Department of Computer and Information Science

Annual
Research
Report
1986

Linköping University and Institute of Technology

ISBN 91-7870-168-6
ISSN 0281-4250

LiTH-IDA-R-86-44
January 1987

The Department of Computer and Information Science

Linköping University

Annual Research Report 1986

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 department has a commonly organized integrated program for graduate studies (PhD and Licentiate degree), with a large faculty engaged in research and thesis supervision. In addition to the research organization and the extensive undergraduate course program, there is also a knowledge transfer program in the department involving cooperation with a number of large Swedish companies on medium to long term R&D issues.

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

Erik Sandewall

1.1 Research Objectives in IDA.

1.1.1 Current Research Objectives.

This is the January, 1987 issue of the yearly overview report of research done at the Department of Computer and Information Science (IDA) of Linköping University. 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 is 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 described in chapters 4-13, 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.

The effort in industrial information technology is still just in the planning stage, and government approval of the plan came in February, 1987. The actual work in the new Center for Industrial Information Technology will begin in mid-1988. It may eventually involve most of the departments of our school. At present, the departments of Mechanical Engineering, Electrical Engineering, and Physics are the ones mostly involved in the planning work besides IDA.

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 May, 1986, and there has been a fairly lengthy debate during the fall about how to proceed for the future. We hope that a consensus solution can be reached soon.

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. Since 1985 there has not been any frame program in our area, and our grants have mostly been for two years at a time. A working group appointed by STU is now preparing a proposal for the next frame program, which would run from January 1, 1988. IDA has a representative (Erik Sandewall) in the working group.

1.2 Organization.

1.2.1 IDA's current research 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.

IDA has presently about 120 employees. This figure includes 16 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 eight (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 (Lawson/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 Organizational Changes During 1986.

The department's organization has only changed at one point during 1986: effective July 1, the previous artificial intelligence laboratory was split into two new laboratories, namely the Laboratory for Artificial Intelligence Environments (AIELAB), headed by Erik Tengvald, and the Natural Language Processing Laboratory (NLPLAB), headed by Lars Ahrenberg. The goals and the activities of these new laboratories are described in their respective chapters of this report.

1.2.3 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 Eva-Chris Svensson. 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 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 project, a Scandinavian project on SYstems Development and Profession Oriented Languages, together with. the universities in Aarhus (Denmark) and Oslo (Norway).

- 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 is now in a planning phase. Prometheus is concerned with future traffic and automobile systems and is a joint effort by the European car manufacturers.

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 University teaching.

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%.

Chapter 2 gives an overview of our undergraduate educations for Swedish readers, i.e. the chapter assumes a knowledge of the Swedish educational system. Here we shall briefly review the system, for the benefit of the international reader, and using terms from the U.S. educational system for comparison.

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. The degree system there is presently being modified.

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.

1.4.2 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 are reported in chapter 3.

1.4.3 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 undergraduate students, such as e.g. Programsystem AB, 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 Research Facilities.

1.5.1 Computer Equipment.

During 1985 Linköping was one of a number of European universities, which were invited by the Rank-Xerox Corporation to submit applications for advanced computing equipment, including AI workstations. Our application covered 6 projects, ranging form knowledge-based systems including applications in medicine, libraries and office work, to programming environments and improved undergraduate education.

In May, 1986, Rank-Xerox made a donation of 18 Xerox workstations, plus additional file servers, print servers, and other computer equipment including software to our department. This grant covered the needs expressed in the 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 experiences of advanced workstations in the undergraduate and masters-level education.

During the year we have also installed 7 SUN-3 workstations, which have been purchased using a grant from *Forskningsrådsnämnden*, and miscellaneous smaller computers.

The AIELAB has started a project to build a transputer-based computer system with a 'hyper-cube' configuration, for use in its own research as well as by other labs.

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.

1.5.2 Office Space.

The department moved into an attractive new building, the E building, in April, 1985. Because of the on-going expansion of the department's activities, we are already very crowded in the new premises. One-third of the department is still in the old building, and in space we have borrowed from other departments. We expect this problem to get worse before it gets better.

Figure 1.1. *Peter Fritzson reports from a year at SUN during the weekly information meeting in the coffee room.*

2.

Undergraduate Education

Anders Haraldsson

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.

2.1 Undergraduate curricula

As the first institute of technology in Sweden we started 1975 the D-line (Computer Science and Technology - "Datateknik-linjen") as a four-year 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. The first class of students from this curriculum has now graduated. It is already quite clear that these students develop a different 'culture', and in particular a more solid basis for graduate research in computer science, than what students in our other lines do. While certainly our other lines will continue to be of very high importance, the computer science line has provided a significant addition. A number of students from the C-line has during the last year been accepted as graduate students.

The D-line will from this 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 except 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 gives 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.

2.2 Continued education for Swedish Industry

We have started a programme for "continued education" of engineers in computer science. 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

Two base-part courses, Computer Science, 20p, have been given to LM Ericsson in Stockholm during 1985-1986. The two additional parts are planned to start during 1987.

2.3 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 has been developed 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.

There is a proposal to start a Swedish International University (SIU) giving Master of Science Programmes. 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. He has had a temporary 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 1988.

2.4 Conference activities

In June 1986 a Conference about the universities role in computer science education was held at Kolmården. The conference was organized by Linköping University in collaboration with the Royal Institute of Technology in Stockholm and sponsored by the Swedish Board of Universities and Colleges. The organizers from Linköping University were Harold Lawson and Anders Haraldsson.

Participants were invited from all Swedish Universities and Colleges and selected participants from the Swedish Industry. Invited guests were Professor Mary Shaw, Software Engineering Institute and Carnegie-Mellon University, presenting their new curriculum for Undergraduate Computer Science and a proposal for "Education for the Future of Software Engineering", and Professor Jacques Habenstreit, Ecole Superieure d'Electricite, France presenting "Computers in Education".

From IDA speeches were held by Harold Lawson, "Challenges and Directions in Computers and Education", Anders Haraldsson, "An overview of Swedish education in Computer Science" and Erik Sandewall, "Fourth generation computer science education".

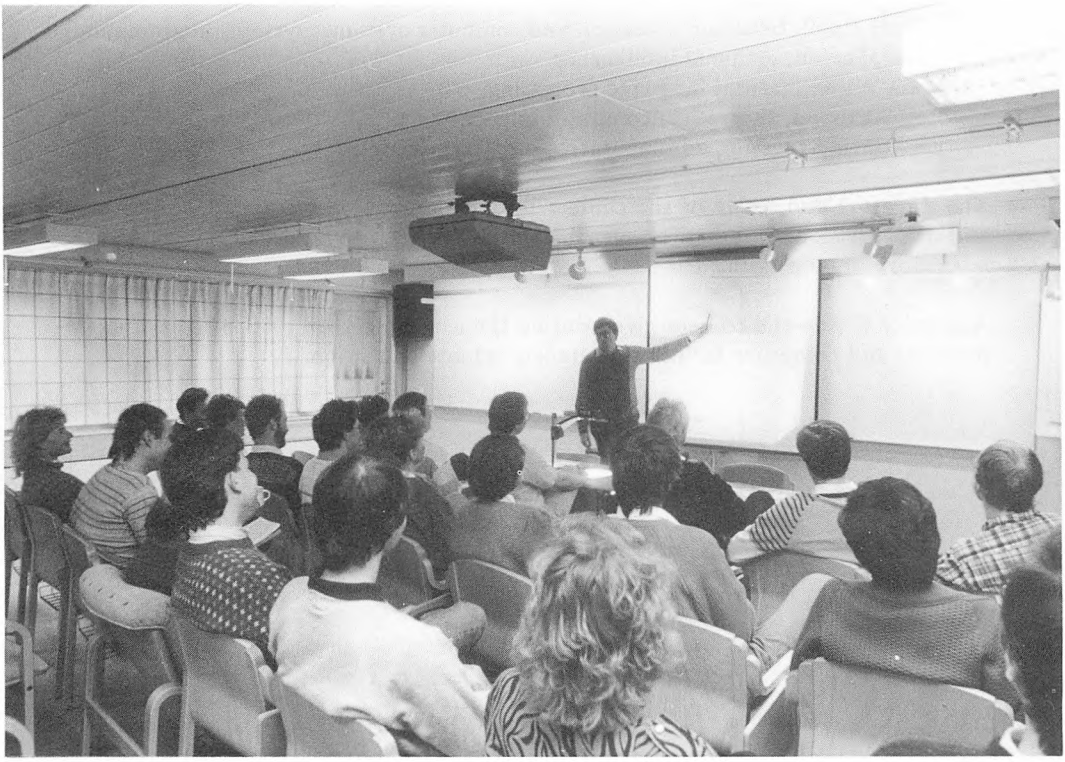


Figure 2.1. *Meeting with the director of studies Anders Haraldsson.*

2.5 Organization

The undergraduate education at IDA is organized as follows:

The Committee for Undergraduate Education (IDUN - IDA's undervisningsnämd), 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
Eva-Chris Svensson, 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.

