DICE Project *(Peter Fritzson et. al.)*

One goal in the DICE project (*Distributed Incremental Compiling Environment*) was to design and implement an appropriate architecture for a full scale integrated environment supporting the development of programs coded in block-structured languages and executing on a separate connected target.

The main results are:

- **the flexibility normally available in an interpreting system can be achieved in a compiling system**
- **the functionality of a high level target debugger can be obtained via the incremental compiler without any target code instrumentation, and without a target resident debugger at all.**

The design and implementation of the DICE system was done during 1980-1984. Some related problems have been studied and reported since then.

During 1987 Rober Bilos completed his work on token-based program representation. A primary goal with this work has been to investigate the consequences of token-based program and document representation. It was found that a program or a document represented as a token sequence saves on the average 50% memory space compared with a textual representation. Furthermore, program representations such as trees or graphs must in general be linearized in order to be processed by current delta algorithms. The token sequence is an alternative linear representation, and trees, for instance, can be recreated very quickly by the use of very fast LR parsing methods. Another advantage, compared to textual representation, is that deltas between program versions stored in source code control systems become insensitive to changes in whitespace or formatting style.

In 1987 Mariam Kamkar and Nahid Shahmehri completed their work on program flow analysis as a basis for interactive queries of programs. It has been found that data flow problems appropriate for query applications often need to keep track of paths associated with data flows. By contrast, flow analysis in conventional compiler optimization applications tend to collect just summary flow information.

Ola Strömfors' work on program editors has been quite independent of the DICE project, but on related problems. During 1987 he has been employed part time by Programsystem AB. He has worked together with Dick Schefström at TeleLogic, and his powerful editor, ED3, is now an integrated part of the next generation of the Ada environment, ARCS, from TeleSoft-TeleLogic.

Dan Strömberg completed his licentiate thesis during 1987. Part of his work was done in the framework of DICE when he was a PELAB member some years ago. In his thesis he also presents results from his current research at FOA, the Swedish Defence Research Institute.

Recent DICE-related publications:

Rober Bilos: *Incremental Scanning and Token-based Editing.* Licentiate Thesis No 108. Linköping Studies in Science and Technology. Linköping 1987.

Rober Bilos, Peter Fritzson: Experience from a Token Sequence Representation of Programs, Documents, and their Deltas. *Proceedings of the International Workshop on Software Version and Configuration Control.* Grassau, FRG. Jan. 1988.

Mariam Kamkar, Nahid Shahmehri: *Affect-Chaining in Program Flow Analysis Applied to Queries of Programs.* Licentiate Thesis No 118. Linköping Studies in Science and Technology. Linköping 1987.

Mariam Kamkar, Nahid Shahmehri, Peter Fritzson: Affect-Chaining and Dependency Oriented Flow Analysis Applied to Queries of Programs. To appear in the *Proceedings of the ACM Symposium on Personal and Small Computers.* Cannes, France. May, 1988.

Sven Moen: *Drawing Dynamic Trees.* Master Thesis. Department of Computer and Information Science, Linköping University. October 1987.

Kjell Post: *DIESEL - Lab Course in Compiler Construction.* Master Thesis. Department of Computer and Information Science, Linköping University. November 1987.

Dan Strömberg: *Transfer and Distribution of Application Programs.* Licentiate Thesis No 126. Linköping Studies in Science and Technology. Linköping 1987.

Dick Schefström, TeleLogic: The System-Oriented Editor - A Tool for Managing Large Software Systems. *Using Ada: ACM SIGAda International Conference Proceedings.* Boston, MA. Dec. 1987.

9.3.2 The PEPSy Project *(Johan Fagerström et.al.)*

The general goal of the PEPSy project is to develop methods and tools for the design and implementation of software for parallel and distributed systems.

The PEPSy paradigm is based on the following characteristics of distributed software systems:

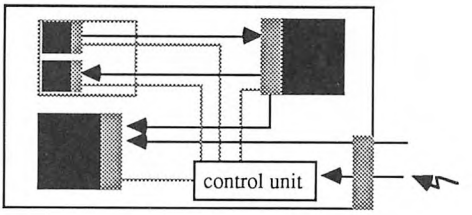
- **Large size.** Distributed systems often consist of a large (dynamically changing) set of modules. Keeping track of these modules is difficult both for the developer and for the support system.
- **Changing environment.** Changing a system often destroys structures designed into it. It also tends to make documentation out-of-date.
- **Natural and introduced non-determinism.** Varying time-delays and user-introduced guarded expressions lead to non-deterministic behavior.
- **Time-dependent behavior.** A debugger will slow down some parts of the system. Modules depending on time (e.g., a time-out) will be affected even if not debugged.

The work so far has concentrated on the first two problem areas. In the following we present a paradigm and a programming environment in which testing is done using problem-oriented terms. Complexity is controlled using an abstraction mechanism. The paradigm supports inter-process debugging using 'borders' around objects similar to [Smith 81]. His system-defined borders have been extended with user-defined borders around sets of objects.

Complexity in distributed systems is also tackled by insisting on (and enforcing) a hierarchical object-oriented design with well-defined interfaces between components. The paradigm also introduces so called 'control units' which support structured and localized changes in designs. The approach is similar to [LeBlanc 85]. However, we have concentrated on integrating the paradigm with a programming environment.

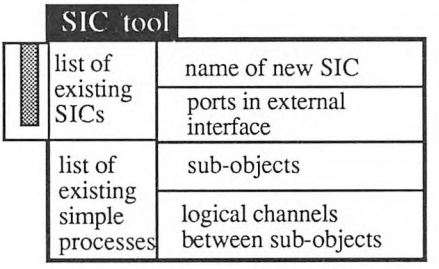
Our paradigm is based on *processes* which are combined into so called *software ICs*. The nature of the processes is language dependent, but our paradigm is independent of any particular programming language. The processes have an *interface* and they communicate via *logical channels*. These channels are uni-directional. A software IC is a set of communicating processes encapsulated into a black box by means of a so called *control unit*. The control unit is responsible for linking and un-linking logical channels inside the software IC. It is also used to construct well-defined states locally inside the software IC, so called clean points.

A complete description of the model and its implementation and integration with a programming environment will be presented in Fagerström's PhD thesis 1988.



A Software IC

Traditional tools like editors, compilers, interpreters, linkers, and loaders play important roles in program development. They must be re-evaluated in the distributed case. For example, the optimizing pass of a compiler might try to add code to increase parallelism and thus performance [Hibbard 82]. 'Objects' provide an appropriate component for editing, compiling, and distributing. Incrementality can be provided at varying granularities. In the DICE project it was on the statement level. In PEPSy it is on larger objects, processes, of varying sizes. We have two editors, one at the process level, language dependent, and one language independent configuration editor used to create software ICs.



Editor for Software ICs

When entered, the configuration editor automatically creates a sub-class of the class control unit, an external interface, and various control channels to the external world. The programmer can then use the editor to record (by pointing and by using menus) the appropriate configuration code executed by the control unit when it is started. Commands include creating and deleting sub-objects, and linking and un-linking channels between interfaces. The code for this, the configuration code, is generated and stored in the control unit. In the same way code for tracing, conditional or unconditional, and for demons can be generated and installed from the editor on ports, interfaces and channels. Probes and interface testers are other tools in our model.

A prototype of the PEPSy model has been implemented in Smalltalk-80. Present and future work includes refinement of the model and the tools. Future implementations will build on CONIC [Sloman 85] on a network of SUN work stations.

Recent PEPSy publications:

Johan Fagerström: Design and Test of Distributed Applications. *Proceedings of the 10th International Conference on Software Engineering.* Raffles City, Singapore. April 11-15. 1988.

Johan Fagerström, Lars Strömberg: A Paradigm and System for Design and Test of Distributed Applications. *Proceedings of IEEE Compcon Spring'88.* San Francisco, CA. Feb. 1988.

Johan Fagerström: *A Paradigm and System for Design of Distributed Systems.* Forthcoming PhD Thesis. Linköping Studies in Science and Technology. Dissertations. 1988.

Johan Fagerström: Enabling Structured Debugging of Distributed Systems. To appear in the *Proceedings of the Workshop on Parallel and Distributed Debugging.* Madison, WI. May. 1988.

Yngve Larsson: A Testbed Environment for Debugging Distributed Systems. To appear in the *Proceedings of the Workshop on Parallel and Distributed Debugging.* Madison, WI. May. 1988.

References:

Hibbard 82] P.G. Hibbard, T.L. Rodeheffer: Optimizing for a Multiprocessor: Balancing Synchronization Cost against Parallelism. *Proceedings of the 5th International Symposium on Programming.* Torino, Italy. 1982.

[LeBlanc 85] T.J. LeBlanc, S.A. Friedberg: *Hierarchical Process Composition in Distributed Operating System.* Computer Science Department, University of Rochester.

[Sloman 85] M. Sloman, J. Kramer, J. Magee: The CONIC Toolkit for Building Distributed Systems. *Proceedings of the 6th IFAC Distributed Control System Workshop.* Monterey, CA. May. 1985.

9.3.3 Next Kernel Project

We have not yet definitely decided about the kernel project to follow PEPSy. What we are considering is some kind of amalgamation of what is often called *the transformational approach* and *very-high-level-languages.* Currently we are collecting material and investigating different ideas. The working title of the new activity is Multi-Level Software Architectures, MLSA.

The background is that we imagine that software specialists designing systems in the future will define their own notions and operations tuned to the specific problem or application domain, rather than implementing very large systems in a standardized general purpose programming language. In our vision there are "languages" on several levels, and the software specialist designs and implements these languages and bidirectional translators for transferring information about data structures, operations, and current state, between the different levels. We are aiming at a software architecture where the topmost

level, closest to the application, will be expressed in application oriented concepts; that the application domain expert will be able to take care of most of the tuning and maintenance himself.

We have started some experiments to find out whether the Refine system from Reasoning Systems, Inc. is an appropriate research vehicle for further work. The Refine language may be appropriate for executable high level specifications of language semantics as well as of tools.

One possible experiment is to generate incremental symbol processing modules of compilers from high-level specifications, using the transformational approach. Another idea discussed tentatively is to construct more powerful partial evaluators, using transformational techniques combined with results from flow analysis.

References:

Douglas Smith, Gordon Kotik, Stephen Westfold: Research on Knowledge-Based Software Environments at Kestrel Institute. *IEEE Trans. on Software Engineering.* Vol SE-11, No 11. Nov. 1985.

Reasoning Systems: *Refine 2.0 Users Guide.* Reasoning Systems Inc. 1801 Page Mill Road, Palo Alto, CA 94304. Sept. 1987.

Peter Fritzson: *Incremental Symbol Processing.* Research report in preparation. Submitted to IEEE Software.

10.

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 often a rather ad hoc activity. Logic (and discrete mathematics) with suitable extensions, may be applied to strengthening the theoretical basis for knowledge engineering. It is the objective of RKLLAB to contribute in that respect.

10.1 Researchers and Projects.

10.1.1 Activities

The activities of RKLLAB during 1987 have been in the following, overlapping and interacting areas:

Non-standard logics and their implementations, in particular:
- non-monotonic logic and reason maintenance
- logic of uncertainty
- constraint programming systems

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

Professional knowledge and information management systems
- the LINCKS project

Representation of knowledge about machinery and processes, in particular:
- theoretical analysis of action structures
- introductory studies of plan guided vehicles
- plan guided manufacturing systems

10.1.2 Laboratory members.

The following researchers have been members of RKLLAB during 1987 (or a part of the year):

Laboratory leader:
Erik Sandewall

Project leaders and senior graduate students:
Dimiter Driankov
Jim Goodwin
Jalal Maleki
Lin Padgham
Michael Reinfrank
Ralph Rönnquist

Graduate students and masters thesis students:
Christer Bäckström
Patrick Doherty
Christer Hansson
Peter Haneklou
Johan Hultman
Arne Stahre
Ulf Söderman
Peter Åberg

10.1.3 Main current achievements.

The following degrees were awarded to RKLLAB members during 1987:

- *Jim Goodwin* completed his Ph.D. The title of the thesis was: "A Theory and System for Non-Monotonic Reasoning"

- *Jalal Maleki* finished his licentiate degree. The title of the thesis was: "ICONStraint, A Dependency Directed Constraint Maintenance System"

- *Ralph Rönnquist* finished his licentiate degree. The title of the thesis was: "Network and Lattice Based Approaches to the Representation of Knowledge".

In addition there has been the following major achievements in terms of finished research results during 1987:

a) work on reasoning under uncertainty, described in more detail below (Driankov)

b) extension of the theory of action-plans, making it possible to characterize parallel, interacting actions (Bäckström)

c) development of a theory of semantic states (Sandewall)

d) completion of the LINCKS hyperobject database system (Padgham, Rönnquist, Stahre, Åberg)

Results which were described in the annual research report for 1986, but with a reference only to a departmental report or manuscript, have reached international publication during 1987. This applies to the departmental reports 86-02 (by Rönnquist), 87-05 (by Bäckström), 87-06 (by Hultman), and to the manuscript by Sandewall mentioned there.

A total of 12 papers by RKLLAB members were published in international journals or conferences during 1987, as listed in the appendix E.

In this chapter we shall first give a brief overview of the different activities during 1987, including not only finished results but also ongoing projects, and then give a detailed account of one project namely Driankov's work on reasoning under uncertainty.

10.2 Focal point of research: Plan-Guided Systems.

During 1987 the lab's research projects and researchers have given special attention to the special topic of Plan-Guided Systems ("Planstyrda system"). From a research perspective, this term can be understood as including:
- the A.I. topic of knowledge based planning
- the topic of autonomous agents as understood in A.I.
- interactions with the field of automatic control
- interactions with the research on multi-sensor data fusion
- software engineering aspects of intelligent robots

From the application point of view, we define a plan-guided system as a system (for example an unmanned vehicle, or an "intelligent" automatic manufacturing cell) which is able to accept an assignment or request, make a plan for how to carry out the assignment, execute the plan, recognize problems which may impede the plan execution, revise the plan if necessary, and report success or failure.

It is clear that A.I. and knowledge engineering techniques are only one part of what is needed for designing plan guided systems. Automatic control, sensor

technology including sensor data fusion and often computer vision are also needed. Our point is however that the successful design of PGS requires a tight connection of results from these various fields. It is not sufficient to let the specialists in the various fields build each their part of the total system. In particular during 1987 we have opened a dialogue with our university's division of automatic control, headed by professor Lennart Ljung who is a world-class authority in his field. It is too early to report any results from these contacts, but it seems already that there are significant research topics of great common interest.

The research on plan guided systems also builds of course on our own work during earlier years. Knowledge based planning is one essential aspect of PGS, and relates to reasoning about time and action, and therefore also to non-monotonic reasoning. More about this research follows below.

The logic of uncertainty is also of great potential importance for plan-guided systems, since they have to deal with uncertain information about the environment they inhabit. This combination has not been pursued yet, awaiting that Dimiter Driankov will finish his thesis, but may become important in the future.

10.3 Non-standard logics and their implementations.

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 extensions over the "standard":

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

- special reasoning mechanisms, such as default reasoning (where conclusions are drawn from the absence of certain knowledge), 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.

10.3.1 Non-monotonic logic and reason maintenance.

Work in this area has been done by Erik Sandewall and Michael Reinfrank during the year. We refer to the publication list, and foresee a more extensive account in a later annual report.

Michael Reinfrank has held a course on Reason Maintenance and prepared a set of lecture notes for the course.

10.3.2 Logic of uncertainty.

Dimiter Driankov has continued his research in this area during the year, and aims at finishing his Ph.D. during 1988. See the special "feature" at the end of the RKLLAB chapter, for a detailed presentation.

10.3.3 Constraint programming systems.

Jalal Maleki presented his licentiate thesis during 1987. (The licentiate is intermediate between M.Sc. and Ph.D.) Afterwards, he has studied the possibilities of applying constraint programming techniques to temporal reasoning.

10.4 Professional knowledge and information management systems

- the LINCKS project.