3.

The Knowledge Transfer Programme

Sture Hägglund

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.

Our department has actively pursued these strategies, e.g. by issuing a special series of reports summarizing important results in central research areas specifically directed towards industry (*"industri-serien"),* by arranging and participating in tutorial conferences, such as e.g. *the AI Week* in Linköping last spring, *the Software Environment Week* in the fall, and the regular SOFT tutorials (on Artificial Intelligence, Software Development Environments, Prolog, AI and Expert Systems, etc.), by developing continuing education programs for industry and by direct consultation and cooperation in applied projects with industry and organizations in the public sector.

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

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

3.1 Orientation of the program

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

* Software competence is becoming an increasingly critical resource, also in areas or companies which are not primarily concerned with information technology;
* Software costs pose serious problems and it is recognized that improved hardware will be no means change this situation;
* Computerized systems are still difficult to change and maintain, while the demand for flexibility and adaptability become increasingly crucial;
* There is a fast international development with huge national and international R&D information technology programmes, where every country strives for a position on the markets of the future.

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

The goal for the KTP program thus is to:

* promote an effective use of the results from the STU (Swedish Board for Technical Development) programmes for knowledge development in information technology and other efforts supported by governmental funds;
* secure the availability of qualified competence within novel information technology areas of high importance;
* provide a environment for early experiments and evaluation of new technologies as a knowledge base for industry;
* Contribute to an awareness about industry needs within the university.

A fundamental assumption is that research projects of a high international standard is carried out within areas of common interest. In connection with this research the university undertake to *organize projects for medium term visitors from industry with an emphasis on learning, technology evaluation and*

other forms of knowledge transfer. The obligation for each participating company is to:

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

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

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

The key ideas of the KTP effort can be summarized as:

1. The university carries out research projects relevant for industry in areas which are expected to have high future potential.
2. The programme involves companies which expect advanced information processing techniques to be of crucial importance for future operation.
3. The emphasis is on next-generation software technology.
4. Novel and advanced equipment and software tools are used in experimental settings.
5. The research content of the program should be of high international quality.
6. The ultimate goal of joint activities is to supply participating companies with a qualified background for strategic decision making, in-house use, and training within the information technology area.

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

- The immediate availability of powerful development environments (advanced work stations and software tools) for experimentation with new software technologies.
- Support for evaluation of new trends, methodologies and products.
- Sharing of resources, especially critical-size research teams in areas where competent personnel is a scarce commodity.
- Participation in pilot projects near the edge of the research front line.

- Education of own personnel.
- Basis for recruiting students after undergraduate education.

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

3.2 Participants and organization

Since the start in 1984 six companies have joined the programme, namely in the order of the time for their affiliation: S-B-Banken, Ericsson, ASEA, Alfa-Laval, Philips and Sandvik Coromant. S-E-Banken has now completed the active participation in Linköping. Ericsson (through Ericsson Telecom) and Alfa-Laval (through SATT Control, formerly Alfa-Laval Automation) have passed on to the status of associated members, which means that no participants from those companies are working in Linköping, but that cooperation and other forms of contacts continue.

The programme has a budget of its own, but participating individuals are associated with one of the research laboratories at IDA and assigned supervisors and assistants within the research organization. Courses and other such activities are as usual within IDA organized at the department level.

Programme leader is professor Erik Sandewall. Sture Hägglund is responsible for project coordination and Arne Jönson for the course program.

3.3 Example of KTP training projects.

The core activity for industry participants in the knowledge transfer program is to carry out what we call *training projects.* Such projects are selected with the aim to cover some important application area from the company, with the constraint that the goals should be feasible within the allocated time, while att the same time provide an optimal learning environment for the involved people. The intention is thus that training projects should result in:

- demonstration prototypes;
- experiences of knowledge acquisition;
- an evaluation of available tools;
- a basis for strategic decision making in the company.

A number of such projects have been carried out. Some of them are described below, while others are concerned with areas such as operations planning in mechanical engineering or support for decision making by operators in real-time systems. Although the intention is not to try to build full-scale

systems during knowledge transfer activities, the results have been very valuable both as a basis for start-up of second-generation application projects in the companies or as an inspiration for continued research projects in the department.

3.3.1 Economical decision making: Real estate transfers

A training project carried out by Lars Bengtsson, S-E-Banken, was concerned with the possibility of providing knowledge-based support for legal and economical advice-giving in a bank, in particular the transfer of real estate within a family. The project resulted in a successful demonstration system (LUCKY), which convincingly showed the potential for knowledge systems as a way of communicating expertise within a service organization.

However the project also illuminated some of the limitations of the pure rule-based backward-chaining approach taken in LUCKY. This inspired a licentiate thesis project (Nordin) where these problems were handled within a framework of non-monotonic reasoning, which also explicitly handled the strategic knowledge involved in an estate transfer consultation. In a related master's thesis project (Hansson) it was then shown how the knowledge base could be reused to provide a training environment for the bank officer.

This work is a good example of the mutual benefits of joint KTP work, where applications inspire research and research exploits new possibilities of practical interest.

3.3.2 Sales support: Spot welding robot configuration

The GARMAN project (sales support for spot-welding robot configuration) was carried out in cooperation with ASEA Robotics. The main task of the system is to support a sales engineer in the configuration of a spot-welding robot, including the welding gun and the transformer. Important aspects are to secure that relevant and complete information is acquired from the customer, to suggest a suitable combination of equipment and to verify that electrical and mechanical constraints are satisfied. Additional complications arise from the fact that customer preferences must be taken into account and that equipment from different vendors has to be fitted together.

The EMYCIN system was used as the basic tool to realize the GARMAN prototype system, although significant parts of the problem were solved directly in Lisp. In later experiments the GARMAN application has been used as a test case in connection with e.g. the use of hybrid development environments.

3.3.3 Maintenance and repair: Separator systems

An important application area for expert systems technology is fault diagnosis of technical equipment. For companies which deliver complicated equipment world wide with a reliability and maintenance responsibility, knowledge systems promise to offer decisive advantages over traditional approaches. In one training project, performed by Börje Rosenberg and Ove Hanebring of Alfa-Laval, such a problem has been studied. A full scale prototype of a trouble shooting expert system for separator-based plants has been developed, based on Epitool after initial experiments with EMYCIN and KEE. Contributions to the system also come from Epitec and from master's thesis students at IDA (graphics editor). The prototype is now further developed inside Alfa-Laval, while core issues in the area are studied in continued research projects at IDA and in the joint KTP cooperation.

3.3.4 Customer support: Performance tuning of computers

Configuration, i.e. the selection and combination of components into a system which fulfils some set of requirements and constraints, is another important application area. In a project conducted by Bengt Rosén from Ericsson Information Systems, the proper configuration of a customer installation of a computer system was studied. The prototype developed allows a customer to analyze system logs and readjust system parameters in order to improve the performance of his installation, without the need to summon a specialist from the computer vendor.

3.4 KTP as a knowledge engineer training program

The major area of interest for the companies participating in the knowledge transfer program 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.

However it is also increasingly clear that the spectrum of applications, tools and required skills for knowledge systems activities is very broad and that different kinds of training is appropriate for different purposes. In particular we recognize the need for *application-oriented knowledge engineers*, who

* are familiar with the application domain;
* build applications with shells;
* work from specifications of well understood problems;
* have limited knowledge of fundamental AI theory;
* develop "shallow reasoning" knowledge systems.

For more large-scale undertakings *AI-oriented knowledge engineers* are needed, who

* build general models of domain knowledge;
* take responsibility for tool selection, adaption, integration, etc.
* are experienced in knowledge acquisition methodology and system prototyping.
* have qualified competence in AI and computer science
* develop also "deep reasoning" knowledge systems.