This activity has previously been listed as "Office systems". We change the term in order to signal an increasing involvement with specific professions using the systems. During 1987 such contacts were established with medical users; other contact areas may be forthcoming.

The class of applications addressed by this research is *profession specific tools for management of information and knowledge.* Such tools should not only be able to handle fully structured knowledge, as used in knowledge-based systems, but also knowledge represented as free text, pictures, or hypertext. The core issue for this research is therefore the *appropriate representations for data and knowledge*, such that it can both be communicated to the human user, and "understood" (= processed) by the computing system. This topic includes all of the following aspects:

- the "local" information that the user formulates and uses himself or herself;
- the encyclopaedic information (textbooks, etc.) that the user needs to access and annotate in the course of his work;
- communication with colleagues, clients, and others. Convenient access to "local" information for inclusion in messages; convenient administration of incoming information;
- convenient access to numerical computation and other computer based services for processing the information.

We wish to explore the possibility of local autonomy in professional knowledge management systems, meaning that the system would perform certain routine actions by itself although within the guidelines and policies formulated by the user. This would result in one type of plan guided system, operating in a world of information rather than a mechanical world.

During 1987, the LINCKS group has concentrated on completing the shared LINCKS 'hyperobject' repository. Hyperobjects are data base objects similar to what one finds e.g. in the Notecard system from Xerox and the Hypercard system from Apple. Thus a hyperobject may have attributes (strings, numerical values, etc.), pointers or links to other hyperobjects, and also a text content whose length is typically one or a few paragraphs. A longer text is constructed as a set of hyperobjects, which are connected in the obvious ways.

The LINCKS hyperobject repository is a server system which runs on a SUN computer, and can be accessed from multiple workstations over the Ethernet. Each hyperobject may exist in several versions, reflecting different generations of update. Generations do not have to be linearly ordered: if several workstations fetch a hyperobject to their local workspaces, work at them independently, and later put back modified versions of the same hyperobject, then they will be represented as parallel and independent updates.

Several end-user systems that communicate with the repository have been implemented, namely:
-- an Interlisp based system running on a Xerox 1186 workstation
-- a Forth based system running on a Data General portable computer
-- an extended Forth based system running on a MacIntosh computer
-- a system in 'C' running on the same SUN computer as the repository itself, supporting "dumb" terminals.

The hyperobject repository is a robust and reliable implementation, and has also been used by a spin-off company, Programsystem AB, in a project developed for a customer. The present end-user systems however should be seen as experimental. They will be succeeded by more definite, profession specific implementations during 1988.

10.5 Representation of knowledge about machinery and processes

10.5.1 Reasoning about time and action.

During previous years, Erik Sandewall and Ralph Rönnquist have done research on action structures, i.e. plans where actions may occur both in sequence and in parallel. The key idea is that each action is characterized in three ways: by its precondition, post-condition, and prevail condition. The first two are standard devices; the prevail condition expresses what must hold constantly while the action is executed. Each of the three conditions is seen as a partial interpretation.

For purely sequential action, the prevail condition is insignificant since it does not matter what happens between the beginning and the end of each action. With prevail conditions, we were able to characterize processes where actions

are allowed to occur in parallel (because they do not disturb each other), while not being required to occur in parallel.

The previous results were however not able to characterize processes where several actions are required to occur in parallel. For example, in order to open a closed door, we must pull (or push) the door open while depressing the handle (European type of handle assumed). Only pulling the door, or only depressing the handle, will usually not cause the door to open.

During 1987, Christer Bäckström has extended the previous theory so that it also accounts for such interdependent actions. He achieves this by introducing a fourth condition, called a keep condition, besides the precondition, post-condition, and prevail condition.

Also during 1987, Erik Sandewall has rewritten the previous results, with some extensions, as a set of lecture notes. Peter Struss, visiting from Siemens AG, has prepared a set of lecture notes for his course on qualitative reasoning. Patrick Doherty has prepared a survey paper of earlier work on temporal logic, as a basis for his continued work in the area.

10.5.2 Introductory studies of plan guided vehicles.

Several of the younger graduate students in the lab have participated in introductory studies of plan guided vehicles. Major parts of these studies have been done in the context of the A.I. branch of the pan-European Prometheus program, a program for research cooperation between European automobile manufacturers. Other parts have been done in cooperation with groups in the Linköping branch of the Research Institute of National Defense (FOA).

10.5.3 Plan guided manufacturing systems.

Johan Hultman has continued the work on the COPPS system, a prototype software system for controlling machinery. This system was also described in last year's report. The design of COPPS is directly based on the action structure theory described under 10.5.1. The implementation consists of two parts, a lower layer which does the actual control, and an upper layer which handles planning, dialogue, etc. The lower layer has been finished and was reported during 1987; work during the year concentrated on the upper layer.

10.6 International activities.

RKLLAB has taken part in the COST 13 collaborative project number 21, on "Advanced issues in knowledge representation", together with Brussels, Pisa, and Rome. In the context of this project, we are organizing a "Second international workshop on non-monotonic reasoning", to be held in Munich on 13-15 June, 1988. The workshop is co-sponsored by COST 13, AAAI, and

Siemens AG. Michael Reinfrank is in charge of the arrangements.

Michael Reinfrank also organized a "Summer course on non-monotonic reasoning" on June 1-4, 1987, with 23 participants.

RKLLAB organizes a one week summer course on "The modelling of uncertainty in expert systems" on May 30 - June 3, 1988. The course is sponsored by Nordisk Ministerråd. Dimiter Driankov is in charge of the arrangements.

The following researchers visited RKLLAB during 1987: Allan Brown (General Electric), Peter Gärdenfors (University of Lund), Kurt Konolige (SRI International), Dag Prawitz (University of Stockholm).

10.7 Special feature. Reasoning under uncertainty: Towards a many-valued logic of belief

Dimiter Driankov

The rest of this section consists of a special feature paper by Dimiter Driankov on resoning under uncertainty.

REASONING UNDER UNCERTAINTY: TOWARDS A MANY-VALUED LOGIC OF BELIEF

1. State Of The Art In Reasoning Under Uncertainty.

The existing approaches in reasoning under uncertainty can be divided in two major classes: *numerical* and *symbolic*. The *numerical* approaches impose harsh restrictions upon the type and structure of the data on which inferences are based: it is usually required that the data should form a measurable-space so that an additive or non-additive set-theoretic measures of uncertainty can be defined. On the other hand, they all represent uncertainty as a precise quantity (scalar or interval) on a given scale, thus requiring *precise* yet *consistent* numerical assessments of the uncertainty associated with the validity of *only* atomic sentences and of their relations. Given the difficulty in consistently eliciting such numerical values, it has become clear that these approaches require an unrealistic level of precision that does not actually represent a real assessment of uncertainty. Among these approaches, one can distinguish three distinct types: the *one-valued*, the *two-valued*, and the *fuzzy-valued* approach. The *one-valued* approaches include some of the more traditional techniques: *Bayes Rule*, *Modified Bayes Rule* and *Confirmation Theory*. A more recent trend is exemplified by the *two-valued* approaches: *Dempster-Shafer Theory*, *Evidential Reasoning*, *Probability Bounds* and *Evidence Space*. Finally, the *Fuzzy-valued* approaches include: *Necessity and Possibility Theory* and *The Linguistic Variable Approach*. Within all these approaches it is possible to define a *calculus* that provides the mechanism for propagating the uncertainty in the data through the reasoning process. Similarly, the use of aggregation operators provides a summary of the information, which can then be ranked against other summaries for performing rational decisions. However, none of these can provide a clear explanation of the *reasons* that led to a given conclusion.

Formal approaches based on *symbolic* representations, are designed to handle a very specific type of uncertainty that derives from the *incompleteness* of information. They have a corresponding logic theory that determines the mechanism by which inferences (theorems) can be proven or *believed* to be true. Thus, they concentrate on the set of tentative conclusions that might be drawn from a less then compelling evidence, as well as the inference rules that are able to produce such conclusions. Furthermore, when certain tentative conclusion used in the deductive process is found to be false, mechanisms built into a so-called *truth-maintenance system* are used to keep the integrity of the data-base of sentences. These formal symbolic representations go under the name of *non-monotonic logics* and a number of distinct approaches have emerged, namely: *NML-I*, *Default Reasoning*, *Autoepistemic Reasoning*, and *Circumscription*. However, as it has been recognized, these approaches lack any facilities for computing *degrees of belief*, which "...may be necessary for summarizing the structure of large sets of admissible extensions as well as for quantifying confidence levels". In contrast to the *numerical* approaches, the *symbolic* ones are more suitable for providing a trace from the sources of the uncertain information through the various inference paths to the final conclusion.

Bonissone, has proposed a list of requirements that should be satisfied by the *ideal* formalism for representing uncertainty and making inference with uncertain information. These are as follows:

Representation Level

1. There should be an explicit representation of the *amount* of evidence for *supporting* and for *refuting* any given sentence, or in other words, an explicit measure of the amount of *belief* and *disbelief* in the validity of a sentence;
2. There should be an explicit representation of the information about the evidence, i.e, *meta-information*, such as evidence source, the reasons for *supporting* and for *refuting* a given sentence, etc. This *meta-information* will be used by the *control level* to remove conflicting assertions provided by different sources;
3. There should be no global requirement that the measures of *belief* and *disbelief* have to be *complementary*, i.e., the fact that the evidence available supports a *belief* to a degree x should not mean that it does support a *disbelief* to a degree 1-x.
4. There representation should allow for describing the uncertainty of the information at the available level of detail, i.e., allowing *heterogeneous information granularity*.

5. There should be *explicit* representation of *consistency*. Some measure of compatibility should be available to detect trends of potential conflicts, and to identify essential contributing factors in the conflict.
6. There should be an explicit representation of *ignorance* to allow for making *non-committing* statements ,i.e., to express the lack of conviction about the certainty of any available choices or events. Some measure of *ignorance*, should be available to guide the gathering of discriminant information.
7. The representation should allow also for *non truth-functional* assessments of uncertainty, i.e , one should be able to determine the uncertainty of compound statements not only on the basis of the individual uncertainty of each of its constituting *atomic* sentences, but knowing the uncertainty of a *compound* statement should lead to imposing certain restrictions on the uncertainty of the *atomic* statements as well.
8. The representation must be, or at least must appear natural to the user to enable him to describe *uncertain input* and to interpret *uncertain output*. The representation must also be natural to the user to enable him to elicit consistent weights representing the strenght of the implication of each rule.

Inference Level

9. The rules that combine the uncertainty estimates for the validity of *atomic* sentences, in order to obtain a uncertainty estimate for *compound* sentences should not be based on global assumptions of *evidence independence*.
10. The combination rules should not assume the *exhaustiveness* and the *mutual exclusiveness* of the elements of the subset of sentences that serve as hypotheses.
11. The combination rules should maintain the *closure* of the syntax and semantics of the representation of uncertainty.
12. Any function used to summarise and propagate uncertainty should have clear semantics. This is needed to maintain the semantic closure of the representation and to allow the control level to select the most appropriate combining rules.
13. The reasoning process should not collapse when inconsistemsy is violated thus, inflecting the whole system; violations of consistency should result in *local* disturbances, not *global*.

Control Level

14. There should be a clear distinction between a *conflict* in the information, i.e., violation of *consistency*, and *ignorance*.
15. There should be a second order measure of uncertainty. It is important to measure the uncertainty of the information, as well as the *uncertainty of the measure* itself.
16. Making pairwise comparisons of uncertainty should be feasible, since the induced *ordinal* or *cardinal ranking* is needed for performing any kind of decision-making activities.
17. The *traceability* of the aggregation and propagation of uncertainty through the reasoning process must be available to resolve conflicts or contradictions, to explain the support for conclusions and to perform meta-reasoning for control.

Now the following table summarizes the comparison among the approaches mentioned:

Representations	*Criteria Number*																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Bayesian	N	N	N	N	N	N	N	Y	N	N	Y	N	N	N	N	Y	N
Confirmation	N	N	Y	N	N	Y	N	N	N	Y	N	N	N	N	N	Y	N
Dempster-Shafer	Y	N	Y	Y	Y	Y	N	Y	N	N	Y	N	N	Y	Y	Y	N
Evidential Reasoning	Y	N	Y	Y	Y	Y	N	Y	N	N	Y	N	N	Y	Y	Y	N
Probability Bounds	Y	N	Y	Y	Y	Y	N	Y	Y	Y	Y	N	N	Y	Y	Y	N
Fuzzy Approaches	Y	N	Y	Y	Y	Y	N	Y	Y	Y	Y	N	N	Y	Y	Y	N
Evidence space	Y	Y	Y	N	Y	Y	N	Y	Y	Y	Y	N	N	Y	Y	Y	N
Symbolic approaches	N	Y	N	N	Y	N	N	Y	Y	Y	Y	Y	Y	N	N	N	Y

Counting the number of *No's* we can see that *Probability Bounds*, the *Fuzzy Approaches* and *Evidence Space* satisfy the same maximal number of criteria. On the other hand, the criteria which these approaches fail to satisfy, namely criteria 12 and 17, is due to the absence of a logic-like process for making inferences. Notice that the *Symbolic Approaches* do satisfy these two criteria. These observations open a possibility: What about a many-valued logic which uses *fuzzy* estimates of *lower probabilities* (or measures of *belief*) and *upper probabilities* (or implicit measures of *disbelief*), as *graded* versions of the classical *true* and *false*, and also employes the *combination* rules used by these two approaches for defining extended versions of the standard logical connectives. However, even such a many-valued logic will fail to satisfy criterion 7, if it is *truth-functional*. Furthermore, it will fail in the context of criterion 13 because it will exhibit one of the so-called *paradoxes of implication*, i.e., A *and not* A *implies* B, saying that anything can be inferred from a contradiction, this leading to *global* disturbances, instead of *local*.

2. Towards A Many-Valued Logic of Belief

The type of logic we have in mind is shaped after the circumstances under which a hypothetical reasoning machine obtains its input-data. In the first place, we consider the case when the input-data on which the inferences are based, come from multiple sources. The crucial point is, that all sources are considered to be equally trustworthy on the whole, but none of them is assumed to be the ultimate truth-teller. In this context contradictions threaten ,e.g., a source tells us to expect a sharp decrease in oil prices while another one, equally trustworthy, predicts that the prices will remain at their present level. If the logic guiding our reasoning machine was the classical two-valued logic, then it must give up altogether talking about anything to anybody or, equivalently, it must say everything to everybody because it is guided by the principle that anything can be inferred from a contradiction. This is in a striking contrast to the usual informal reasoning, in which the presence of contradictory assertions does not cause any great unpleasantness for the latter: the source of contradictions is usually quickly identified, the contradictory assertions are detected and dealt with in an appropriate way. Thus, the least we may require from our reasoning machine is, that it should be able to accept contradictory reports and also indicate the presence of contradictions when such are encountered otherwise, we have no way of knowing that its data-base contains contradictory information. A more radical approach, which we intend to take, will be to formulate operations on both inconsistent and consistent assertions and see what their effect on the consistent ones are.

In the second place, instead of being only *true* or *false* the input-data carry a credential regarding the belief in their validity. This credential takes the form of a *quantification* of the amount of belief in the validity of data in question. In this context the logic should allow for explicit representation of "truth-values" in terms of quantified assessments of the amount of belief hold with respect to the validity of propositions. Furthermore, since we want to be able to represent contradictory assessments of belief, a "truth-value" must allow for an integrated representation of two items of information: a report on how strongly the validity of a proposition is believed, and a report on how strongly its validity is disbelieved, i.e., how strongly the negation of this same proposition is believed. This will very much differentiate the logic guiding our reasoning machine from the classical two-valued logic with its binary *true/false* scheme for assessment of the validity of data. On the other hand, we have to provide for such logical operators which will be able to manipulate in some sensible way not only "truth-values", representing quantified belief and/or disbelief, but also the classical *true* and *false* assessments of validity meant to express firm belief and/or disbelief in the validity of propositions.