Our knowledge engineer training program tries to cover both these needs. Thus it should fill the gap between commercially available introductions including at best a course program for a number of days or a few weeks, and 4 year computer science curriculums in the area (e.g. the "*datavetenskaliga linjen*"). We offer thus, in addition to the KTP project participation with apprenticeship in research, training projects and tools evaluation activities, a one year half time course program including:

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

There is also currently a master's programme in computer science / knowledge engineering being planned to be offered within the Swedish International University. This activity is described in more detail in the chapter on undergraduate education.

3.5 Experiences and plans for further KTP development

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

Before the program 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 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 program. However it still seems that the the problem of allocating the right personnel to program 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 a central position in company. The other person is then younger and with a qualified computer science education. The period of participation 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 main interest has, as could be expected, been in the area of AI and expert systems, but also to a certain degree in programming environments. For companies that have completed the knowledge transfer programme we offer an associated membership, which does not include active project work in the department but cooperation in various forms.

We are presently investigating the possibilities to further develop the kind of activities offered within an associated membership in KTP. 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. It should be a common interest to encourage and develop such contacts within the frame of some kind of advanced cooperative research and knowledge development program. Current proposals for a competence center in industrial information technology at Linköping University might further improve the possibilities to achieve a effective effort in this direction.

4.

ACTLAB
The Laboratory for Complexity of Algorithms

Andrzej Lingas

4.1 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 group believe that work on algorithm and data structures efficiency is no less important than the development of new, faster computers.

The laboratory is a continuation 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. In 1986, the laboratory acquired a new graduate student, Ola Petersson, whereas its other graduate student, Christos Levcopoulos, defended his licentiate thesis entitled "*New Results about the Approximation Behavior of the Greedy Triangulation*".

The second 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. The idea of the project was to concentrate on the problems in which the group members already gained international recognition: data structures on bounded domains, geometric decomposition problems and subgraph isomorphism. The considered problems have

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

applications in VLSI chip design and fabrication, graphics, robotics, numerical analysis, chemistry and optimization.

Comparing the research areas of the laboratory in 1986 with those in 1985, one can observe a more explicit interest in parallel algorithms (see Section 8.2.3) and a greater interaction between research topics in computational geometry and data structures (see Section 8.2.1.1, 8.2.2.2). The new student, Ola Petersson, has started preliminary research on the design and analysis of fast and hardware-feasible parallel algorithms for geometric and combinatorial problems.

In addition to the research, the group 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. Interactions with other members of the department are a source of new research problems for the laboratory members. For instance, the laboratory members have recently become interested in the problem of efficiently drawing trees raised by Sven Moen, and maintained their interest in graph concepts in network semantics considered by Ralph Rönnquist.

4.2 Group Members

Group leadership : Andrzej Lingas, Ph.D.

Bodil Mattsson-Kihlstrom, secr.

Supervisors: Rolf Karlsson, Ph.D.

Graduate Students:

Christos Levcopoulos, Ph.D. (spring -87)

Ola Petersson

4.3 Current Research

4.3.1 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.

Triangulations of Planar Figures (Christos Levcopoulos, Andrzej Lingas)

The so called greedy triangulation is one of the most known heuristics for minimum weight (length) triangulation of planar figures (the latter triangulation has applications in interpolation 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 his licentiate thesis, Christos Levcopoulos has 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. Christos has also derived a low approximation constant for an interesting class of polygons. On the other hand, Christos has 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 a greedy triangulation of the so called semi-circular polygons has been presented in Christos's licentiate thesis.

Andrzej Lingas has shown independently that the greedy triangulation for convex polygons approximates the optimum, using a quite different method than Christos. Also, he has developed a natural generalization of Voronoi diagrams to include visibility barriers and shown it to be useful in finding a shortest diagonal of the input figure and consequently in implementing the greedy triangulation efficiently. Fast algorithms for constructing such diagrams in the general and polygon case have been designed.

The lower bound of Christos on the length of the greedy triangulation has been already accepted for a journal publication. For the most cases, the other discussed results are submitted for publication.

Recognizing Polygons (Andrzej Lingas)

A new class of so called pseudo star-shaped polygons has been introduced. 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. Algorithms for testing whether a polygon is pseudo star-shaped from a given point, and for constructing all regions from which the polygon is pseudo star-shaped, respectively running in linear and quadratic time, have been designed. Also, it has been shown that many standard computational geometry problems can be solved efficiently for pseudo star-shaped polygons. The above results have been obtained jointly with Dr. Dean of Northern Bell Research and Dr. J. Sack of Carleton University, and already have been published.

4.3.2 Data Structures

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)

New results concerning optimal binary search trees with zero key access probabilities (with applications eg. in point location) have been derived. 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 $O(n^3 m^2)$, and if the polygon is convex, in time $O(n^3 \log m)$. The above results have been partly obtained in cooperation with Dr. J. Sack, and they are submitted for publication.

4.3.3 Parallel Graph Algorithms

(Andrzej Lingas)

The subgraph isomorphism problem is one of the most general NP-complete graph problems and it has many applications in computer science. It consists in determining whether a graph is isomorphic to a subgraph of another graph. A parallel algorithm for the subgraph isomorphism problem has been designed using an earlier sequential algorithm for this problem. When the input graphs are connected, have a good separator (e.g. planar graphs) and relatively small valence, the algorithm runs in sub-linear or even poly-log time using sub-exponential or pseudo-polynomial number of processors, respectively. Also, it has been shown that the problem of subgraph isomorphism restricted to trees and that of bipartite perfect matching are mutually reducible in poly-log time using a polynomial number of processors. In consequence, the problem of subgraph isomorphism for trees can be solved by a random parallel algorithm running in poly-log time and using a polynomial number of processors. The latter results have been partly obtained in cooperation with Prof. M. Karpinski of Bonn University. The discussed results have been already submitted for publication.

4.4 External contacts

Rolf Karlsson: written two papers with Mark Overmars, Rijksuniversiteit Utrecht, The Netherlands, a research continued when Dr Overmars visited IDA in March. Writing a paper with Thomas Strothotte, Stuttgart University, West Germany, who visited IDA in April. Joint research to extend conference papers is under way with Ian Munro, University of Waterloo, Canada, and Ed Robertson, Indiana University, USA; work continued during a visit to Waterloo in June and when Dr Munro visited IDA in November. During a trip

to Brazil in July and August research contacts with 5 universities (in Recife, Belo Horizonte, Sao Paulo, Campinas and Campina Grande) were initiated. Gave an invited talk at the *6th Brazilian Congress on Computing* .

Christos Levcopoulos: Presented his paper at the *2nd ACM Symposium on Computational Geometry,* Yorktown Heights, USA.

Andrzej Lingas: Presented papers at the *3rd Symposium on Theoretical Aspects of Computer Science,* Orsay, France (January), and at the *2nd ACM Symposium on Computational Geometry,* Yorktown Heights, USA. Written two papers on computational geometry with Dr. J. Sack during his visit in the School of Computer Science at Carleton University, Ottawa, Canada (May, June). A revisit of Dr. Sack in June and July of 1987 is expected. A cooperation with Prof. M. Karpinski of Bonn University, Bonn, West Germany, has been established in the area of superfast parallel graph algorithms, during Prof. Karpinski's visit at IDA (October).

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

Courses for Graduate Students

An important task of the group is to spread the knowledge of algorithm analysis and complexity theory among graduate students within the department. The following graduate courses are offered for the academic year 86-87:

Amortized Computational Complexity

Computational Geometry

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)

5.

AIELAB
The Artificial Intelligence Environments Laboratory

Erik Tengvald

5.1 Introduction

By the first of July the artificial intelligence laboratory (AILAB) was split into two laboratories, the laboratory for artificial intelligence environments (AIELAB) and the natural language processing laboratory (NLPLAB).

The research activity of AILAB had long been concentrated in two main research areas namely, knowledge representation and natural language. The rationale for the split was the increase of both the natural language and knowledge representation sides of the AILAB research group. It is our experience that too big research groups do not function well.

The research outlook and current projects of AIELAB are of course based in the considerable experience gained in the previous knowledge representation work in AILAB.

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

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

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

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

In the ICONStraint project Jalal Maleki [Maleki 86, Maleki 87] has researched the possibility of using the constraint programming paradigm as basis for such a declarative system. The choice of constraints as opposed to other declarative paradigms was based in another experimental experience of the OBS project, namely the very great computational demands of geometric reasoning.

The results of the ICONStraint project indicate that the expressibility of constraints is indeed sufficient for handling many geometric problems encountered in the OBS setting. Unfortunately it seems as if constraint based systems executing on todays machines is too slow to make the explorative programming method applicable in a geometric reasoning context.

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

Alas, if you do not increase the raw processing power of the hardware. This is the intended method of attack chosen in the below described AIM project.

A subsidiary knowledge representation research activity at AILAB has been Jim Goodwins extremely promising but more formal research on non-monotonic logic [Goodwin 84, Goodwin 85]. Research in the non-monotonic logic area has been transfered to the RKLLAB and is currently not pursued at the AIELAB.

5.2 Researchers and Project

The AIELAB research interest is primarily 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 goal of the AIM project is to design an AI-environment able to support expertsystems performing combined symbolic and geometric reasoning. The 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 expertsystem shell feasible and usable. Such an increase in processing power is only possible by the use of parallell programming.

Consequently, AIELAB has parallell programming as a subsidary research area. During this and probably also during the next year, parallell

programming is the area in which most if not all research will be concentrated. The parallell programming research is formalized in the hideshape subproject.

We would like to stress that the parallell programming research in the hideshape subproject is focused and constrained by the AIM setting. This focusing greatly simplify the parallell programming research.

5.2.1 Laboratory members

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

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

5.2.2 The AIM project

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

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

We have found it useful to coin a new word for this kind of reasoning, namely geombolic reasoning. The characterizing trait of geombolic reasoning is the intense interdependence between the geometric and the symbolic reasoning processes.

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

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

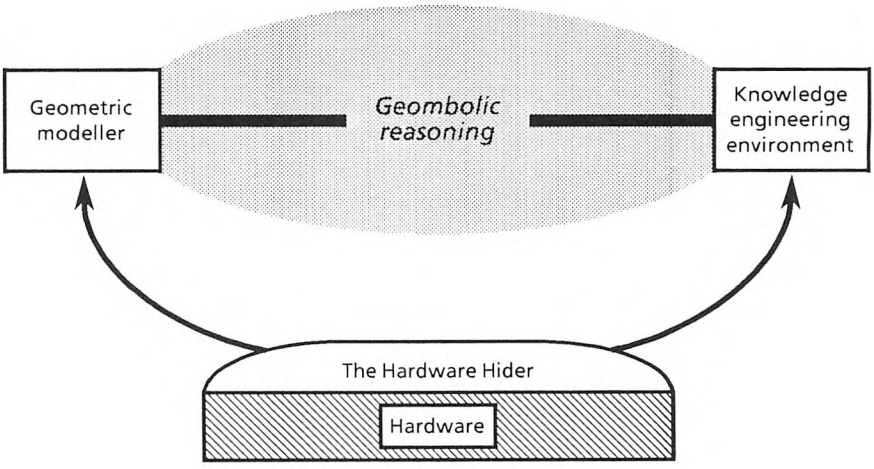


Figure 5.1. *The overall structure of the research vehicle.*

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

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

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

A hardware hider is basic systemware, which hides the complexity of the hardware. More precisely, it frees the programmer from the tedious tasks of memory and process management. This must be done without introducing any unnecessary restrictions in the use of the hardware. Metaphorically: A hardware hider is hiding the nitty gritty details of the hardware, without hiding it's soul. This is illustrated in Fig. 5.1.

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

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

Knowledgeshape.

Our knowledge engineering environment is intended to be pluralistic environment supporting many programming paradigms under an integrated debugging environment. Idealy, this system would look like.

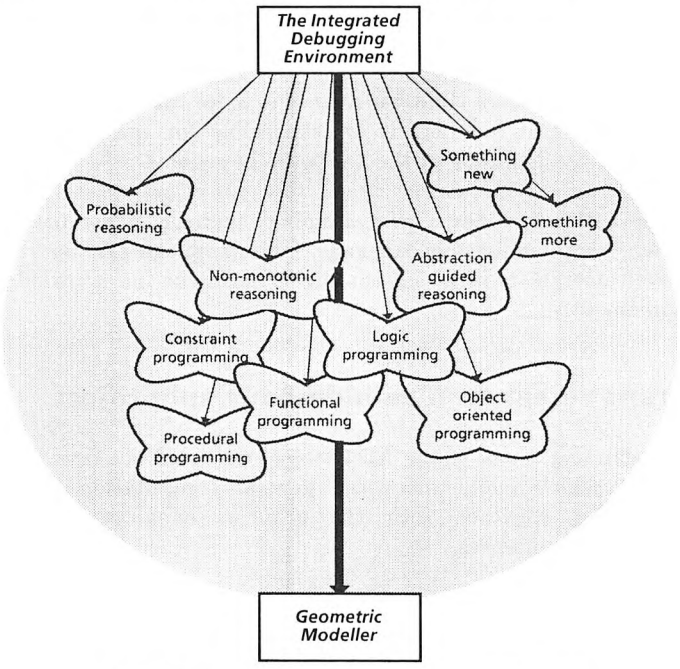


Figure 5.2. *The knowledge engineering environment*

Of course we will not be able to implement all this software in the AIM-project. However, we find it meaningful to keep the ideal knowledge engineering environment in mind while designing the other parts of the AIM system. This will protect us from making low level design decisions which would make the ideal unreachable.

There are of course some programming paradigms which are necessary for a geombolic AI-environment. To begin with we intend to incorporate the object oriented and constraint programming paradigms. Object oriented programming is very natural when describing systems of interacting things. Such systems are very common in the manufacturing industry. Moreover the window interface is best implemented using object oriented programming. Constraints are perfect for describing invertible relationships. Most relationships of physics and geometry are invertible.

In a slightly longer perspective it is natural to begin incorporating the logic programming paradigm in knowledgeshape. In this we hope to establish practical cooperation with the logic programming laboratory at our department. The exact order of the introduction of further programming paradigms is currently left open.

Solidshape.

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

Hideshape.

The hardware hider is called Hideshape, since it hides the complex shape of the hardware from the programmer. It is responsible for memory and process management, and the handling of the hardware in general.

The hideshape system defines a programming language. More precisely, it is the interpreter/compiler for this language. This language can be viewed as a very high level assembler, in which the higher levels of the knowledge system are to be implemented.

5.3 Current research: The hideshape subproject

In the initial planning phase of the AIM-project we could not find any machine on the market which could meet the general computation needs of the geometric modeller together with the graphics capability necessary for presentation of the geometries.

Consequently, we initiated design of such a machine based on INMOS transputers. In a short time there did however appear a machine satisfying our requirements on the market. This is the NCUBE/ten supplemented with the manufacturers graphics board.

Unfortunately we could not get funding for the purchase of an NCUBE machine. In this situation we reverted to a transputer based solution. Fortunately, the education department were planning to buy transputer board for educational purposes. We could use 8 such board supplemented with a small graphics board and a home made communications board, In an NCUBE style transputer system, like this:

Moreover, a group at the national defence research institute have applied for a small (16 nodes) NCUBE. They have kindly invited us to use this machine when (and if) it arrives. Then we no longer need to rely on home made hardware.

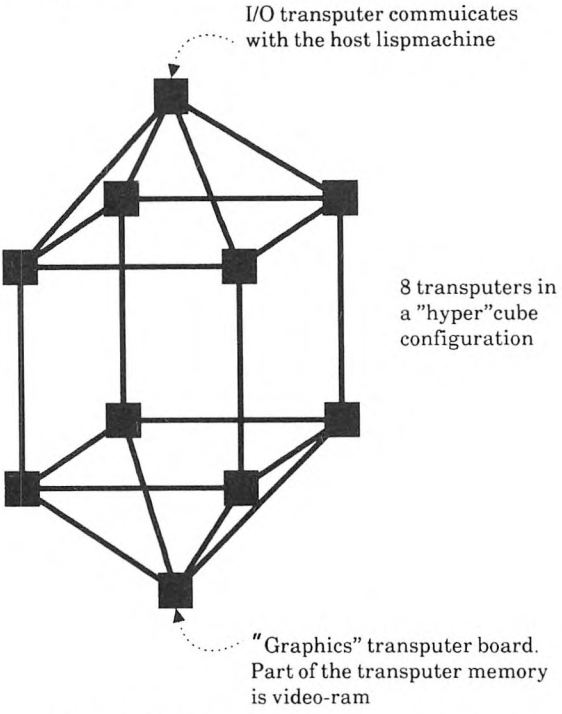


Figure 5.3. *An NCUBE style transputer system*

While waiting for the hardware to arrive we have concentrated our resources in litterature studies and design discussions. These design discussions have been concentrated on the hideshape subproject, while taking due regard to the higher levels of the AIM-environment.