In the third place, we consider the case of *independent* as well as *dependent* assessments of belief in the validity of the input-data. In the first case, quantified assessments of belief and/or disbelief, provided by a source, are established independently from assessments supplied by other sources. When a source takes into account belief and/or disbelief assessments provided by other sources, in order to determine the amount of belief and/or disbelief in the validity of its own reports, then we have the case of *dependent* assessments of belief. This again will differentiate the logic guiding our reasoning machine from the classical logic, since when using the latter we can reason only about facts where the validity of each fact has been established independently from the validity of all other facts in the data-base.

2.1 Representing Uncertainty As Belief/Disbelief Pairs

The notion of a verbally defined *belief/disbelief pair* /**A**/ of an atomic proposition **A** is introduced in the following manner [1]:

$$/\mathbf{A}/ = [s(\mathbf{A}), p(\mathbf{A})]$$

where, s(**A**) is a linguistic estimate indicating how *likely* it is that **A** may turn out as *true*; p(**A**) is a linguistic estimate indicating how *likely* it is to fail to refute the validity of **A** and is determined from the knowledge about how *likely* it is that *not* **A**, denoted as ¬**A**, may turn out as *true*:

$$p(\mathbf{A}) = E_9 - s(\neg\mathbf{A})$$

Furthermore, a *set-up* is defined in terms of a table assigning one particular *belief/disbelief pair*, from a given set of *belief/disbelief pairs*, to each atomic formula. Such a table is in fact a mapping from atomic formulas into the set I, that is, a collection of verbally defined *belief/disbelief pairs* that are distributed in six belief-states as follows:

A *contradictory* (C) belief-state is present when there is a *meaningful* enough belief in the validity of **A** as well as in the validity of its negation ¬**A**. In terms of the *degrees of belief* of **A** and ¬**A** this would mean that the *likelihood* of **A** to turn out as *true* and this of ¬**A** turning *true* are both at least *meaningful*.

A belief-state *believed to a degree* (B), would mean that it seems *meaningful* to believe in the validity of **A** while it does not seem *meaningful* at all to believe its falsity (or the validity of ¬**A**).

A belief-state *rather believed than disbelieved* (RB) corresponds to the case when it is *meaningful* to believe in the validity of **A** while it also makes sense to hold a belief in its falsity, though a much lesser such.

A belief-state *disbelieved to a degree* (D) would mean that it does not seem *meaningful* at all to believe in the validity of **A** while it seems *meaningful* to assert its falsity (or the validity of ¬**A**).

A belief-state, namely *rather disbelieved than believed* corresponds to the case when it is *meaningful* to believe the falsity of **A** while it also makes sense to hold a belief in its validity, though a much lesser such.

An explicit representation of *ignorance*, i.e., the case when it is not *meaningful* to believe either the validity or the falsity of **A**, would mean that the *likelihood* of **A** to turn out as *true* as well as this of ¬**A** turning *true* are both of such a dimension so that very little is known to allow us to classify /**A**/ in one of the already discussed belief-states. We call this belief-state the *unknown* (U).

To be able to avoid the problems of the *granularity* of the representation, the semantics of each linguistic estimate E_i is provided by a *fuzzy number* N_i on the interval [0, 1]; N_i is characterized by a parametric membership-function $m_i(x)$, $x \in [0, 1]$ and this parametric representation is achieved by the 4-tuple $(a_i, b_i, \alpha_i, \beta_i)$. We use a verbal scale consisting of nine linguistic estimates for the *likelihood* of the validity of a proposition: *impossible*, *extremely unlikely*, *very low chance*, *small chance*, *it may*, *meaningful chance*, *most likely*, *extremely likely*, *certain*. Thus the *likelihood* of **A** to turn out as *true* is being used as a measure of our *degree of belief* in the validity of **A**, while the *likelihood* of ¬**A** to turn *true* is a measure of the *degree of disbelief* in the validity of **A** (or the *degree of belief* in the validity of ¬**A**). A very important feature of the linguistic estimates proposed is that each one of them ,say E_i , has its *mirror-image* E_{n+1-i} (n is the index of the *linguistic estimate* of maximum *likelihood*) about *it may*.

2.2 The Logical Lattice

The partial order $\preceq$ (less-or-equally believed to be true than) on *belief/disbelief pairs* is such that, for every two *belief/disbelief pairs* /**A**/ = [s(**A**), p(**A**)] and /**B**/ = [s(**B**), p(**B**)] we say that,

$$/\mathbf{A}/ \preceq /\mathbf{B}/ \text{ iff } s(\mathbf{A}) \leq s(\mathbf{B}) \text{ and } p(\mathbf{A}) \leq p(\mathbf{B})$$

Furthermore, a belief-state X is *above*, in the sense of $\preceq$, another belief-state Y (or Y is *below* X) if and only if for at least one any $/A/ \in X$ there is at least one $/B/ \in Y$ such that

$/B/ \prec /A/$, and there is no such $/A/ \in X$ and $/B/ \in Y$ so that $/A/ \preceq /B/$. We say also that X and Y can not be ordered in the sense of $\preceq$ if and only if for any $/A/ \in X$ and any $/B/ \in Y$ these can not be ordered with respect to $\preceq$. Here $/A/$ and $/B/$ stay for the *belief/disbelief pairs* of any proposition, atomic or compound. Then we showed that the belief-state *believed to a degree* is *above* any other belief-state; *rather believed than disbelieved* is *below believed to a degree*, but *above* any one of the remaining belief-states; *contradictory* and *unknown* can not be ordered, but are *below rather believed than disbelieved* and *above rather disbelieved than believed* and *disbelieved to a degree*; *rather disbelieved than believed* is *below* both *contradictory* and *unknown*, but *above disbelieved to a degree* which itself is at the bottom of all belief-states.

It was shown [1], that the elements of I form a lattice, called the *logical lattice*, under the partial order from above. Particular l.u.b. and g.l.b. operations were introduced and used to define *and* and *or* logical connectives, as well as *negation*. Furthermore we used these logical operations to induce a semantics for a language involving them in just the usual way: given an arbitrary *set-up*, a mapping from atomic formulas into the set of *belief/disbelief pairs*, we extended it to a mapping of all formulas into this set of *belief/disbelief pairs*. Finally, *entailment*, as introduced in [1], relied on this same *logical lattice* - given any formulas A and B that might be compounded by the use of the *and* and *or* and *negation* logical connectives, we said that A *entails* B whenever one of the following cases is present:

(I) The case when $/B/$ and $/A/$ belong to the same belief-state.

(II) The case when belief-state of $/B/$ is *above* (in the sense of $\preceq$) the belief-state of $/A/$.

Finally, we introduced a number of semantically valid, and taken together, semantically complete set of principles for the reasoning machine to use in making its inferences. These turned out to be exactly the so-called *tautological entailments* of *relevance logic*. Some important observations in connection with this result are the following:

The "truth-values" represented as *belief/disbelief pairs* are definitely not supposed to be used for determining which formulas count as the so-called *logical truths*, i.e., as tautologies. In fact no formula takes always a *belief/disbelief pair* belonging to the belief-state *believed to a degree* so, that property surely will not do as semantic account of *logical truth*.

Not derivable from our set of inference principles and not semantically valid, are the paradoxes of *implication*: $/A \wedge \neg A/ \rightarrow /B/$ and $/A/ \rightarrow /B \vee \neg B/$. The failure of the first of them means that just because we have that the validity of A is believed and that it is disbelieved we cannot conclude everything. Indeed we may know nothing about the belief in the validity of B, or just that it is not believed. The failure of the second paradox is equally evident: From the fact that there is belief in the validity of A , we can not conclude that we know something about the belief in the validity of B. For $/B \vee \neg B/$ to belong to the belief-state B is either $/B/$ to belong to B or $/B/$ to belong to D; and it may be neither of the two. These inferences are not wanted in a logic that is designed not to break down in the presence of *contradictions* so, their absence is justified.

The generality of our approach does not suffer at all because of the use of the particular scale L_2 which elements describe the different amounts of believe and/or disbelieve in the validity of propositions. Any scale, its elements being fuzzy or crisp numbers, or intervals, will suffuce as long as it is possible to compute its *closure* under each logical operator and its elements exhibit the *mirror-image* property.

3. Non Truth-Functional Aspects

3.1 The Information Lattice

Another type of partial order $\sqsubseteq$ (conveys less-than-or-equal amount of information) on *belief/disbelief pairs* can be defined as,

$$/A/\sqsubseteq/B/ \text{ iff } s(A) \leq s(B) \text{ and } p(A) \geq p(B)$$

so that the set I of all *belief/disbelief pairs is a partially ordered set with respect to* $\sqsubseteq$. Furthermore, in [2] we defined a specific g.l.b ($\sqcap$), and l.u.b.($\sqcup$) operators which were: *idempotent*, *commutative associative*, *distributive*, *monotone*, and *strictly increasing* in both arguments. As to the ordering between belief-states in the sense of $\sqsubseteq$ we say that a belief-state X is *above* another belief-state Y (or Y is *below* X), if and only if for at least one

$/A/ \in X$ there is at least one $/B/ \in Y$ such that $/B/ \sqsubseteq /A/$ and $/B/ \neq /A/$, and there is no such $/A/ \in X$ and no such and $/B/ \in Y$ so that $/A/ \sqsubseteq /B/$. We also say that X and Y can not be ordered if and only if any /B/ ∈ Y and any /A/ ∈ X can not be ordered with respect to $\sqsubseteq$. Then it can be shown that the belief-state *contradictory* is *above* any other belief-state; *rather believed than disbelieved* and *rather disbelieved than believed* are *below contradictory*, and can not be ordered; *believed to a degree* and *disbelieved to a degree* can not be ordered and are *below rather believed than disbelieved* and *rather disbelieved than believed* respectively; both of *believed to a degree* and *disbelieved to a degree* are *above unknown*, which in turn is at the bottom of all belief-states.

3.2 Representing Non Truth-Functional Beliefs

Suppose we are told that *Taxes will be raised or deficit will be high.* Let this complex formula $\mathcal{F}$ be assigned a *belief/disbelief pair* belonging to B, but no *belief/disbelief pairs* are assigned to the atomic formulas **A** = *Taxes will be raised* and **B** = *Deficit will be high.* In this case a single *set-up* can not represent the *epistemic state* in which we should be if told that $/\mathcal{F}/ \in B$, but are told nothing about /**A**/ and /**B**/. For any single *set-up* in which **A** *or* **B** is assigned a *belief/disbelief pair* belonging to B is, a *set-up* in which either **A** or **B** is assigned a *belief/disbelief pair* belonging to B and therefore has too much information. Such a *set-up* would cause an affirmative answer either to the question *Will taxes be raised?* or to the question *Will deficit be high?* We should not be able to answer either questions since we have been told only that $/\mathcal{F}/ \in B$, but not that either /**A**/ or /**B**/ belong to the same belief-state.

The solution to this problem is to use a *collection of set-ups* to represent this *epistemic state.* When we are told that $/\mathcal{F}/ \in B$, we will represent this information by building two *set-ups*: one in which /**A**/ ∈ B and /**B**/ ∈ U , and the other in which /**A**/ ∈ U and /**B**/ ∈ B. Later, when asked *Will taxes be raised?* we will answer that we do not know, i.e., a *belief/disbelief pair* belonging to U will be assigned to **A** since this atomic formula does not have a *belief/disbelief pair* belonging to B in every *set-up.* We will also give the same answer to the question *Will deficit be high?.* But when asked *Will taxes be raised or deficit will be high?* we will produce an affirmative answer, i.e., $/\mathcal{F}/$ will belong to B, since in both *set-ups* $/\mathcal{F}/ \in B$.

Let us then define the *epistemic state*, E, of the reasoning machine as a *collection of set-ups.* In the context of E, the *belief/disbelief pair* of the complex formula $\mathcal{F}$, is denoted by $E(/\mathcal{F}/)$, and is determined by taking the *meet* of all *belief/disbelief pairs* that can be assigned to $\mathcal{F}$ in the separate *set-ups* of E:

$$E(/\mathcal{F}/) = \sqcap\{\ s(/\mathcal{F}/)\colon s \in E\ \}$$

Why this particular definition for the value of $\mathcal{F}$ in E ? It was already noted that *set-ups*, when considered separately, tend to convey more information than there actually is about a formula. This in terms of the *information lattice*, would mean that:

$$E(/\mathcal{F}/) \sqsubseteq s\ (/\mathcal{F}/) \quad \text{for each } s \in E$$

So, it seems natural to define $E(/\mathcal{F}/)$ as maximal while retaining the relationship from above. In other words, the *belief/disbelief pair* to be assigned to the complex formula $\mathcal{F}$ in the *epistemic state* E, that is $E(/\mathcal{F}/)$, is to be defined as the g.l.b of all $s(/\mathcal{F}/)$ for $s \in E$.

To sum up, we will stress upon the following two cases: First, let the *epistemic state* E be given as a *collection of set-ups* { s_i }, where *set-up* consists of a number of atomic formulas $\mathbf{A}_j$ with s_i (/$\mathbf{A}_j$ /) being their *belief/disbelief pairs* in the *set-up* s_i . Now one is ready to answer questions concerning the *belief/disbelief pair* to be assigned to an arbitrary complex formula $\mathcal{F}$, a logical combination of particular $\mathbf{A}_j$'s, in the *epistemic state* E. Using the logical operators we can determine s_i $(/\mathcal{F}/)$ for each i. Then to compute the *belief/disbelief pair* of $\mathcal{F}$ in E we have to find the g.l.b. of all s_i $(/\mathcal{F}/)$ by using $\sqcap$.

Secondly, let a complex formula $\mathcal{F}$ together with $/\mathcal{F}/$ be supplied to us. The question is: how do we represent this knowledge so, that we can not only answer questions regarding the belief in the validity of $\mathcal{F}$, but also questions concerning the belief in the validity of the atomic formulas that constitute $\mathcal{F}$? The solution is as follows: Represent $/\mathcal{F}/$ as an *epistemic state* E, i.e , a *collection of set-ups* s_i , each *set-up* consisting of the atomic formulas $\mathbf{A}_j$ that are in $\mathcal{F}$.

Then to each $\mathbf{A}_j$ in a *set-up* s_i assign s_i $(/\mathbf{A}_j/)$ such that the g.l.b. of all s_i $(/\mathbf{A}_j/)$ belongs to U, while the g.l.b of all s_i $(/\mathcal{F}/)$ is equal to the *belief/disbelief pair* with which $\mathcal{F}$ was initially supplied.

It is to be stressed here that there are certain cases when once a complex formula $\mathcal{F}$ is assigned a *belief/disbelief pair* $/\mathcal{F}/$, the *belief/disbelief pairs* of its constituents become uniquely determined and they are far from belonging to U. In [2] we consider all cases when a two-place *conjunction* and *disjunction* which constituents are atomic propositions, is assigned a *belief/disbelief pair* belonging to a particular belief-state. Then we find out for which one of the six belief-states it is possible to construct a representation in terms of a collection of *set-ups*, so that the g.l.b. of the *set-ups* involved provides U for each of the constituent atomic formulas while keeping intact the *belief/disbelief pair* that was already assigned to the *conjunction* or *disjunction* in question. Once this is established for two-place *conjunctions* and *disjunctions* it can be readily used for representing the *epistemic state* of any complex formula. It is also proposed how to represent non truth-functional beliefs in the case of formulas with *exclusive or*.

3.3 Formulas As Inputs

Let us first introduce the notion of a partial ordering $\subseteq$ (contains less-or-equal amount of information) between *set-ups*:

> $s \subseteq s^*$ iff for at least one atomic formula $\mathbf{A}_i$ either $s(/\mathbf{A}_i/) \sqsubseteq s^*(/\mathbf{A}_i/)$, or they can not be ordered, but $s^*(/\mathbf{A}_i/)$ belongs to a belief-state that is *above* the belief-state of $s(/\mathbf{A}_i/)$, while all other $s(/\mathbf{A}_k/)$ and $s^*(/\mathbf{A}_k/)$ $(k \neq i)$ belong to the same belief-state and can not be ordered.