Our current design of the hideshape system can be pictorially be described like:

The hideshape language is, like for example lisp, inspired by recursive function theory. The basis for hideshape is consequently a fairly small set of subroutines implemented in assembler. This set of basic subroutines is called the "hideshape core".

The rest of the hideshape language and its interactive and inkremental environment is implemented on top of the hideshape core. The hideshape core does not contain any primitives with explicite access to the processors, memories and communication links of the hardware. The only hardware oriented primitives in the hideshape core, are primitives for terminal and file, input and output. Since the rest of the hideshape language and the environment is buildt on top of the hideshape core, the same is true for the hideshape language and environment.

Consequently the hideshape core will act as a barrier which protect the hideshape programmer from the complex structure of the parallell hardware.

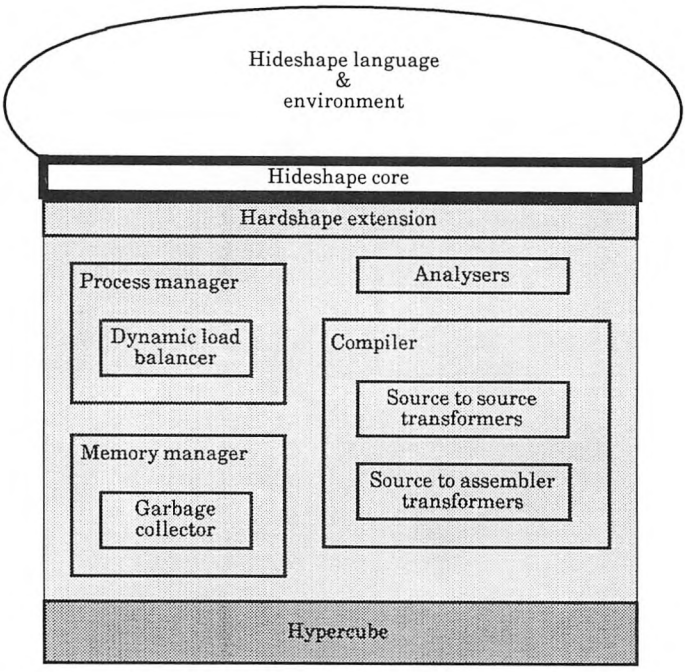


Figure 5.4. *Overall view of the hideshape system*

By extending the hideshape core with a small number of primitive subroutines able to handle the hyper-cube's processors, memories and communication links, we obtain quite simply a very inefficient interpreter for the hideshape language on the hyper-cube. This extension to the hideshape core is known as the hardshape extension.

As well as acquiring our hideshape interpreter, we now also have an interpreter for another language, hardshape, from which the hypercube structure is directly accessible. More precisely, the hideshape language is a 'hardware independent' subset of the hideshape language.

The system designers (ie. we) can implement in this language efficient system-ware to support hardshape and thereby hideshape, too. Hardshape acts as out system implementation language. We are, of course, designing the hardshape extension so that its 'concepts' lie close to the intended application of the hardshape extension, namely the construction of the other system-ware components.

The compiler is, hopefully, a well-structured and modernised version of a standard lisp compiler. For source to source transformations we envisage simple partial evaluation transformations, beta expansion, constant folding and, possibly, type folding. For source to assembly transformations we have: code concatenation, box-unbox, list-array and recursion removal. The various transformers exploit applicable results from the analysers.

The memory manager including its main component, the garbage collector, is relatively conventional. A degree of complication arises in the handling of what corresponds to pointers going over processor borders.

The process manager's duties include placing processes in the hypercube's processors so that the hypercube is used maximally. Ideally no processor should remain unused unnecessarily. Many researchers considers the design of such a perfect process manager to be very hard if not impossible [McCharty 86].

In our case however, we work in the AIM setting. In this setting the major part of the computational resources will be consumed by the geometric routines.

The geometric routines are not too many and not too complex. Consequently, the hypercube utilisation patterns of the geometric routines can be handcoded. Thus, they can be made to utilize the hypercube very efficiently, in many cases perfectly.

Now, the major part of the computational resources will be consumed by routines which are very efficient. Thus the overall system processor utilization will be reasonable. In principle we could let the symbolic reasoning take place on a single node of the hypercube. Symbolic reasoning do progress with sufficient speed on Lisp-machines no stronger the a single NCUBE node.

However, to let all the other nodes wait for calls to geometric routines from a single sequential process, seem a bit wasteful. Consequently, we will supply the hideshape system with a simple dynamic load balancer. We could for example use a simple master-slave scheme as discussed in [Williams 86].

The task of all the system-ware components except the hardshape extension is to make more efficient the execution of programs written in the hardshape language. Also all the system-ware components are written in the hardshape language. As we successively improve the various components, the efficiency of all the components will increase. An improved compiler will make the process manager, the analyser, the memory manager and the compiler itself more efficient. The same applies to the other components.

The hardshape system is, thus, implemented in itself. We expect to be be able to utilise the advantages of such an implementation strategy to a wider extent than usual. This expectation forms the basis of our intention to implement the not insignificant hardshape system and thereby also the hideshape language and its environment in a short time.

5.4 Research cooperation

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

We have also been in contact with Oslo University, discussing the geometric modeller. As noted above we also cooperate with the national defence research institute.

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

References

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

[Goodwin 84] J.W. Goodwin, WATSON - A Dependency Directed Inference System, In Proc. of the AAAI Workshop on Non-Monotonic Reasoning, New Palz, NY, 1984.

[Goodwin 85] J.W. Goodwin, A Process Theory of Non-Monotonic Inference. in Proc. of the Int. Joint Conf. on Artificial Intelligence, IJCAI, 1985.

[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.

[Maleki 87] J. Maleki, "VIVID, The Kernel of a Knowledge Representation Environment Based on the Constraints Paradigm of Computation", Proc. HICSS-20, Hawaii, January 1987.

[McCharty 86] J. McCharty, Personal communication.

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

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

[Williams 86] W. Williams, Load Balancing and Hypercubes, A Preliminary Look, Second Conference on Hypercube Multiprocessors, Knoxville, Tennesse, September 29 - October 1, 1986

6.

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 with 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. Projects usually take an experimental approach and emphasize participation in application-oriented projects with industry and the public sector.

6.1 Projects and Researchers

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

6.1.1 Summary of research 1986.

During the last year, activities in the laboratory have been carried out mainly in four 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

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

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.

- *Knowledge acquisition and maintenance environments.* It is a fact generally agreed upon, that knowledge representation and knowledge acquisition are key issues in successful expert system development. Our approach emphasizes the importance of making knowledge reusable, both for simplified development of generic applications and for the reuse of problem solving knowledge for e.g. teaching and training. Current efforts are concerned with the acquisition and representation of knowledge about technical equipment in a maintainable way. Practical applications are studied together with SATT Control (previously Alfa-Laval Automation), where we are implementing expert systems for fault diagnosis including graphical interfaces for visual interaction. This work is done by Nordin (who was visiting the Carnegie-Mellon University in Pittsburgh during 1985/86, working on the PRODIGY learning apprentice project under Jaime Carbonell) together with Hansen et al.

- *Knowledge-base migration.* 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 will among other things be presented in a forthcoming licentiate thesis by Sandahl. 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.

- *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). 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 discourse rather

than simple translation from internal to external representation (Rankin).

- *Architecture of development tools and environments.* 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 forces us to devote some effort to this issue (Rehmnert).

o **Statistical information systems.** This group, under the leadership of adj. professor Bo Sundgren, is studying systems for processing of aggregated information concerning groups of objects in a universe of discourse. A major activity during the year has been the study of how expert systems can be an aid for this purpose. The result of this study will be presented in a forthcoming licentiate thesis by Chowdhury. More applied work has been done as a preliminary explorative implementation of certain aspects of a *statisticians workstation* on a Xerox Lisp machine (Chowdhury, Sisk et al.) and as a study of the need for more precise and exact methods for strategic interpretation of administrative business data (Wallgren and Wallgren).

o **Method-driven software development environments.** During 1985/86 Kevin Ryan from Trinity College in Dublin has visited the department, working on the use of knowledge-based support for software engineering in a context of the ESPRIT ToolUse project. In particular he has studied how the supporting tools can be made to work more intimately integrated with a chosen development methodology. Although this work is oriented towards supporting a traditional programming process, it shares many of the the underlying research issues with other work on knowledge-based expert systems in the laboratory.

o **Knowledge transfer and other externally oriented activities.** During the year KTP participants from ASEA, Alfa-Laval and Philips have been associated with our lab. Under this period especially Alfa-Laval has inspired research tasks and master's theses assignments for lab members.

We have also conducted a study for *Sveriges Mekanförbund* in the area of knowledge-based expert systems, together with a support committee with representatives from industry. A follow-up project in the same area started in the fall 1986.

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 AI week in Linköping, April 1986, with more than 300 visitors.

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

6.1.2 Personnel, ASLAB, spring 1987.

The following list presents persons active in ASLAB during 1986 and early 1987.

Project leadership/thesis supervision:

Sture Hägglund, PhD, docent, lab leader
Gunilla Lingenhult, secr.

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

Graduate students:

Shamsul Chowdhury, BSc, MSc
Tim Hansen, MSc
Malin Johansson, MSc (starting late fall 1986)
Henrik Nordin, MSc, Tekn. Lic. (at CMU 1985/86)
Ivan Rankin, MA
Roland Rehmnert, MSc
Kristian Sandahl, MSc (at Epitec AB 1986/87)
Pål Sørgaard, MSc (visiting from Århus 1986)

Associated persons:

This list includes persons who are actively participating in ASLAB projects, either as industry participants in the knowledge transfer program, as cooperating researchers in other departments or as undergraduate students doing their masters thesis projects in the lab.

Martin Andersson, undergraduate student
Hans Block, SCB, Stockholm
Henrik Eriksson, undergraduate student
Torbjörn Eriksson, undergraduate student
Ove Hanebring, Alfa-Laval Automation, Lund
Stefan Hammar, Philips Elektronikindustrier, Uppsala
Christer Hansson, research assistant
Kerstin Johansson, undergraduate student
Christian Krysander, lecturer
Jonas Löwgren, undergraduate student

Sven Moen, undergraduate student
Gösta Nilsson, Högskolan i Örebro
Erling Nordmark, Philips Elektronikindustrier, Järfälla
Pablo Lozan-Villegas, ASEA, Västerås
Lars Reshagen, Dept of medical informatics
Börje Rosengren, Alfa-Laval Automation, Lund
Päivi Sisk, undergraduate student
Tomas Sokolnicki, undergraduate student
Toomas Timpka, Dept of medical informatics
Anders Wallgren, Dept of math/statistics
Britt Wallgren, Dept of math/statistics

Figure 6.1. *Researchers in ASLAB. From the left: Shamsul Chowdhury, Roland Rehmnert, Malin Johansson, Henrik Nordin, Kristian Sandahl, Sture Hägglund and Tim Hansen. Missing on the picture is Ivan Rankin and Bo Sundgren.*

6.2 Direction of research.

A central theme for our research is the application of knowledge-based techniques for software development, both for improved development support environments and for extending established software design practises with

knowledge-based techniques. In this process we emphasize the potential benefits of applied AI and knowledge-based methods for producing more useful, easy-to-change and understandable software, rather than as a way to solve computationally difficult problems, but also as a way to introduce the following qualities into software development, maintenance and use:

- Interactive support for application modelling, through the use of AI-inspired representation techniques which allow incremental modification and maintenance of knowledge stored in the system.
- Advanced dialogue management, including (restricted) natural language explanations, queries and result presentations.
- Learning support (for the user), based on the fact that information inside the system may be inspectable and also reusable for teaching and training.
- More maintainable software, since the distance between what is stated by the domain expert and what is entered into the system can be shorter than in conventional programming.

Areas

Presently we are particularly interested in problems related to

- knowledge acquisition and maintenance, especially for representations supporting qualitative reasoning in addition to shallow heuristics with a high potential for reusability of stored knowledge.
- generic knowledge systems, where similar representation and reasoning patterns reappear in different applications and where the specific functions needed are specified by changing or extending a generic knowledge base.
- iterative methods for system development, which combine database modelling with knowledge-based prototyping of system functions and allow adaptive maintenance when requirements are changing.

Goals.

The goals for our research can be summarized as that we want to:

- develop methodology and tools for knowledge-based expert systems;
- understand the basis for human-computer cooperative problem solving;
- improve our ability to create transparent reusable software;
- promote an increased productivity and maintainability in applications software development.

Style of research.

Our style of research is characterized by a heavy emphasis on founding the

selection or research problems and the assessment of results on practical experiences from realistic applications. On the other hand there is a limit on the ability to undertake real application projects in a research group, which in general tends to bias research towards "programming-in-the-small" rather than towards "programming-in-the-large".

We try to cope with this problem by associating people from industry in knowledge transfer activities and to cooperate with other researchers in application-oriented projects. In such projects we attempt to apply and evaluate previous results regarding methods and techniques. Experiences are then generalized when possible and form the basis for theory development, as well as for the design and redesign of methods and tools.

6.3 Review of current research activities.

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

6.3.1 Knowledge-Based Software Systems.

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

This work is oriented towards the application of knowledge-based techniques for software development, in particular architecture and development of generic application systems, knowledge-based design and maintenance of software and the integration of methodology for developing expert systems with more conventional information technology such as database management and office information systems.

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. It is our intention to focus on this problem area in our work on methods and tools for knowledge system development.

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.

In the KAD environment a set of common services for knowledge acquisition and knowledge representation will be provided together with a layered structure of generic knowledge about certain problem types (such as e.g. diagnosis, configuration, planning, etc.) and application areas (such as e.g. medicine, economy, technical equipment, etc.). Subsequently application-specific information can be stored in order to customize the development system to a certain class of applications. Developing an application can then be seen as a further specialization and instantiation of object classes, processing rules, integrity constraints, input/output behavior, etc.

In the KAD environment we thus expect to find e.g. the following facilities:

- Tools and methodological support for knowledge acquisition.
- The knowledge base, containing e.g. knowledge about the application, control knowledge, and meta-level knowledge relating to the various services provided.
- Mechanisms for utilizing the knowledge for problem-solving and other purposes ("inference engines" and model interpreters).
- Other tools for analysis, migration to the delivery environment, etc.

We expect to restrict our study of KAD environments to certain classes of applications. One such application area is 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 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.

Currently 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.

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 connected to the Epitool-based knowledge system.

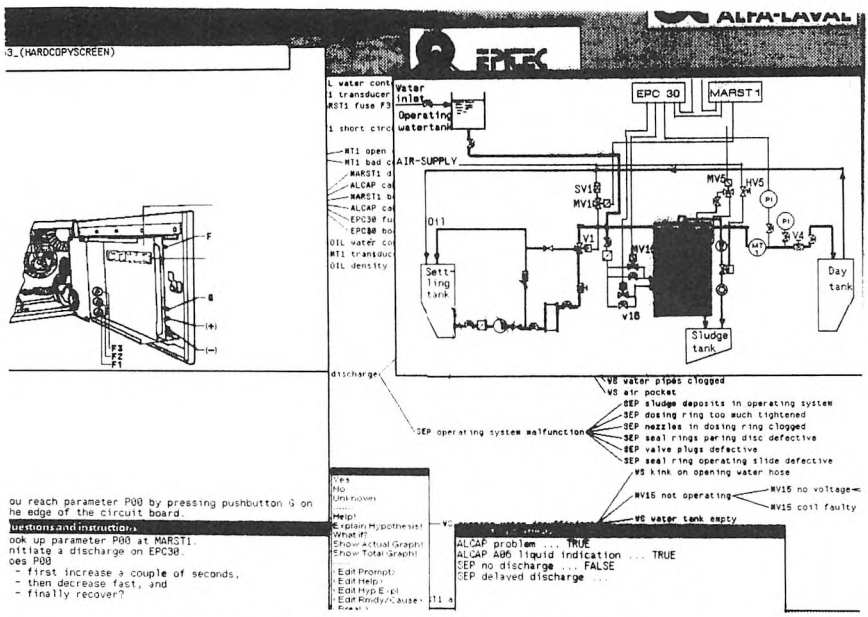


Figure 6.2. ***Graphics support for separator trouble shooting.***

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 or by scanning from a paper representation, and it will in the near future be possible also to enter such information directly from existing CAD systems.