It can be easily shown that the *set-ups* constitute a lattice with respect to $\subseteq$, with $\sqcup$ and $\sqcap$ being a l.u.b. and g.l.b. operations. Moving to *epistemic states* we can also talk about one *epistemic state* containing less-or-equal amount of information than another one. Formally speaking,

$$E \subseteq E^* \quad \text{iff} \quad \forall\, s^* \in E^*, \exists\, s \in E \text{ such that } s \subseteq s^*$$

It should be stressed here that the ordering $\subseteq$ on *epistemic states* that are *collections of set-ups* does not yield a lattice because *anti-symmetry* fails.

Consider now atomic formulas only and try to answer the following question: What to do with the present *epistemic state* E when an atomic formula $\mathbf{A}$, already represented in E by a $/\mathbf{A}/$, is at some new point in time affirmed or denied ? We say that $\mathbf{A}$ is affirmed if $/\mathbf{A}/ \in B$ or RB, and denied if $/\mathbf{A}/ \in D$ or RD. Furthermore, we represent the affirmation of $\mathbf{A}$ by generating a *set-up* $\mathbf{A}_t$ in which $/\mathbf{A}/ \in B$ or RB and all other atomic formulas belong U. In a similar way, denying $\mathbf{A}$ generates a *set-up* $\mathbf{A}_f$ in which $/\mathbf{A}/ \in D$ or RD, and all other atomic formulas belong to U. Let us also associate with $\mathbf{A}$ two functions: a function $f^+(E)$ representing the transformation of the present *epistemic state* E into a new *epistemic state* E^* when $\mathbf{A}$ is affirmed, and $f^-(E)$ when $\mathbf{A}$ is denied. To define what we want E^* to be, we assume that we always use an input to increase the information about the beliefs in the validity of the formulas, or at least we never use input to throw away information. In terms of the partial ordering between *epistemic states* this condition can be expressed as,

$$E \sqsubseteq f^+(E)$$

Secondly, we assume that $f^+(E)$ should say no less than the affirmation of $\mathbf{A}$, this being expressed as:

$$\mathbf{A}_t \sqsubseteq f^+(E)$$

Lastly, we want $f^+(E)$ to be the minimum mutilation of E which causes $/\mathbf{A}/$ to belong at least to either B or RB, namely we want the least of those *epistemic states* satisfying the above two requirements. This means that we should define $f^+(E)$ and $f^-(E)$ in the following way:

$$f^{+}(E) = \{ s \sqcup \mathbf{A}_t : s \in E \} \qquad f^{-}(E) = \{ s \sqcup \mathbf{A}_f : s \in E \}$$

Thus, when $/\mathbf{A}/ \in B$, we make a run through each *set-up* of E and perform $\sqcup$: this will make $/\mathbf{A}/ \in B$ if $/\mathbf{A}/$ belonged to U before, it will leave $/\mathbf{A}/$ the same if it already belonged to B or C, and will make it belong to C if it already belonged to D. The case of a complex formula being an input and the corresponding transformation of the present *epistemic state* into a new one, are considered in a similar way (for details see [2]).

3.4 Further Developments: Inference Rules As Input

We have in the previous section found the way of giving meaning to a formula as an input: the reasoning machine is to improve its present *epistemic state* in the minimum possible way so as to make the formula belong to B or RB. So, we can look forward to treating $A \rightarrow B$ as signifying a mapping from one *epistemic state* to another such that $A \rightarrow B$ belongs to B or RB in the latter. To be able to pursue this line we need the answers to the following three questions:

1. What is it for $A \rightarrow B$ to be valid in a *set-up*;
2. What is it for $A \rightarrow B$ to be valid in an *epistemic state*;
3. How to define $A \rightarrow B$ as a mapping from E to E^* so that it represents the minimum mutilation of E yielding the validity of $A \rightarrow B$ in E^*.

We have given answers [3] to all three questions for rules of inference that are different from the *tautological entailments* of *relevance logic*. These are a special class of *IF A THEN B* rules to be used in cases when:

1. it is *firmly* believed that belief and/or disbeliefin the validity of B follows from belief and/or disbelief in the validity of A;
2. it is believed, but only to a degree, that belief and/or disbelief in the validity of B follows from belief and/or disbelief in the validity of A.

Furthermore we consider also inference rules that are augmented with the so-called *exception* condition, that is rules of the form, *IF A THEN B UNLESS C* [4]. Here the *IF - THEN* part of the rule expresses the major relationship between A and B, i.e., it is believed (firmly or to a degree) that belief and/or disbelief in the validity of B follows from belief and/or disbelief in the validity of A. Then the *UNLESS* part acts as a switch that transforms the *belief/disbelief pair* of B from one expressing belief in its validity to one indicating belief in the validity of $\neg B$ (or disbelief in the validity of B), whenever there is a meaningful enough belief in the validity of the exception condition C. On the other hand, whenever there is a meaningful enough belief in the validity of $\neg C$ (or a meaningful enough disbelief in the validity of C), or there is an absence of both belief and disbelief in the validity of C, then the belief and/or disbelief in the validity of B depends only upon the corresponding belief and/or disbelief in the validity of A.

The results obtained in [3] and [4] can be easily extended so as to provide for the treatment of *tautological entailments* as mappings between *epistemic states*.

References

1. D. Driankov, *A many-valued logic for belief-intervals: The logical lattice*, in Preprints of the 2nd IFSA Congress, July 20-25, 1987, Tokyo, Japan, pp.426-430.
2. D. Driankov, *Amany-valued logic for belief/disbelief pairs*, in Z.W. ras and M. Zemankowa(eds.): *Methodologies for Intelligent Systems, North-Holland, 1987, pp.25-33.*
3. *D. Driankov, Towards a many-valued logic of belief: Detachment operators*, unpublished paper, 1987.
4. D. Driankov, *Towards a many-valued logic of belief: Detachment operators with an exception condition*, unpublished paper, 1987.

11.

ADP Administrative Data Processing

Göran Goldkuhl

11.1 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.

11.2 Research activities.

Post-graduate and research activities related to the ADP undergraduate programs cover the following areas:

- 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.

11.3 Personnel during the year:

Göran Goldkuhl, PhD, senior lecturer
Lise-Lotte Raunio, MSc, lecturer, director of undergraduate studies
Carina Björkman, secretary
Carita Lilja, secretary
Siv Söderlund, secretary

Pia Arendell, BSc, lecturer
Johan Eltes, undergraduate assistant
Hans-Olov Ganning, undergraduate assistant
Hans Holmgren, MScEng, lecturer
Rolf Nilsson, BSc, lecturer
Anders Knutsson, undergraduate assistant
Dan Michaeli, undergraduate assistant
Torbjörn Näslund, postgraduate assistant
Annie Röstlinger, BSc , lecturer
Eva-Chris Svensson, MSc, lecturer
Roger Zollner, lecturer
Elisabeth Zsiga, undergraduate assistant
Per Övernäs, BSc, lecturer

Appendix A

Administrative organization

The Department of Computer and Information Science (IDA) at Linköping University covers three teaching subjects (computer science, telecommunication and computer systems, and administrative data processing). The Department was formed in 1983, bringing together groups previously in the Mathematics and the Electrical Engineering departments. A considerable flexibility was allowed when the internal organization and routines were to be decided. The basic idea was to build research within the department upon vital, autonomous, and cooperating research groups, each with a distinct leader and about five to ten more teachers, researchers, and employed graduate students. From the beginning there were four such groups or laboratories. Today there are ten.

The lab leader is responsible for supervision and guidance of the work in his group, and also for writing grant proposals and reports to funders. Each lab also takes responsibility for maintaining competence in its area of research and some related areas, and to make it available to the rest of IDA in graduate courses and seminars, as well as in the undergraduate course program. The set of labs is designed to provide a sufficiently wide basis for a vital computer science department and also to give the necessary spectrum required for the undergraduate courses given by the department. At the same time it is important that research is sufficiently focused and that a group can achieve critical size in its area of specialization.

Important and general issues regarding research or undergraduate studies are treated by the research committee or the committee for undergraduate education respectively. The research committee, headed by Erik Sandewall and with Lillemor Wallgren as secretary, handles research activities and graduate education. This committee suggests the annual budget for each lab, based on grant situation, and can also modify the lab structure by merging, splitting, creating, or deleting labs and appointing lab leaders. Admission of doctorate students has to be confirmed by the research committee. The committee also discusses and takes appropriate actions on research and equipment strategy in general, and coordinates the lab-based activities. The philosophy, however, is to support and assist rather than to control and supervise the labs.

The Committee for Undergraduate Education, headed by Anders Haraldsson with Carina Björkman as secretary, is responsible for the organization of undergraduate courses and continuing education for industry. Most of the

Department of Computer and Information Science

Organization:

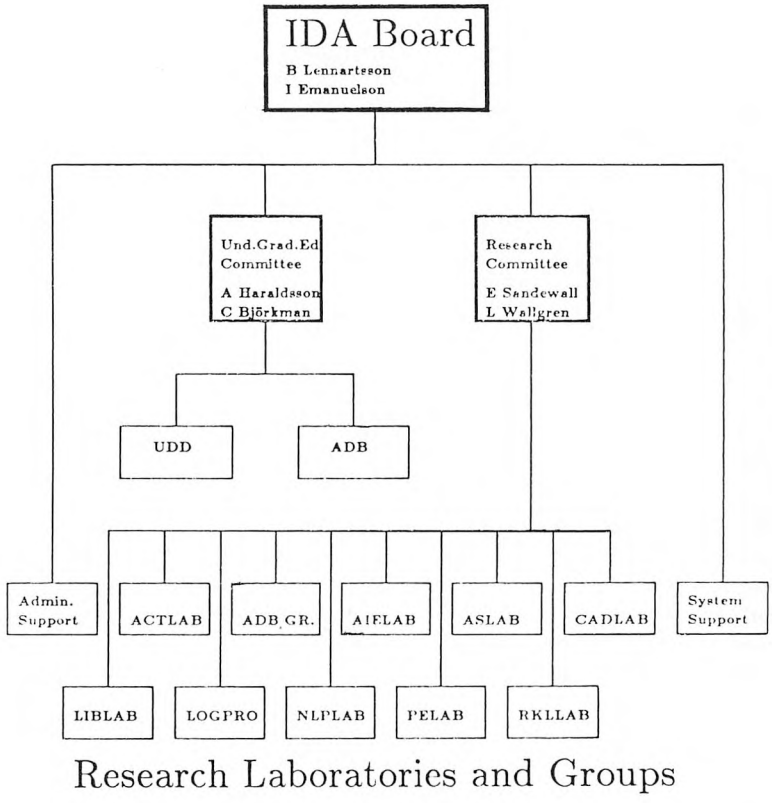


Figure A.1. *Administrative organization of the department.*

teachers and lecturers are also members of the research labs and the decision about teaching load for each individual, in terms of percentage, is taken annually in conjunction with the budget negotiation process. The executive responsibility for undergraduate studies are taken by the *directors of studies*, with Anders Haraldsson responsible for the study programs within the School of Engineering and Lise-Lotte Raunio for those in the School of Arts and Science.

Formally all significant administrative decisions, such as the annual budget are taken by the Department Board, which is prescribed to exist. The board is chaired by Bengt Lennartsson, with Inger Emanuelson as secretary. Annually the board delegates to the two committees all issues about research and graduate studies, and about undergraduate education, respectively. The board also handles items related to both committees, normally by approving their coordinated proposals.

Running economy and personnel issues are handled by Inger Emanuelson, who is also the leader for the group providing administrative services. The system support group under Anders Aleryd and Mats S Andersson is responsible for computer systems and services, as well as for other kinds of equipment at the department. Computer resources and other equipment are normally not reserved for a specific group or project, but shared as far as possible and supported at the department level. This allows a good economy for support costs and effective use of the facilities, although projects needing exclusive access to a particular equipment of course can be granted that right for a specific period of time.

The department budget for the fiscal year 1987/88 balances at 29.6 MSEK. (One MSEK is at present approximately 0.15 USD.) Of this sum, the resources for undergraduate education supplied by the university amount to 13.2 MSEK, and corresponding resources for research and graduate education are 4.0 MSEK. The research activities are thus heavily dependent on external sources, where the Swedish Board for Technical Development, STU, is the main contributor (87/88: 6.8 MSEK). Additional funds are provided by the Delegation for Technical and Scientific Information Supply, DFI, (87/88: 1.0 MSEK) and the Natural Science Research Council, NFR, (87/88: 0.1 MSEK). Occasional sources, such as contributions from companies participating in the knowledge transfer programme and shorter projects supported by e.g. *Sveriges Mekanförbund*, are in the order of 2.6 MSEK. Commissioned education programmes for industry are budgeted at about 2 MSEK 1987/88. Costs for office space and investment in equipment are not included in the above figures.

Department leadership:

Bengt Lennartsson, department chairman

Erik Sandewall, research committee chairman
Anders Haraldsson, undergraduate education committee chairman

Administrative office:

Inger Emanuelson, administrative manager
Carina Björkman, general educational secretary
Lillemor Wallgren, general research secretary

Britt-Marie Ahlenbäck, secretary
Anne-Marie Jacobson, secretary
Barbara Ekman, secretary

Carita Lilja, secretary
Lisbeth Linge, secretary
Gunilla Lingenhult, secretary
Bodil Mattsson Kihlström, secretary
Siv Söderlund, secretary
Lena Wigh, office assistant

Technical services:

Anders Aleryd, managing engineer
Mats S Andersson, senior research engineer

Leif Finmo, research engineer
Dimitrios Fotiadis, research engineer
Ulf Dahlén, research engineer
Arne Fäldt, senior research engineer
Björn Nilsson, senior research engineer
Peter J. Nilsson, research engineer
Katarina Sunnerud, research engineer

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.

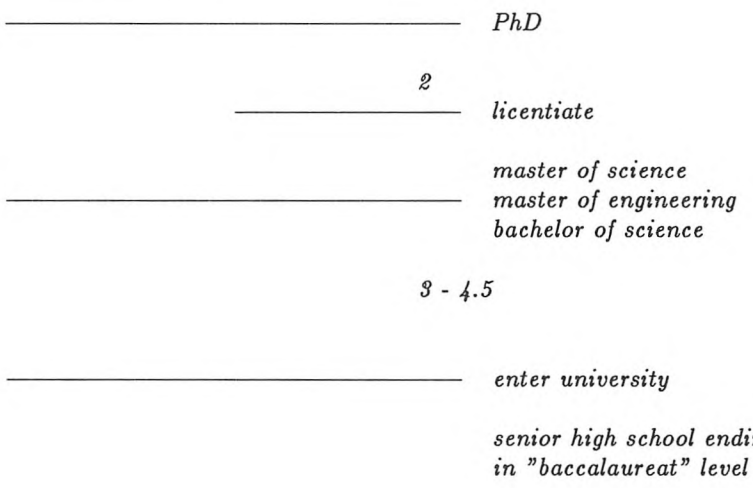


Fig B.1. *Levels of degrees*

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 C.

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 organized 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. Thesis projects are always done within or in association with the laboratories or research groups. Admission to graduate studies is nominally free for students with the appropriate qualifications, but it is not realistic nor

recommended to start studies without being admitted as a member of one of the research groups.

Faculty presently engaged in graduate study program.

Ahrenberg, Lars, BA. PhD, Uppsala 1987. Assistant professor (*högskolelektor*), computational linguistics. Group leader, NLPLAB. Previous affiliation Uppsala and Göteborg.

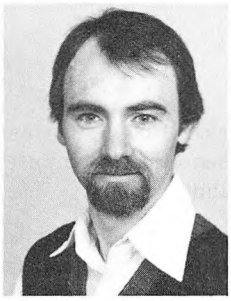
Natural language processing, computational linguistics, user interfaces.

Douglas Busch, PhD, Rockefeller 1973. Assistant professor (*högskolelektor*) of logic and theoretical computer science. Previous affiliation Mcquarie University, Sydney, Australia.

Application of theories from formal logic to problems in theoretical computer science and artificial intelligence; algebraic specification theory, intuitionistic type theory non-monotonic logic; philosophical questions in artificial intelligence.

Peter Fritzson, PhD, Linköping 1984. Assistant professor (*högskolelektor*), computer science. Thesis supervision in PELAB.

Tool generation, incremental tools, programming environments.



Göran Goldkuhl, PhD, Stockholm 1980. Associate professor (*docent, högskolelektor*), administrative data processing. Group leader in ADP research. Previous affiliation Göteborg.

Information requirement analysis, behavioral aspects of information systems, research methodologies, information systems and quality of working life.

Anders Haraldsson, PhD, Linköping 1977. Associate professor (*högskolelektor*), computer science. Director of undergraduate studies in computer science. Previous affiliation Uppsala. Thesis supervision in PELAB.

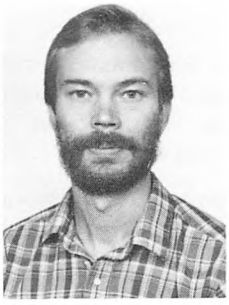
Programming languages and systems, programming methodology, program manipulation.

Roland Hjerppe. Researcher. Group leader, LIBLAB. Previous affiliation KTH, DFI and expert mission Tanzania.

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. Acting professor of knowledge-based systems. Group leader, ASLAB. Previous affiliation Uppsala.

Expert systems and artificial intelligence applications, database technology, human-computer interaction.



Rolf Karlsson, PhD, Waterloo 1984. Assistant professor (*högskolelektor*), theoretical computer science. Previous affiliation Lund.

Data structures, algorithm analysis, computational complexity, computational geometry.

Krzysztof Kuchcinski, PhD, Gdansk 1984. Assistant professor (*gästforskare*), computer science. On leave from Institute of Computer Science, Politechnika Gdanska. Group leader, CADLAB.

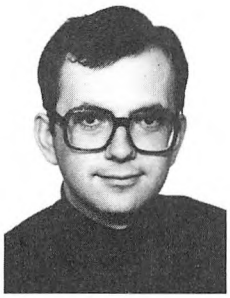
Computer architecture, CAD, real-time operating systems, system testing.

Harold W. Lawson Jr., PhD, Stockholm 1983. Professor of telecommunication and computer systems. Several previous affiliations, also in industry. On leave for Swedish International University 1987-88.

Computer architecture, VLSI, Computer-aided design, methodology of computer-related education and training.

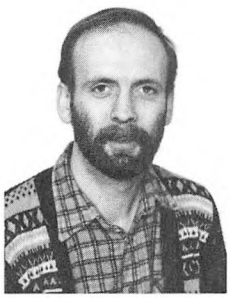
Bengt Lennartsson, PhD, Göteborg 1974. Associate professor (*högskolelektor*), software engineering. Previous affiliation Luleå. Group leader, PELAB.

Programming environments, real-time applications, distributed systems.



Christos Levcopoulos, PhD, Linköping 1987. Assistant professor (*forskarassistent*), theoretical computer science.

Computational geometry, analysis of algorithms, data structures.

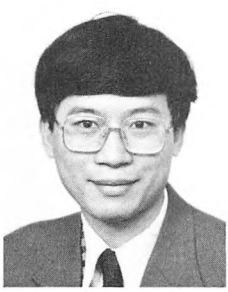


Andrzej Lingas, PhD, Linköping 1983. Associate professor (*docent, högskolelektor*), theoretical computer science. Previous affiliation Warszawa and MIT. Group leader in ACTLAB.

Complexity theory, analysis of algorithms, geometric complexity, graph algorithms, logic programming, VLSI theory.

Jan Maluszynski, PhD, Warszawa 1973. Professor of programming theory. Several previous affiliations. Group leader in theoretical computer science.

Logic programming, software specification methods.



Zebo Peng, PhD, Linköping 1987. Assistant professor (*högskolelektor*), computer architecture.

Automated synthesis of digital systems, formal description of hardware, VLSI, computer-aided design, computer architecture.

Erik Sandewall, PhD, Uppsala 1969. Professor of computer science. Group leader in RKLLAB. Several previous affiliations.

Representation of knowledge with logic, theory of information management systems, office information systems, autonomous expert systems.

Erik Tengvald, PhD, Linköping 1984. Assistant professor (*högskolelektor*), computer science. Group leader, AIELAB.

Artificial intelligence, knowledge representation, planning and problem solving, expert systems.

Graduate Study Course Program 1986-87

Basic and Occasional Graduate Courses:

Communicating Sequential Processes and Calculus of Communicating Systems (Johan Fagerström, Jan Maluszynski).

Non-Monotonic Reasoning: Theories, Systems, and Applications (Michael Reinfrank)

Informationssystem i Organisationer - seminarieserie (Göran Goldkuhl)

Principles of Database Systems (Sture Hägglund, Bo Sundgren)

Attribute Grammars and Logic Programs (Jan Maluszynski)

Analysis and Complexity of Parallel Algorithms (Andrzej Lingas)

Computational Geometry (Christos Levcopoulos, Andrzej Lingas)

Algorithm Analysis and Design (Andrzej Lingas)

Program Transformation (Anders Haraldsson, Jan Maluszynski)

Constructive Mathematics and Specification Languages (Douglas R Busch)

Semantiska Modeller för Naturligt Språk - Seminarieserie (Lars Ahrenberg)

Lower Bound Techniques (Rolf Karlsson)

Amortized Computational Complexity (Rolf Karlsson)

Machine Learning (Jalal Maleki)

Research-Related Courses and Seminars:

Kunskapsomgivningar på parallella maskiner (Erik Tengvald)

Informationssystem i organisationer - Seminarieserie (Göran Goldkuhl)

AI and Software Engineering (Sture Hägglund)

Statistiska informationssystem (Bo Sundgren)

HYPERCATalog-projektet (Roland Hjerppe)

Kunskapsorganisation - teknik och metoder (Roland Hjerppe)

Logikprogrammering - seminarieserie (Jan Maluszynski)

XEROX Development Environment - studiecirkel (Bengt Lennartsson)

Smalltalk -80 - studiecirkel (Lars Strömberg)

Temporal logic - studiecirkel (Patrick Doherty, Dimiter Driankov)

Special Courses for the Knowledge Transfer Program

Introduction to Epitool (Roland Rehmnert)

Issues in AI and Expert Systems (Video lectures, supplemented by seminars.)

Knowledge engineer training program, fall 1986:

Introduction to AI and expert systems (Arne Jönsson, Sture Hägglund)

Discrete mathematics (Karl-Johan Bäckström, dept. of math.)

Mathematical Logic (Erik Sandewall)

Knowledge engineer training program, spring 1987:

AI programming systems (Anders Haraldsson et al.)

AI - cognitive processes (Arne Jönsson)

AI - knowledge representation (Douglas Busch)

Expert systems (Sture Hägglund)

Graduate Study Course Program 1987-88

Basic and Occasional Graduate Courses:

Lower Bound Techniques (Rolf Karlsson)

Analysis and Complexity of Parallel Algorithms (Andrzej Lingas)

Algorithm Analysis and Complexity Theory (Andrzej Lingas)

Computational Geometry (Christos Levcopoulos, Andrzej Lingas)

Office Information Systems - OIS (Roland Hjerppe)

Advanced Computer Architectures (Krzysztof Kuchcinski, Mikael Patel)

Negation in Logic Programming (Maurizio Martelli, Jan Maluszynski)

Logic Survey (Douglas Busch)

Advanced Course in Compiler Construction, especially Incremental Compilation (Peter Fritzson and invited lectures).

Reason Maintenance Systems (Michael Reinfrank)

Qualitative Reasoning (Peter Struss)

Introduction to Petri Nets (Krzysztof Kuchinski, Jan Maluszynski)

Human-Computer Interaction (Lars Ahrenberg, Sture Hägglund, Arja Vainio-Larsson)

Research-Related Courses and Seminars:

Förändringsanalys (Göran Goldkuhl)

Systemvärdering - seminarieserie (Göran Goldkuhl)

Systemutvecklingsmetoder - en jämförande analys (Göran Goldkuhl, Birger Rapp, IPE)

Kunskapsomgivningar på parallella maskiner (Erik Tengvald)

Conceptual Structures and Knowledge Base Management Systems (Sture Hägglund)

HYPERCA Talog-projektet (Roland Hjerppe)

Logikprogrammering - seminarieserie (Jan Maluszynski)

Planstyrda system (Erik Sandewall)

Software Reliability (Bo Bergman, IKP)

A Selection of Seminars 1987

General seminars spring 1987

9/1 Peter Struss, Siemens AG, Muenchen. Problems on Qualitative Reasoning.

9/1 Peter Struss, Siemens AG, Muenchen. Representing Structure and Function in ATMS-Based Problem Solvers.

22/1 Gerd Brewka, GMD, Bonn. BABYLON, a hybrid expert system development tool.

22/1 Gerd Brewka. Circumscription and the Semantics of Frames (replacing his talk on a non-monotonic logic theorem prover).

3/2 Ferenc Belik, Lunds Universitet. En graf-modell och dess applikation till distribuerade resursallokeringssystem.

23/2 Hartmut Freitag, SIEMENS AG, Muenchen. An extension of the RETE-pattern matching algorithm to non-monotonic rules and dependency-networks.

25/2 David McLeman, SEQUENT EUROPE, och Jonny Svensson, Macrotek. SEQUENT datorerna och parallel programming.

3/3 Staffan Truve, Chalmers Tekniska Högskola. Towards a specification language for physical objects.

6/3 Michael Reinfrank, IDA. Problems in non-monotonic reasoning.

10/3 Bonnie Lynn Webber, University of Pennsylvania. Questions, Answers, Responses: Multi-functional Interactions.

12/3 Jerry Hobbs, Artificial Intelligence Center. Local pragmatics and common sense knowledge.

27/3 Peter Gärdenfors, Lund. Modeller för kunskapens dynamik.

1/4 Torbjörn Näslund, IDA. An experimental implementation of a compiler for two-level grammars.

8/4 Kurt Koonolige, SRI International och CSLI. On the relation between default Theories and Autoepistemic Logic.

18/5 Dag Westerståhl, filosofiska institutionen i Göteborg. Kvantifiering i naturligt språk.

19/5 Kurt Nörmark, Aalborgs Universitetscenter. MUIR - A Language Development Environment.

22/5 Masataka Sassa. Attributed Grammars and the RIE System.

25/5 Dino Pedreschi, Pisa. A Functional Meta Level for Logic Programming.

26/5 Allan Brown, General Electric. Truth Maintenance.

26/5 Fosca Gianotti, Pisa. Incremental Constraint Solving with Logic Programming.

26/5 Carl Gustaf Jansson, Inst för ADB, Stockholms Universitet. Taxonomic representation.

26/5 Mike Kamrad. Honeywell S&RC. Distributed ADA.

2/6 Hilding Elmqvist, Satt Control, Lund. Anamation and Information Zooming in a System for Monitoring Industrial Process Control.

2/6 C Tully. A presentation of the information systems Factory project.

16/6 Mike Weintraub, Ohio State University. Survey of Knowledge Systems Research at Ohio State University.

General seminars fall 1987

18/8 Jörg Sack, Carleton University. A motion problem in the plane: separability.

21/9 Lars Hallnäs, SICS, Stockholm. Generalized horn clauses.

28/10 Robert A Greenes, Boston. Hypermedia applications in medical education and clinics.

3/11 Bo Bergman, Bengt Lennartsson. Diskussionseminarium om programvarutillförlitlighet.

17/11 H.N. Djidjev, Bulgarian Academy of Science. VLSI algorithms.

20/11 Dag Prawitz, Universitetet i Stockholm. Naturliga och Normala Bevis.

25/11 Jörgen Fischer Nilsson, University of Denmark. Translating Higher-order Prolog into Prolog.

25/11 Jörgen Fischer Nilsson, University of Denmark. W-order Knowledge Bases.

27/11 Gunn Johansson, Psykologiska institutionen, Universitetet i Stockholm. Abstraktisering, överbelastning, underbelastning och andra mänskliga begränsningar.

27/11 Kevin Poulter, GEC Research, UK. The Knowledge-Based Programmer's Assistant.

29/11 Joseph Wiezenbaum, MIT. On the Status of Knowledge in the Information Society.

4/12 Peter Marwedel, University of Kiel. Digital Hardware Design Automation

17/12 Jan Komorowski, Harvard University. Exploration of Medical Knowledge in a Semantic Network.

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 in School of Engineering, Linköping University. These study programs, and number of students accepted annually, are:

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 study programs run over 4 - 4.5 years and lead to a Master of Engineering or (for the C-program) a Master of Science degree.

There are also single-subject courses given as part-time and evening courses, and external courses given directly to companies and organizations. A program for "continuing education" in computer science has also started. This program has been developed by IDA in cooperation with Oktogonen, a Swedish engineering industry group. There are also programs in artificial intelligence and expert systems.

Courses. During 87/88 IDA will give a total of approx 75 different courses.In the engineering study programs IDA gives 51 courses with a total of 3300 students, 10 single-subject courses, and about 14 external courses for industry with about 500

participants. Due to the reorganization of the engineering programs to run over 4.5 years, a number of courses are postponed to the next year. All engineering programs have at least one introductory course in computer science and programming.

In the C- and D-programs and in the variants towards computer science in the M- and Y-programs (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-programs include two software projects. One done individually during the first year and one in a group during the third. In the projects both oral presentations and written reports are required.

In the C-program a number of human-oriented courses are given:

- linguistics, introductory course
- computational linguistics
- psychology, introductory course
- psychology of communication
- interactive systems

There are 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 and supporting about 60 terminals.

There are two UNIX computers (one PDP-11/70 and one Gould PN6000) for teaching purposes with totally 25 terminals, two PC laboratories with

MacIntoshes and Ericsson PC's, and one laboratory with eight Xerox LISP machines and one with eight SUN workstations.

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 done by full or half time employed lecturers, by other persons with research appointment, by graduate students having teaching assistantships, and by the students themselves as part-time course assistants.

During 87/88 the staff consists of

6 full time and 1 half time senior lecturers (associate professors)
7 full time and 2 half time lecturers (assistant professors)
9 other persons, professors and research assistants
about 40 postgraduate students with 25% - 50% teaching assistantships
c. 5 teachers from other subjects and from industry
c. 35 part-time course assistants

Personnel.