Knowledge-base migration and expert systems architectures

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.

For instance, knowledge-based systems have as one of their main advantages that the knowledge base do act as a parameter structure, which specifies the problem solving behaviour of the program as governed by domain knowledge. By changing this domain knowledge, a system can be adapted to a slightly different application environment, i.e. be customized to a particular installation. We perceive this situation to be typical for many commercial knowledge systems, where a generic application is developed as a prototypical program, which is then supplemented with local and installation-specific information, before delivery to the customer. In this process it is reasonable to believe that differences in local hardware and software should also be catered for. Thus the only viable approach is to make the development tools as independent as possible of any particular delivery environment.

The first step in such a migration project has been undertaken for the Antibody Analysis Advisor (A^3) [Sandahl 1985]. 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 centres 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.

Intelligent human-computer interaction.

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

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

Important subjects for study in ASLAB are knowledge-based models of human-computer interaction, effective methods for producing explanations of system behaviour and results, and tutoring techniques for build-up and maintenance of user competence. Thus work on expert systems has e.g. clearly demonstrated the great practical value of even simple schemes for producing natural language presentations of facts and inference structures represented inside the system.

One area of particular interest to us is 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.

Initial applications will be taken from the area of technical descriptions and manuals. A typical situation is that directions for use employ examples to explain how a certain result should be achieved. However it is often the case that these examples differ in significant aspects from the user's situation, which easily causes misunderstandings or confusion. Likewise a typical manual refer at the same time to many different models of the equipment, which make them hard to read and understand. Thus the benefits of a system which can generate a customized example or description from its knowledge of the current equipment and the user's situation should be obvious.

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 now plan to develop a generalized approach, which tries to incorporate support for *knowledge-based training*, in particular of *emergency procedures*, in a hybrid expert systems development environment. 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. Initial experiments will continue previous work in the medical area.

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 (Hansson). 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.

Other aspects of systems development support are currently studied within the laboratory. Kevin Ryan, from Trinity College in Dublin, has been a guest researcher in ASLAB during 1985/86. He has conducted research in the context of the ESPRIT ToolUse project, which is concerned with the study of software engineering environments, and in particular with the possibilities to integrate tools supporting method-based software development. Dr Ryan's work here concentrated on the investigation of knowledge-based support tools

for a method-driven environment.

Here the software development process is viewed as a series of successive transformations Starting from a vague idea of user requirements the "specification" is successively transformed until an acceptable implementation is achieved. In this paradigm a "method" can be informally defined as a body of knowledge that guides the choice of transformation, while a "tool", in its simplest form, is a piece of software which assists in carrying out the chosen transformation. Many existing or proposed software engineering environments facilitate or even automate these individual transformations, but a method driven environment must, in addition, guide and inform the developer in the *choice* of applicable and appropriate transformations.

Problems studied involve support for requirements engineering and knowledge-based systems design. In an experimental implementation (Hansson), the possibilities to support systems design according to the JSD method using the KEE system on Xerox Lispmachines was investigated.

During 1986 Pål Sørgaard visited the group from Aarhus, with an interest in the studying of the relationship between a traditional information system development process and one where knowledge-based techniques are used. In particular he emphasizes the need for end user involvement and the development of evaluation standards, which recognizes the importance of assessing the performance of an expert system in an actual use situation.

The issues raised by Ryan and Sørgaard during 1986 are still considered of prime importance for research in the lab, although current staffing forces activities in this area to be concentrated on the iterative development and adaptive maintenance issues.

6.3.2 Statistical information systems.

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

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

The Statistician's Workstation.

Current efforts include the study of consultation systems for statistical analysis. The background is the well-known problem of understanding how to apply different tools for statistical analysis as correctly as possible on a given

data material. The broad availability of statistical library software as well as computer-stored information bases will significantly increase the danger of misuse or even making faulty conclusions due to a lacking understanding of the often intricate problems involved in the proper use and interpretation of statistical data.

The goal is to build an integrated environment to support the analysis and effective presentation of aggregated information. Subtasks involve quality control of available data, assistance for selection of appropriate statistical methods, for adjustment of data, and for preparing parameters for the corresponding analysis programs, support for interpretation of results and for tabular, graphical and verbal presentation of abstracted information. In preliminary studies a knowledge-based assistant for eliminating seasonal variations in statistical studies of industry and trade has been implemented with existing shells (SAGE on a PC and KEE on a Lisp machine).

As part of current activities an experimental implementation of a *statistician's workstation* is 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 illustrate 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 will be published as a licentiate thesis by Shamsul Chowdhury during the spring 1987.

An algebra of base operators for production of statistics.

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

A set of such *statistical base operators* is being defined in an international joint effort with participation of Statistics Sweden [Sundgren 1985]. In connection with that work, we are developing a theoretical framework in the form of an algebra, with the purpose of giving a firm basis for implementation and application of such operators as well as a conceptual integration with current practices in the field of relational databases. (Sundgren, Nilsson.)

6.4 External cooperation.

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

1. Department of Medical Informatics. Previous cooperation on advanced CAI systems (MEDICS) is now followed by joint work on medical expert systems (Gill, Reshagen, Timpka). Renewed joint activities in the area of knowledge-based training systems are starting spring 1987.
2. Alfa-Laval Automation. Joint work on expert systems for fault diagnosis and maintenance within the Knowledge Transfer Program (Rosenberg, Hanebring).
3. ASEA. Previous cooperation on a consultation system for robot configuration is now followed by Knowledge Transfer Program activities (Lozan-Villegas).
4. 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 (Hammar, Nordmark).
5. National Bureau of Statistics. Study of the design of statistical information systems. (Block, Nilsson, Wallgren and Wallgren. See also above.)
6. Nordic cooperation with Oslo (Kristen Nygaard) and Århus (Lars Mathiassen) in the SYDPOL programme (System Development Environments for Profession-Oriented Languages.) This programme is partly supported by Nordforsk and consists of national projects and four inter-nordic working groups, where Aslab participates in those concerned with systems developments methods and medical expert systems respectively.

6.5 Publications

External publications: For a full listing of published papers, including departmental reports, see appendix E. Below the last year's external publications by lab members are listed for an easy reference.

1. **Shamsul Chowdhury:** Expert System Aid in Statistical Analysis and Interpretation of Data. In *Proc. of the Society of Reliability Engineers*, Outaniemi, 1986.

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

3. **Sture Hägglund**, Redskap för expertsystem, i *Proc. NordDATA -86*, och i *Nordisk DATAnytt*, no 8, 1986.

4. **Sture Hägglund**, Kunskapsbaserade expertsystem i administrativa tillämpningar, i *Nordisk DATAnytt*, no 6, 1986.

5. **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.

6. **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.

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

8. **Kevin Ryan,** et al., Surveying Software Tools for a Method Driven Environment, *Proc IFIP-86*, Dublin, 1986.

9. **Kevin Ryan,** The Value of Mixed Metaphors in Computer Education, *Proc. Nat. Computer Education Conf.*, San Diego, 1986.

10. **Kristian Sandahl:** The Migration of Expert Systems into Production Environments. *Proc. Nord-Info Seminar on Knowledge Engineering*, Köpenhamn, 1986.

11. **Pål Sørgaard,** Evaluating Expert Systems Prototypes. Presented at *The 9th Scand. Sem. on Use and Development of Information Systems,* Båstad 1986.

ASLAB Memo series 1985 and 1986

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

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

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

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

85-05 **Doherty,** A Rule Interpreter for an EMYCIN-like Expert System Tool. Master's thesis, 1985.

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

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

86-01 **Hanson**, WATT - A Knowledge-Based Case-Oriented Teaching System. Master's thesis, 1985.

86-02 **Hägglund,** Datorstödda kunskapssystem i framtidens kontor. (Publiceras även som TELDOK-rapport, 1986.)

86-03 **Hägglund, Sandahl and Timpka,** Expert Systems in Medical Care.

86-04 **Sørgaard,** Evaluating Expert Systems Prototypes.

86-05 **Chowdhury:** Expert System Aid in Statistical Analysis and Interpretation of Data.

86-06 **Wallgren, B, Wallgren, A:** Företagets Informationssystem. Statistisk analys med företagets administrativa data.

7.