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

The following persons from IDA are teaching one or more courses:

Lars Ahrenberg, PhD
Pia Arendell, BSc
Rober Bilos, MSc
Douglas Busch, PhD
Nils Dahlbäck, B.A.
Patrick Doherty, BSc
Johan Fagerström, MSc
Björn Fjellborg, MSc
Christian Gnosspelius
Anders Haraldsson, PhD
Sture Hägglund, PhD
Arne Jönsson, MSc
Rolf Karlsson, PhD
Bengt Karlstrand, MSc
Christian Krysander, MSc
Tony Larsson, MSc
Bengt Lennartsson, PhD
Fredrik Lindström, MSc
Jalal Maleki, MSc
Jan Maluszynski, PhD

Magnus Merkel, B.A.
Rolf Nilsson, BSc
Kerstin Olsson, MSc
Tommy Olsson, MSc
Mikael Patel, MSc
Zebo Peng, MSc
Ola Petersson, BSc
Ivan Rankin, BSc
Roland Rehmnert, MSc
Erik Sandewall, PhD
Nahid Shahmehri, MSc
Ola Strömfors, MSc
Katarina Sunnerud, MSc
Eva-Chris Svensson, BSc
Olle Willen, BSc
Mats Wiren, MSc

Listing of Undergraduate Course Program 1987/88

Course (in Swedish)	Teacher
Databaser (D3, I4)	Christian Krysander
Databaser (C3, Y3, Y4, Md4)	Christian Krysander
Programmeringsspråk (C4, D4)	Tommy Olsson
Orientering datateknik och datorutrustning (C1, D1)	Christian Gnosspelius
Programmering i Ada(C4, D4)	Tommy Olsson
Programmeringsmiljöer (C4, D4)	Bengt Lennartsson
Systemutveckling, teori och tillämpning (C4, D4)	Pia Arendell
AI-programmering (C4)	Jalal Maleki
Logik, grundkurs (C1, D4)	Erik Sandewall
Psykologi grundkurs (C2)	Nils Dahlbäck
Databehandling av naturligt språk (C4)	Mats Wiren
Operativsystem (D3tk, D4, Y4, I4)	Ola Strömfors
Konstruktion och analys av algoritmer (C4)	Rolf Karlsson
Programmering Y, grundkurs (Y2)	Christian Gnosspelius
Programmering Y, fortsättningskurs (Y3, Y4)	Tommy Olsson
Kompilatorer och interpretatorer (Y4, I4)	Nahid Shahmehri
Programmeringsteori II (C4)	Douglas Busch
Logikprogrammering (C3, D4)	Jan Maluszynski
Programmeringsteori (C3)	Nahid Shahmehri
Processprogrammering (D3pv, D4, Md4, Y4, I4)	Fredrik Lindström
Operativsystemteori (C3, D3pv, M4d)	Bengt Karlstrand
Programmering och projektarbete i Pascal (C1, D1)	Kerstin Olsson
Lagringsstrukturer (C2, D2, Md3)	Ola Petersson
Programutvecklingsmetodik och programmeringsprojekt D (D3)	Christian Krysander
Programutvecklingsmetodik M (Md3)	Olle Willen
Programutvecklingsmetodik och programmeringsprojekt C (C3)	Bengt Karlstrand
Data och programstrukturer D (D3)	Anders Haraldsson
Data och programstrukturer C (C2)	Roland Rehmnert
Expertsystem - metodik och verktyg (C4, D4)	Sture Hägglund
Distribuerad problemlösning (D4, C4)	Johan Fagerström
Datorer och datorutrustning (I1)	Christian Gnosspelius
Beräkningsbarhet och komplexitet (C4)	Rolf Karlsson

Datorspråk (C3, D3, I4)	Rober Bilos
Artificiell intelligens C (C3)	Arne Jönsson
Artificiell intelligens D (D4)	Arne Jönsson
Programutveckling (I1)	Olle Willen
Datastrukturer och programutvecklingsmetodik (I2)	Christian Gnosspelius
Datalingvistik (C2)	Lars Ahrenberg
Programmering i inkrementellt system (C1)	Anders Haraldsson
Programmering i inkrementellt system (D1)	Patrick Doherty
Interaktiva system (C1, D3pv, D3tk)	Sture Hägglund
Lingvistik grundkurs (C1)	Magnus Merkel
Formella språk och automatateori (C2)	Douglas Busch
Kommunikationspsykologi (C3)	Nils Dahlbäck
Diskret simuleringsteknik (D3, Y3)	Zebo Peng
Datornät (D4, Y4, I4, Md4)	Björn Fjellborg
Datorarkitektur (D4, Y4)	Mikael Patel
Datorstödd elektronikkonstruktion (D4, Y4, I4)	Tony Larsson
Datalogi 1 - baskurs (enstaka kurs Norrköping)	Rolf Nilsson
Datalogi 3 (enstaka kurs Linköping)	Christian Krysander
Programmering i Ada (enstaka kurs Linköping)	Olle Willen

Continuing education 1987

Datalogi, 20p, Ericsson, 20 deltagare 86/87
Datateknik, 20p, Ericsson Information Systems, 20 deltagare, 1987
AI/Expertsystem, 25p, 8 deltagare, 86/87
Datornät, Foa, 25 deltagare, maj 87
Datorsystem och programutveckling, 15p, Ericsson, 22 deltagare
(ingår i Dator- och teleteknik, 60p) ht 87
Programmering Y, Saab-Scania, 12 deltagare (ingår i högre elektroteknisk kurs)
Processprogrammering/operativsystem, 5p, Ellemtel, 21 deltagare, ht 87
LISP/Prolog/AI, 10p, Ericsson, 21 deltagare, 87
AI/Expertsystem, 28p, ASEA, 20 deltagare, började hösten 87
Programutveckling, 2p, Ericsson, 22 deltagare, (ingår i Dator- och teleteknik, 60p)
Datornät, Ellemtel, 25 deltagare, kurs på Kreta, september 87
Kurs i ADA, Ericsson Radio Systems, 10 deltagare, oktober 87
Introduktion till AI och expertsystem, ASEA, 33 deltagare, november 87

2. Undergraduate teaching in the School of Arts and Science

The group for administrative data processing (the ADB-group) is responsible for the courses given by IDA in the undergraduate *Systems analysis* study program in the School of Arts and Science, Linköping University.

The program for systems analysis ranges over three years of fulltime studies. It aims at professional activities of design, evaluation and implementation of computer-based information systems. ADP-systems analysis dominates the program but nevertheless great importance has been attached to other subjects in order to give the program 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 program 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, other languages are taught as well. Within the field of computer science the students take courses in database design, development of interactive systems, communication, evaluation of computer systems, programming methodology, etc. Other subjects 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 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 of 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), a Xerox Ethernet with 32 1108/1109/1186 Lisp Machines, file servers and laser printers, and 15 SUN-3 workstations with file servers.

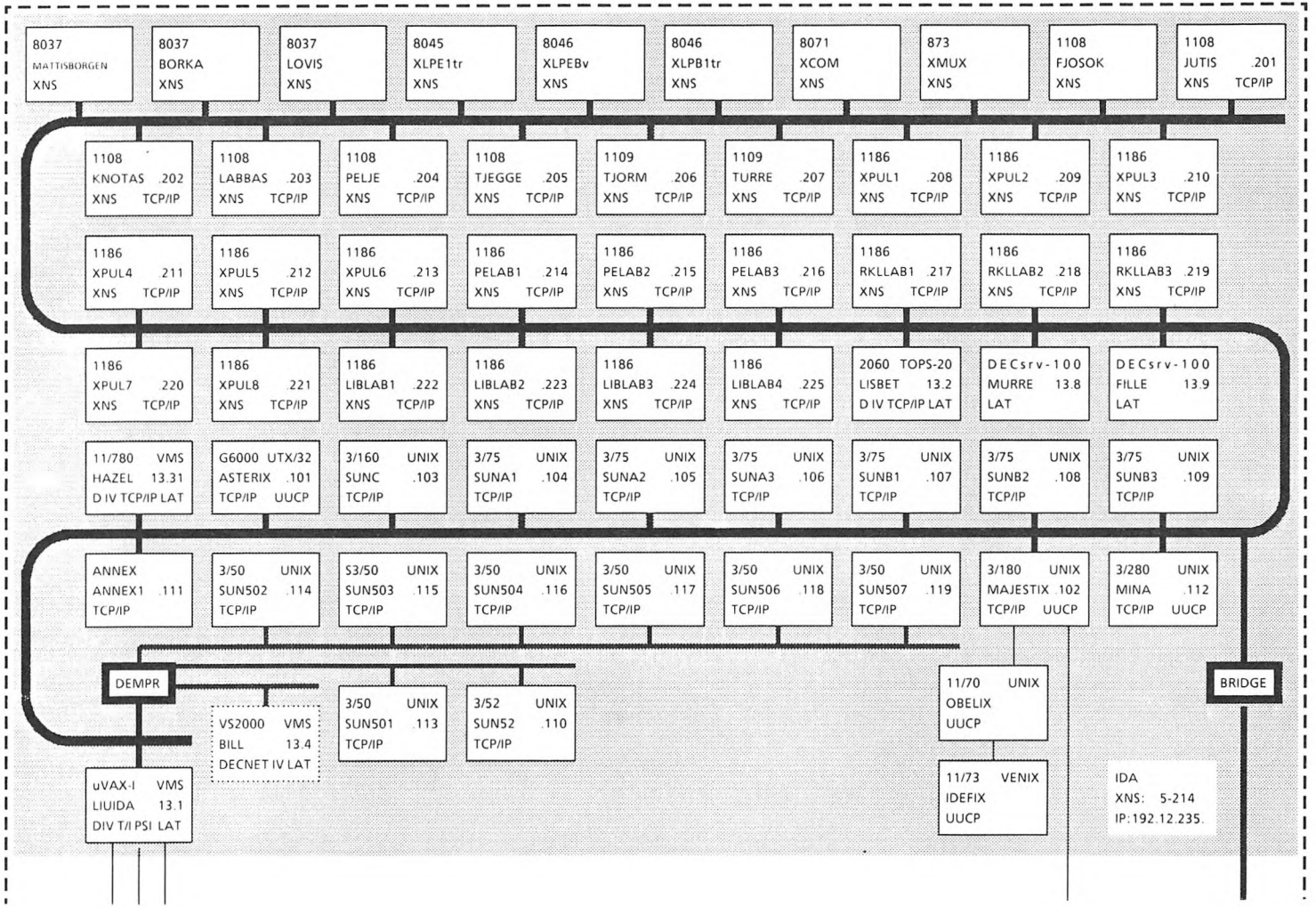
In addition there are lots of smaller computers (MicroVax, PDP-11:s, Macintoshes and other PC:s of various kinds.) There is also special purpose equipment, especially for text processing or for specific research projects.

For research on architecture-related problems, the department also has acquired a number of transputers and a shared NCUBE parallel computer.

Part of the work station equipment was made available through the Xerox Corporation / Rank Xerox University Grants Programme. Our department was awarded 18 Xerox 1186 AI work stations, together with additional services, such as printers and file servers. The application included 6 projects, ranging from knowledge-based application systems to programming environments and use in undergraduate courses. The Linköping Grant was the largest awarded in Europe.

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

Network visible from IDA LiU 880202



Appendix E

Publications

DISSERTATIONS:

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

No 14 **Anders Haraldsson:** A Program Manipulation System Based on Partial Evaluation, 1977.

No 17 **Bengt Magnhagen:** Probability Based Verification of Time Margins in Digital Designs, 1977.

No 18 **Mats Cedwall:** Semantisk analys av processbeskrivningar i naturligt språk, 1977.

No 22 **Jaak Urmi:** A Machine Independent LISP Compiler and its Implications for Ideal Hardware, 1978.

No 33 **Tore Risch:** Compilation of Multiple File Queries in a Meta-Database System, 1978.

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.

No 155 **Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons, 1987.

No 165 **James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning, 1987.

No 170 **Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems, 1987.

(*Dissertation by IDA member published elsewhere.*)

Lars Ahrenberg: Interrogative Structures of Swedish: Aspects of the Relation between Grammar and Speech Acts. (Reports from Uppsala University Department of Linguistics No. 15, 1987).

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, Mikael 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.

No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems. 1986.

No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System. 1987.

No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs. 1986.

No 74 **Christos Levcopoulos:** New Results about the Approximation Behaviour of the Greedy Triangulation. 1986.

No 104 **Shamsul I. Chowdhury:** Statistical Expert Systems - a special application area for knowledge-based computer methodology. 1987.

No 108 **Rober Bilos:** Incremental Scanning and Token-based Editing. 1987.

No 111 **Hans Block:** Sport Sort - Sorting algorithms and sport tournaments. 1987.

No 113 **Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge. 1987.

No 118 **Mariam Kamkar and Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs. 1987.

No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs. 1987.

No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and Use Acceptance of Expert Systems. 1987.

EXTERNAL PUBLICATIONS SINCE 1985

(*Papers published in books, journals or international conference proceedings.*)

1. **Lars Ahrenberg:** Lexikalisk-Funktionell Grammatik på svenska. In *Papers from the 5th Scandinavian Conf. of Computational Linguistics,* University of Helsinki, Dept of General Linguistics pp. 1-12, 1986.
2. **Lars Ahrenberg:** Parsing into Discourse Object Description. In *Proc. of the 3rd Conf. of the European Chapter of the Association for Computational Linguistics* in Copenhagen, April 1-3, 1987.
3. **Lars Ahrenberg, Arne Jönsson:** An Interactive System for Tagging Dialogues. *XIV ALLC Conf.* in Göteborg, June 1-5, 1987.
4. **Lars Ahrenberg:** Functional Constraints in Knowledge-Based Natural Language Understanding. In the 12th International Conf. on *Computational Linguistics* in Budapest, August 22-26, 1988.
5. **Rober Bilos:** A Token-Based Syntax Sensitive Editor. In *Proc of the Workshop on Programming Environments - Programming Paradigms.* Roskilde, 1986.
6. **Rober Bilos, Peter Fritzson:** Experience from a token sequence representation of programs, documents, and their deltas, to appear in *International Workshop on Software Version and Configuration Control,* Grassau, FRG, January 27-29, 1988.
7. **Hans Block:**, SPORT-SORT - Sorting Algorithms and Sport Tournaments. In *Proc. of the 25th Annual Allerton Conf. on Communication, Control, and Computing,* University of Illinois at Urbana-Champaign, 1987.
8. **Christer Bäckström:** A Representation of Coordinated Actions. *Proc. of the 1st Scandinavian Conf. on Artificial Intelligence,* Tromsö, Norway, March 9-11, 1988.
9. **Christer Bäckström:** Logical Modelling of Simplified Geometrical Objects and Mechanical Assembly Processes. In *Proc. of the Workshop on Spatial Reasoning and Multi-Sensor Fusion,* St. Charles, Ill., USA, 1987. Extended version to appear in the *Advances in Spatial Reasoning* book series, ABLEX, 1989.
10. **Shamsul Chowdhury:** Expert System Aid in Statistical Analysis and Interpretation of Data. In *Proc. of the Society of Reliability Engineers,* Outaniemi, 1986.
11. **Shamsul Chowdhury:** State of the art in statistical expert systems. In *Proc. of the Conf. on Expert Systems and their Applications,* Avignon, 1987.
12. **Shamsul Chowdhury, Ove Wigertz:** Microcomputer Oriented Knowledge-Based System for Health Care Improvement in the Developing World. In *Proc. of the IEEE/EMBS 9th Ann. Conf.,* Boston, 1987.
13. **Nils Dahlbäck, Arne Jönsson:** Analyzing Human-Computer Dialogues in Natural Language. Accepted to the *3rd IFAC/IEA Conf. on Man-Machine Systems Analysis, Design and Evaluation,* Oulo, Finland, June, 1988.
14. **Nils Dahlbäck, Arne Jönsson:** Talking to a Computer is not Like Talking to Your Best Friend. *Proc. of the 1st Scandinavian Conf. on Artificial Intelligence,* Tromsö, Norway, March 9-11, 1988.
15. **James A Dean, Andrzej Lingas, Jörg Sack:** Recognizing Polygons or How to Spy, in *The Visual Computer, International Journal of Computer Graphics,* Springer Verlag. See *Proc. of the Allerton Conf. on Communication, Control and Computing,* Urbana, Illinois, 1986, for a preliminary version.
16. **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.
17. **Pierre Deransart, Jan Maluszynski:** Relating Logic Programs and Attribute Grammars. *Journal of Logic Programming,* vol 3, No. 2, pp 119-158, 1985.
18. **Pierre Deransart, Jan Maluszynski:** Programmation logique et grammaires d'attributs, in: *Informatique-85, Symposium sovieto-francaise,* Valgus, Tallinn, 1987, 4-10.
19. **Wlodzimierz Drabent:** Do Logic Programs Resemble Programs in Conventional Languages? In *Proc. of the 4th IEEE Symposium on Logic Programming,* San Francisco, Aug. 31 - Sept.4, 1987

20. **Wlodzimierz Drabent, Jan Maluszynski:** Proving Runtime Properties of Logic Programs. In *Proc of the TAPSOFT'87*, Pisa 1987, LNCS 250, 167-181.
21. **Wlodzimierz Drabent, Jan Maluszynski:** Inductive Assertion Method for Logic Programs. Accepted for publication in the special issue of *Theoretical Computer Science*, 1988.
22. **Dimiter Driankov:** An outline of a fuzzy sets approach to decision-making with interdependent goals. In *Proc. of the 1st IFSA Congress,* Palma de Mallorca, July, 1985. *Int. Journal of Fuzzy Sets and Systems,* vol 21, pp. 275-288. (Extended abstract).
23. **Dimiter Driankov:** Inference with single fuzzy conditional proposition. *Int. Journal of Fuzzy Sets and Systems* vol 24, No 1 pp. 51-63.
24. **Dimiter Driankov:** A calculus for belief-intervals- representation of uncertainty. In *Proc of the Int. Conf. on Information Processing and Management of Uncertainty,* Paris, June 30 - July 4, 1986, pp 235-239. (Extended abstract). In Lecture Notes on *Computer Science,* B. Bouchon and R.R. Yager (eds) vol 286, pp. 205-216.
25. **Dimiter Driankov:** Many-valued logic for belief-intervals: The logical lattice. In *Proc. of the 2nd World Congress of the Int. Fuzzy Sets Association*, pp. 426-429, Tokyo, July 20-25, 1987.
26. **Dimiter Driankov:** Uncertainty Calculus with Verbally Defined Belief Intervals. In *Proc. Int. Joint Conf. of Intelligent Systems*, 1987.
27. **Dimiter Driankov:** A Many-Valued Logic for Belief/Disbelief Pairs: In *Proc. of the 2nd Int. Symp. on Methodologies for Intelligent Sytems*, Charlotte, NC, USA, October, 1987. (Extended abstract).
28. **Dimiter Driankov:** Inference with consistent probabilities in expert systems. In *Proc. Int. Joint Conf. on A.I,* pp 899-901 vol 2, Milano, August,1987. To appear in *Int. Journal of Intelligent Systems*, 1988.
29. **Dimiter Driankov:** Non-truthfunctional Aspects of Belief/Disbelief Pairs. To appear in *Proc of the 1st IFSA-EURO Workshop on Approximate Reasoning*, April 5-9, 1988.
30. **Johan Fagerström:** Experiences with Occam: A Simulator for Asynchronous Processes. In *Proc of the 19th Hawaii Int. Conf. on System Sciences, Hawaii,* Jan, 1986, pp. 95-102.
31. **Johan Fagerström:** Tradeoffs in an Architecture based on Asynchronous Processes. In *Proc of the 2nd Nordic Symposium on VLSI in Computers and Communications*, 1986.
32. **Johan Fagerström, Mikael R.K. Patel:** High-level Simulation of Systolic Architectures. In *Proc of the Int. Workshop on Systolic Arrays,* Oxford, 2-4 July, 1986.
33. **Johan Fagerström, Yngve Larsson and Lars Strömberg:** Debugging Techniques for Distributed Environments. In *Proc. of the Workshop on Compiler and Incremental Compilation* in Bautzen, East Germany, October 11-18, 1986 and the *Proc. of the Workshop on Programming Paradigms and Programming Environments* in Roskilde, Denmark, October 22-24, 1986.
34. **Johan Fagerström, Lars Strömberg:** A Paradigm and System for Design and Test of Distributed Applications. In *Proc. of IEEE Compcon Spring 1988.* San Fransisco, CA, February, 1988.
35. **Johan Fagerström:** Design and test of Distributed Applications. Accepted to the *10th Int. Conf. on Software Engineering*, Raffles City, Singapore, April 11-15, 1988.
36. **Johan Fagerström:** Enabling Structured Debugging of Distributed Sytems. To appear in the *Proc. of the Workshop on Parallel and Distributed Debugging.* Madison, WI, May, 1988.
37. **Hartmut Freitag, Michael Reinfrank:** An Efficient Interpreter for a Rule-Based Non-Monotonic Deduction System. In *Proc. of the German Workshop on AI, GWAI-87,* Springer Verlag, 1987.
38. **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.

39. **Peter Fritzson:** Systems and Tools for Exploratory Programming. Overview and Examples. In *Proc. of the Workshop on Programming Environments - Programming Paradigms,* Roskilde University Centre, Denmark, October 22-24, 1986.
40. **Peter Fritzson:** A Common Intermediate Representation for C, Pascal, Modula-2 and Fortran-77. In *Proc. of the Workshop on Compiler Compilers and Incremental Compilation,* Bautzen, DDR, October 12-17, 1986.
41. **James W. Goodwin:** A Process Theory of Non-Monotonic Inference. In *Proc. of the Int. Joint Conf. on Artificial Intelligence, IJCAI,* 1985.
42. **Sture Hägglund, Christer Hansson, Tomas Sokolnicki:** Knowledge-Based Training of Case Management Routines and Emergency Procedures. In *Proc. of the 3rd Int. Conf. on Expert Systems,* London, 1987.
43. **Sture Hägglund:** Emerging Systems for Computer-Based Knowledge Processing in Office Work. Accepted for publication in *Office: Technology and People,* 1988.
44. **Sture Hägglund:** The Linköping Approach to Technology Transfer in Knowledge Engineering. To appear in *The Knowledge Engineering Review,* vol 2, No 3, Cambridge University Press, 1988.
45. **Tim Hansen:**, Diagnosing Multiple Fault Using Knowledge about Malfunctioning Behaviour. To appear in *Proc. of the 1st Int. Conf. on Industrial and Engineering Applications of AI and Expert Systems*, Tullahoma, Tennessee, USA, 1988.
46. **Roland Hjerppe, Birgitta Olander, Kari Marklund:** Project ESSCAPE - Expert Systems for Simple Choice of Access Points for Entries: Applications of Artificial Intelligence in Cataloging. *IFLA 51st Conf,* Chicago, 18-24 August, 1985.
47. **Roland Hjerppe:** Project HYPERCATalog: Visions and preliminary conceptions of an extended and enhanced catalog. Published in *Intelligent Information Systems for the Information Society,* B C Brookes. Ed. In *Proc. of the IRFIS 6th Conf.* (International Research Forum in Information Science), Frascati, Italy, 15-18 Sept, 1985. Elsevier Science Publishers B.V. (North-Holland), 1986, pp. 211-232.
48. **Roland Hjerppe:** Electronic Publishing: Writing Machines and Machine Writings. The impact of computers on text. Published in *Annual Review of Information Science and Technology,* vol. 21, 1986, M Williams. Ed. Knowledge Industry Publications Inc, pp. 123-166.
49. **Roland Hjerppe:** Knowledge Organizing, Collection Derived, and User Established Structures. Published in Online Public Access to Library Files: *2nd National Conf.* J Kinsella Ed. Elsevier International Bulletins, Oxford 1986, pp. 101-110.
50. **Roland Hjerppe:** LINS - LIBLAB's Name Handling System. A knowledge-based system for authority control of personal names according to AACR2, Ch.22, Headings for persons. June, 1987, p.8. I: C. Bossmeyer, ED. The Library of the Future. In *Proc. of the ELAG (European Library Automation Group),* 11th Library Systems Seminar, Frankfurt, April 1-3, 1987. Deutsche Bibliothek, Frankfurt am Main. 1987. ISBN:3-922051-19-7. pp.67-80.
51. **Roland Hjerppe:** Computer Networks as a Publication Medium? Implications for Libraries. August, 1987, p 4. Presented at *53rd IFLA General Conf.* ,Brighton, England, August 16-21, 1987.
52. **Roland Hjerppe:** IFLA and Professional Communication. Presented at the *IFLA 53rd General Conf.* ,Brighton, England, August 16-21, 1987.
53. **Johan Hultman:** COPPS A Software System for Defining and Controlling Actions in a Mechanical System. In *Proc. of the IEEE Workshop on Languages for Automation*, Wien, September, 1987.
54. **Arne Jönsson, Nils Dahlbäck:** Talking to a Computer is not like Talking to Your Best Friend. In *Proc. of the SCAI,* Tromsö, Norway, 1988.
55. **Mariam Kamkar, Nahid Shahmehri, Peter Fritzson:** Affect-Chaining and Dependency Oriented Flow Analysis Applied to Queries of Program. To appear in *Proceedings of the ACM Symposium on Personal and Small Computers.* Cannes, France, May, 1988.
56. **Mariam Kamkar, Nahid Shahmehri:** Runtime Dependent Program Flow Analysis. In *Proc. of the Workshop on Programming Environments - Programming Paradigms,* at Roskilde University Centre, Denmark, October 22-24, 1986.
57. **Rolf G. Karlsson, Ian Munro:** Proximity on a Grid. In *Proc. of the 2nd*

Symposium on Theoretical Aspects of Computer Science (1985), Springer-Verlag Lecture Notes on Computer Science 182, pp. 187-196.

58. **Rolf G. Karlsson, Ian Munro, Ed Robertson:** The Nearest Neighbour Problem on Bounded Domains. In *Proc. of the 12th Int. Colloquium on Automata, Languages and Programming* (1985), Springer-Verlag Lecture Notes on Computer Science 194, pp 318-327.
59. **Rolf G. Karlsson:** Point Location in Discrete Computational Geometry. In *Proc. of the 6th Brazilian Congress on Computing,* 1986, pp. 561-569.
60. **Rolf G. Karlsson:** Greedy matching on a grid. To appear in *BIT,* 1988.
61. **Rolf G. Karlsson, Mark Overmars:** Normalized Divide-and-Conquer: A Scaling Technique for Solving Multi-dimensional Problems. In *Information Processing Letters,* 26, pp. 307-312, 1988.
62. **Rolf G Karlsson, Mark Overmars:** Scanline Algorithms on a Grid. To appear in *BIT,* 1988.
63. **Jan Komorowski, Jan Maluszynski:** Logic Programming and Rapid Prototyping. Report TR-01-86, Harward University, Aiken Computation Laboratory. In *Science of Computer Programming,* 9, 1987, pp. 179-205.
64. **Krzysztof Kuchcinski and Zebo Peng:** Microprogramming Implementation of Timed Petri Nets. *In Proc. of the 2nd Nordic Symp. on VLSI in Computers and Communications,* Linköping, Sweden, June, 1986.
65. **Krzysztof Kuchcinski, Zebo Peng:** Parallelism Extraction from Sequential Programs for VLSI Applications. Presented at *Euromicro 87,* Southsea-Portsmouth.
66. **Krzysztof Kuchcinski, Zebo Peng:** Microprogramming implementation of timed Petri nets. In *North-Holland INTEGRATION, the VLSI Journal,* No 5, 1987.
67. **Krzysztof Kuchcinski, B. Wiszniewski:** Path Analysis of Distributed Programs. In *Proc. of the 1988 ACM Computer Science Conf.*, Atlanta, Georgia, February, 23-25 1988.
68. **Tony Larsson:** Semantics of a Hardware Specification Language and Related Transformation Rules. *In Proc. of the 2nd Nordic Symp. on VLSI in Computers and Communications,* Linköping, Sweden, June, 1986.
69. **Tony Larsson:** Semantics of a Hardware Specification Language. *Microprocessing and Microprogramming,* vol.18, No 1-5, pp. 637-643, 1986.
70. **Tony Larsson:** Specification and Verification of VLSI Sytems Actional Behaviour - *8th International Symposium on Computer Hardware, Description Languages and their Applications,* Amsterdam, April 27-29, 1987.
71. **Tony Larsson:** Semantics of a Hardware Specification Language and Related Transformation Rules. In *North-Holland INTEGRATION, the VLSI Journal,* No 5, 1987.
72. **Yngve Larsson:** A Testbed Environment for Debugging Distributed Systems. To appear in the *Proc. of the Workshop on Parallel and Distributed Debugging.* Madison, WI, May, 1988.
73. **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.
74. **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.
75. **J.Leszczylowski, Staffan Bonnier, Jan Maluszynski:** Logic Programming with External Procedures: Introducing S-unification. Accepted for publication in *Information Processing Letters,* 1987.
76. **Bengt Lennartsson:** Programming Environments and Paradigms - Some Reflections. In *Proc. of the Workshop on Programming Environments - Programming Paradigms,* Roskilde, Denmark, October 1986.
77. **Christos Levcopoulos:** Minimum Length and "Thickest-First" Rectangular Partitions of Polygons. In *Proc. of the 23rd Allerton Conf. on Comm., Control and Computing,* Illinois, October, 1985.
78. **Christos Levcopoulos:** A Fast Heuristic for Covering Polygons with Rectangles. In

Proc. of the 5th Int. Conf. on Foundations of Computation Theory, GDR, (1985), *Lectures Notes in Computer Science*, vol 199, Springer Verlag.

79. **Christos Levcopoulos:** Fast Heuristics for Minimum Length Rectangular Partitions of Polygons. In *Proc of the 2nd ACM Symposium in Computational Geometry*, Yorktown Heights, June, 1986.
80. **Christos Levcopoulos:** An Omega (square root(n)) Lower Bound for the Non-Optimality of the Greedy Triangulation. In *Information Processing Letters*, No 25 (1987), pp. 247-251.
81. **Christos Levcopoulos, Andrzej Lingas:** On the Approximation Behavior of the Greedy Triangulation for Convex Polygons. In *Algorithmica*, No 2, 1987, pp. 175-193.
82. **Christos Levcopoulos:** Improved Bounds for Covering General Polygons with Rectangles. In *Proc of the 7th Conf. on Foundations of Software Technology and Theoretical Computer Science*, Pune, INDIA, December 17-19, 1987. *Lecture Notes 287, Springer Verlag.*
83. **Christos Levcopoulos, Andrzej Lingas och Jörg Sack:** Nearly Optimal Heuristics for Binary Search Trees with Geometric Generalizations. In *Proc of the ICALP'87* Karlsruhe, West Germany, July, 1987. *Springer Verlag Lecture Notes 267.*
84. **Andrzej Lingas:** A Linear-Time Heuristic for Minimum Weight Triangulation of Convex Polygons. In *Proc. of the Allerton Conf. on Communication, Control, and Computing*, Urbana, Illinois, 1985.
85. **Andrzej Lingas:** Subgraph Isomorphism for Easily Separable Graphs of Bounded Valence. In *Proc. of the 11th Int. Workshop on Graphtheoretic Concepts in Computer Science*, Castle Schwanberg, Wuerzburg, Germany, June, 1985.
86. **Andrzej Lingas:** On Partitioning Polygons. In *Proc of the 1st ACM Symposium on Computational Geometry*, Baltimore, Maryland, June, 1985.
87. **Andrzej Lingas:** Subgraph Isomorphism for Biconnected Outerplanar Graphs in Cubic Time. In *Proc. of the 3rd Symposium on Theoretical Aspects of Computer Science*, January, 1986, Orsay, France. *Lecture Notes in Computer Science, vol 210, Springer Verlag.* To appear in *Theoretical Computer Science.*
88. **Andrzej Lingas:** On Approximation Behavior and Implementation of the Greedy Triangulation for Convex Planar Point Sets. In *Proc of the 2nd ACM Symposium in Computational Geometry*, Yorktown Heights, June, 1986.
89. **Andrzej Lingas, Andrzej Proskurowski:** Fast Parallel Algorithms for the Subgraph Homeomorphism and the Subgraph Isomorphism Problems for Classes of Planar Graphs. In *Proc. of the 7th Conf. on Foundations of Software Technology and Theoretical Computer Science*, Pune, INDIA, December 17-19, 1987, *Springer Verlag Lecture Notes 287.*
90. **Andrzej Lingas, Christos Levcopoulos, Jörg Sack:** Algorithms for Minimum Length Partitions of Polygons of Polygons. In *BIT*, Vol 27, No 4, pp. 474-479.
91. **Andrzej Lingas:** A New Heuristic for Minimum Weight Triangulation. In *SIAM Journal of Discrete and Algebraic Methods.* Vol 8, No 4, pp. 646-653. (This is an improved version of LiTH-IDA-R-84-15).
92. **Andrzej Lingas:** A Space Efficient Algorithm for Greedy Triangulation.In *Proc. of the 13th IFIP Conf. on System Modelling and Optimization*, Tokyo, 1987, Springer Verlag Lecture Notes.
93. **Andrzej Lingas, Maciej Syslo:** A Polynomial-time algorithm for subgraph isomorphism of two-connected series-parallel graphs. To appear in *Proc. of the ICALP*, Tampere, July, 1988. Lecture Notes in Computer Science, Springer Verlag.
94. **Jalal Maleki:** VIVID. The Kernel of a Knowledge Representation Environment Based on the Constraints Paradigm of Computation. In *Proc. of the 20th Hawaii Int. Conf. on System Sciences*, Kailua-Kona, 1987.
95. **Jan Maluszynski, H. Jan Komorowski:** Unification-Free Execution of Logic Programs. In *Proc of the IEEE Symposium on Logic Programming*, Boston, 1985.
96. **Magnus Merkel:** The Interpretation of Swedish Temporal Frame-Adverbials.In *Proc. of the 10th Scandinavian Conf. on Linguistics*, Bergen, June 11-13, 1987.
97. **Magnus Merkel:** A Novel Analysis of Temporal Frame Adverbials. In *Proc. of the 12th International Conf. on Computational Linguistics* in Budapest, August 22-26,

1988.

98. **Minton, Carbonell, Knoblock, Kuokka and Henrik Nordin:** Improving the Effectiveness of Explanation-based Learning. In *Proc. of the Workshop on Knowledge Compilation*, September 24-26, Oregon State University, 1986.
99. **Torbjörn Näslund:** An Experimental Implementation of a Compiler for Two-Level Grammars. In: Z.W Ras and M. Zemankova (eds.). In *Proc. of the 2nd Int. Symposium, Methodologies for Intelligent Systems*, North-Holland, 1987, pp. 424-431.
100. **Ulf Nilsson:** AID: An Alternative Implementation of DCGs. *New Generation Computing,* vol 4, No 4, pp. 383-399, 1986.
101. **Henrik Nordin:** Using Typical Cases for Knowledge-Based Consultation and Teaching. In *Proc of the 3rd Annual Conf. on Applications of Expert Systems*, Orlando, Fla., 1986.
102. **Lin Padgham:** A Description of LINCKS: Linköpings Intelligent Knowledge Communication System. In *Proc. of the IFIP Working Conf. on Methods and Tools for Office Systems*, Pisa, Italy, October 22-24, 1986.
103. **Lin Padgham, Ralph Rönnquist:** From a Technical to a Humane Environment: A Software System Supporting Co-operative Work. In *Proc. of the GDI International Conf. on USER INTERFACES*, Rüschlikon, Switzerland, October 20-21, 1986.
104. **Lin Padgham, Ralph Rönnquist:** An Imperative Object Oriented System. In *Proc. of the 20th Hawaii International Conf. on System Sciences*, 1987, vol 1, p 516.
105. **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.
106. **Mikael Patel:** A Threaded Interpretive Language Supporting Programming in the Large. In *Proc. of the 6th Rochester Forth Conf,* University of Rochester, Rochester, New York, June 11-14, 1986.
107. **Zebo Peng and K Kuchcinski:** Synthesis of Control Structures from Petri Net Descriptions, *Microprocessing and Microprogramming,* Vol.18, No 1-5, 1986, pp. 335-340.
108. **Zebo Peng:** Synthesis of VLSI Systems with the CAMAD Design Aid. In *Proc. of the 23rd ACM/IEEE Design Automation Conf,* Las Vegas, June, 1986.
109. **Zebo Peng:** Integration of VLSI Design Tools by a Unified Design Representation. Published as a part of the *Proc of the 2nd Nordic Symposium on VLSI in Computers and Communications,* June 2-4, 1986.
110. **Zebo Peng:** A Formal Approach to the Synthesis of VLSI Systems from their Behavioral Descriptions. In *Proc of the 19th Annual Hawaii Int. Symp. on System Sciences, Hawaii,* January 1986, pp. 160-167.
111. **Zebo Peng:** Construction of Asynchronous Concurrent Systems from their Behavioral Specifications. In *Proc. of the 10th World Computer Congress IFIP-86,* Dublin, Ireland, September 1986, pp. 859-864.
112. **Ivan Rankin:** SMORF - An Implementation of Hellberg's Morphology System. In Papers from the *5th Scandinavian Conf. of Computational Linguistics,* University of Helsinki, Dept of General Linguistics, pp. 161-172.
113. **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.
114. **Michael Reinfrank, Hartmut Freitag:** An Integrated Non-Monotonic Deduction and Reason Maintenance System. In Herbert Stoyan (ed.) *Proc. of the Workshop on Truth Maintenance*, Berlin, 1986. Springer Verlag.
115. **Michael Reinfrank:** Reason Maintenance Systems. In Herbert Stoyan (ed.) *Proc. of the Workshop on Truth Maintenance*, Berlin, 1986. Springer Verlag.
116. **Michael Reinfrank:** Multiple Extensions, where's the Problem? In F. Brown (ed.) *The Frame Problem in Artificial Intelligence. Proc. of the 1987 Workshop.*
117. **Ralph Rönnquist:** The Information Lattice of Networks Used for Knowledge Representation. In *Proc. of the Int. Symp. on Methodologies for Intelligent Systems*, Charlotte, NC, October, 1987.
118. **Piotr Rudnicki, Wlodzimierz Drabent:** Proving Properties of Pascal Programs in MIZAR 2, *Acta Informatica,* vol 22, pp. 311-331, 1985.

119. **Kevin Ryan:** The Value of Mixed Metaphors in Computer Education. In *Proc. of the Nat. Computer Education Conf.*, San Diego, 1986.
120. **Kevin Ryan, J A Redmond, and Others:** Surveying Software Tools for a Method Driven Environment. *IFIP,* Dublin, September, 1986.
121. **Kristian Sandahl, Sture Hägglund, Jan-Olof Hildén, Roland Rehmnert, Lars Reshagen:** The Antibody Analysis Advisor and its Migration into a Production Environment. In *Proc. of the 1st Int. Conf. on Expert Systems*, London, 1985.
122. **Kristian Sandahl:** The Migration of Expert Systems into Production Environments. In *Proc. of the Nord-Info Seminar on Knowledge Engineering*, Köpenhamn, 1986.
123. **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.
124. **Erik Sandewall, Ralph Rönnquist:** A Representation of Action Structures. In *Proc. of the 5th National Conf. on Artificial Intelligence, AAAI-86*, Philadelphia, 1986.
125. **Erik Sandewall:** Non-Monotonic Inference Rules for Inheritance with Exception. In *Proc. of the IEEE, Special Issue on Knowledge Representation,* 1986.
126. **Erik Sandewall:** Specification Environments for Information Management Systems. Panel position paper in *Proc. of the IFIP Congress,* 1986.
127. **Erik Sandewall:** The Pipelining Transformation on Plans for Manufacturing Cells with Robots. In *Proc. of the Int. Joint Conf. on A.I.*, Milano, August, 1987.
128. **Ola Strömfors:** Editing Large Programs Using a Structure-Oriented Text Editor. In *Proc. of the Int. Workshop on Advanced Programming Environments.* Trondheim, Norway, June, 1986.
129. **Ola Strömfors:** A Structure Editor as a Template for Programming Environment Functions. In *Proc. of the Workshop on Programming Environments - Programming Paradigms*, at Roskilde University Centre, Denmark, October 22-24, 1986.
130. **Bo Sundgren:** How to Satisfy a Statistical Agency's Need for General Survey Processing Programs. In *Proc. of the 45th Session of the International Statistical Institute*, Amsterdam, August 12-22, 1985.
131. **Erik Tengvald:** The AIM-project: Establishing a Knowledge Engineering Environment for the Application of Artifical Intelligence Techniques in the Manufacturing Industry. Presented at *Artificial Intelligence - Supercomputers*, Umeå, Sweden, June 22-24, 1987.
132. **Arja Vainio-Larsson:** Metaphors as Communicators of Conceptual Ideas. In *the 9th Scandinavian Research Seminar on Use and Development of Information Systems,* Båstad, August 19-22, 1986.
133. **Mats Wirén:** A Comparison of Different Rule-Invocation Strategies in Context-Free Chart Parsing. In *Proc. of the 3rd Conf. of the European Chapter of the Association for Computational Linguistics*, in Copenhagen, Denmark, April 1-3, 1987.
134. **Mats Wirén:** A Control-Strategy-Independent Implementation of PATR. *Proc. of the 1st Scandinavian Conf. on Artificial Intelligence*, Tromsö, Norway, March 9-11, 1988.

DEPARTMENTAL REPORTS 1987

LiTH-IDA-R-87-01 **Wlodzimierz Drabent:** Do Logic Programs Resemble Programs in Conventional Languages?
LiTH-IDA-R-87-02 **Rober Bilos:** A Token-Based Syntax Sensitive Editor.
LiTH-IDA-R-87-03 **Mikael Patel:** A Threaded Interpretive Language Supporting Programming in the Large.
LiTH-IDA-R-87-05 **Christer Bäckström:** Logical Modelling of Simplified Geometrical Objects and Mechanical Assembley processes.
LiTH-IDA-R-87-06 **Johan Hultman:** COPPS A Software System for Defining and Controlling Actions in a Mechanical System.

LiTH-IDA-R-87-07 **Peter Haneclou:** A Formal Approach to Reason-maintenance Based on Abstract Domains.

LiTH-IDA-R-87-08 **Christos Levcopoulos, Andrzej Lingas, Jörg-R. Sack:** Nearly Optimal Heuristics for Binary Search Trees with Geometric Generalizations.

LiTH-IDA-R-87-09 **Andrzej Lingas, Marek Karpinski:** Subtreee Isomorphism and Bipartite Perfect Matching are Mutually NC Reducible.

LiTH-IDA-R-87-10 **Andrzej Lingas:** On Parallel Complexity of the Subgraph Isomorphism Problem.

LiTH-IDA-R-87-11 **Sture Hägglund, Christer Hansson, Tomas Sokolnicki:** Knowledge-Based Training of Case Management Routines and Emergency Procedures.

LiTH-IDA-R-87-13 **Mats Wirén:** A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing.

LiTH-IDA-R-87-14 **Peter Fritzson:** Window System Architectures - an Overview.

LiTH-IDA-R-87-15 **Nils Dahlbäck:** Kommunikation med datorer i naturligt språk - vad är det och vem behöver det?

LiTH-IDA-R-87-16 **Shamsul I. Chowdhury:** State of the Art in Statistical Expert Systems.

LiTH-IDA-R-87-17 **Tony Larsson:** Specification and Verification of VLSI Systems Actional Behaviour.

LiTH-IDA-R-87-18 **Henrik Nordin:** Reuse and Maintenance Techniques in Knowledge-based Systems.

LiTH-IDA-R-87-19 **Zebo Peng:** Construction of Asynchronous Concurrent Systems from their Behavioral Specifications.

LiTH-IDA-R-87-20 **Krzysztof Kuchcinski, Zebo Peng:** Parallelism Extraction from Sequential Programs for VLSI Applications.

LiTH-IDA-R-87-21 **Harold W. Lawson:** Challenges and Directions in Computers and Education.

LiTH-IDA-R-87-22 **Lars Ahrenberg, Arne Jönsson:** An Interactive System for Tagging Dialogues.

LiTH-IDA-R-87-24 **Sven Moen:** Drawing Dynamic Trees.

LiTH-IDA-R-87-25 **Arne Jönsson:** Naturligt språk för användardialog och databasförfrågningar

LiTH-IDA-R-87-26 **Jonas Löwgren:** Applying a Rapid Prototyping System to Control Panel Dialogues.

Diskussion Debatt:

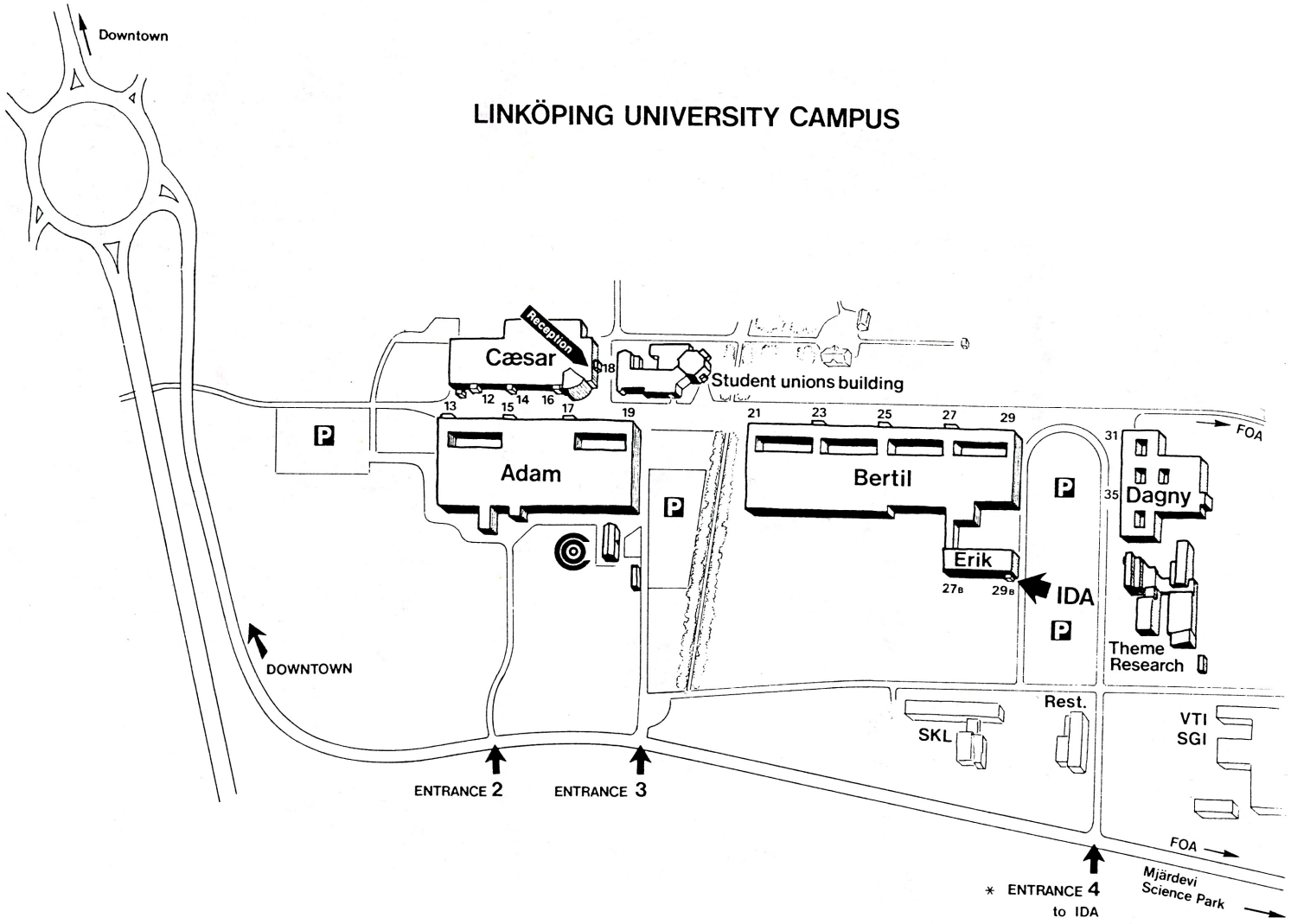
LiTH-IDA-R-87-04 **Arja Vainio-Larsson:** Datavetenskap: Teknik och Vetenskap.

LiTH-IDA-R-87-12 **Sture Hägglund:** Yrkeskompetens och kunskapskrav i 90-talets datormiljö.

LiTH-IDA-R-87-23 **Börje Langefors:** Idén om informationssystem.

LIBLAB RESEARCH REPORTS

LiU-LIBLAB-R-87:1 **Roland Hjerppe:** LINS - LIBLAB'S Name Handling System. A Knowledge-Based System for Authority Control of Personal Names According to AACR2, Ch.22., Headings for Persons.



* Entrance 4 to the Dept. of Computer Science marked with IDA on the picture.