CADLAB
The Laboratory for Computer-Aided Design of Digital Systems

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

Krzysztof Kuchcinski

7.1 Introduction

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

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

The first years of VLSI Design Center were devoted to building competence and acquiring basic software. The year 83/84 marked the transition from the building phase to initiating new research. The "fruits" of these early years are now being harvested and CADLAB is able to report on some significant

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

progress made during 1986. This progress will be highlighted in a later section. CADLAB is broadly concerned with many aspects of the problem of silicon compilation; the process of translating a high level description of a system to a silicon layout. One model of the silicon compiler is that of a translator which takes a high level description of a chip and transforms the semantic content of the description into a machine as indicated in Fig. 7.1.

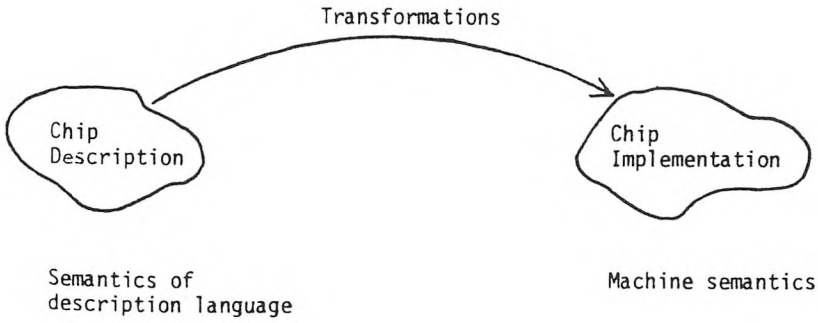


Figure 7.1. *Silicon Compiler Model.*

In the framework of the silicon compilation model, our research activities concentrate mainly on different tools which constitute a complete silicon compilation environment. They include the development of methods, algorithms as well as integrated tools. As a result of this work, we have implemented, among others, the ASL language and the CAMAD system.

7.2 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 a logic level. It provides a possibility to choose a better design at an early stage of the design process. Second, it yields special purpose systems (rather than general purpose systems) which are embedded in a wide variety of products such as telecommunication systems, electronic and biological instruments, robots and automatic control systems. For the 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 ASAP project.

7.3 Asynchronous Architectures

In the ASAP 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 .5mm as it does to go along a coaxial cable 75cm 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.
- The approach, where timing is handled by the asynchronous strategy, provides a basis to design *systems* rather than single chips. To find "optimal" system solution one may also include into CAD tools a hardware-software trade-offs. CAD tools must provide, in such case, various forms of synthesis and support the evaluation of the designs in order to compare the performance. This helps a designer to investigate the design space.

We have now identified the major advantages of using asynchronous architecture approach at every level- 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 proposed ASAP architecture benefits there must exist, however, CAD systems supported this approach. The CAD systems must conveniently provide many different tools that utilize an efficient graphical man-machine interface as well as design database which possess a knowledge about various design views. The CAD systems should contain many different tools like synthesizers, simulators, verifiers and also more specialized like testability rule checkers and enforcers.

7.4 Ongoing ASAP Projects

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

System Specification

Specification, Synthesis and Analysis

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

Simulation

As part of the overall goal of studying architectures and silicon compilers, this project aims at investigating different simulation techniques and different architectures. We started from detailed definition of the main components of the ASAP architectural strategy. A register level simulator for the family of this architectures has been produced. Then, based on the knowledge gained during realization of the project, the simulator for systolic arrays has been proposed. Finally, a simulation study of different binary tree structures has been done. As a result, the performance evaluation comparison of different tree structures has been obtained. This project includes also a study of the programming language Occam and its usability for this type of application.

Synthesis of Behavioral Descriptions

The CAMAD project aims at the development of a formal design methodology and an integrated set of automatic as well as computer aided design tools for digital VLSI systems. We are particularly interested in the synthesis of VLSI systems from their high level behavioral descriptions. Four problems of this synthesis process have been identified; first the automatic generation of VLSI implementation structures from a behavioral description which specifies only what the system should be able to do. The second problem is how to partition the implementation structure into a set of system modules so that each module can be implemented independently. The third problem is the optimization of the system implementation in terms of cost and performance. Finally, we have also studied the problem of how to automatically generate microprograms to implement the control structures of VLSI circuits.

To address these four synthesis problems, a unified design representation model, the extended timed Petri net (ETPN), has been developed. This design representation consists of separate but related models of control and data part. This separation makes it possible to represent both structures and behaviors of VLSI systems, thus the generation of implementation structures from their high level behavioral descriptions can be done in an iterative way. The ETPN allows also formal manipulation of the design space and different optimization trade-offs between performance and cost. Partitioning of systems into submodules can be provided both on the data part and on the control part, which produces a set of pairs of corresponding data subparts and control subparts. As such, asynchronous operation of the designed systems as well as physical distribution of the modules is possible. The use of such a formal representation model also leads to the effective use of CAD and automatic tools in the synthesis process and the possibility of verifying some aspects of a design before it is completed. An integrated design environment, the CAMAD design aid system, has been developed based on the ETPN model.

Pipeline Extraction

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 tries to find a way of describing a system so that extraction of "pipelinable" parts can be done automatically. 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, while the second simplifies description of the state of the computations. Also, formalisms for process definition in this context need to be developed.

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.

Representation and Reasoning

To support the ASAP architecture approach of VLSI System Design, knowledge and design representation should be such that architectural decisions may be reasoned about both manually and automatically. The Representation and Reasoning subproject of ASAP aims at embedding the ASAP methodology into AI based tools, such as Design Assistants (Expert Systems), on different levels and phases in the design project.

One particular problem is the representation of the design itself, the design task, and the design knowledge. A number of international research groups are very active in this area, such as the PALLADIO design environment at Stanford University, COMPUTER, Dec. 1983, and the Knowledge-based Design System at Carnegie-Mellon University. Through the use of design representation in the form of dependency networks, it is possible to reason about the interaction between subparts and their interrelation so that an optimal partitioning is possible. Thus support can be provided to the partitioning task in the Synthesis of Behavioral Descriptions subproject of ASAP.

Reusability is an important issues related to designing complex system. To be able to reuse design or parts of design is a natural topic to explore. The suggested method of reusablity is to use partial evaluation, e.g. extract subparts of a previous design and minimize these according to their usage within the new design.

Methods for reusal may be found within other parts of Computer Science and as a software technique is referred to as Program Transformation. Within IDA several research efforts have been related to this area. The work done thus far has resulted in two doctoral dissertations, a thesis by Anders Haraldsson, entitled "A Program Manipulation System based on Partial Evaluation" and a thesis by Pär Emanualsson entitled "Performance enhancement in a well-structured pattern matcher through Partial Evaluation".

Now that we have considered the general course of research activities, let us highlight the progress made during 1986.

7.5 Progress During 1986

The CADLAB group made several important advances on the ASAP project during 1986. This has resulted in the completion of another licentiate thesis. Further, many papers have been published and presented at the international and nordic conferences.

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

Via the licentiate of Tony Larsson, we have a language for a specification of asynchronous architectures. The language and a proposed set of calculus-hiding- binding and event reduction-rules forms a basis for further research in the field of verification and synthesis CAD tools. The highlights of this work are presented in section 7.6.

In the area of silicon compilation, the work performed to develop the CAD system called CAMAD (Computer Aided Modelling, Analysis, and Design of VLSI Systems) shows a significant progress. Based on the formal model of the VLSI circuits the optimization, partitioning and synthesis algorithms have been developed and implemented. The work has been described in 6 papers published on the international and nordic conferences. The highlights of this work are presented in section 7.7.

In the area of simulation and ASAP architecture definition, we investigated different methods of simulation using Occam language. This work resulted in three publications this year. The highlights of this work are presented in section 7.8.

7.6 On the Specification and Verification of VLSI Systems

Increasing complexity of VLSI systems and the requirements for more or less automated analysis and synthesis tools have made it essential to improve system specification and verification methods. The design of large systems composed of subsystems, designed by different design teams, requires that these subsystems can communicate with each other. This requires that each subsystem has well specified and verified communication interfaces.

In the thesis, a combination of issues related to the *specification* and *verification* of (VLSI) hardware systems are treated. Traditional hardware description languages (HDL:s) are considered too feature rich to keep the number of transformation rules reasonable as a media for verification. Hence, a small but general specification language, ASL, is proposed as an experimental

tool for specification, verification, and synthesis of system architectures. The language supports specification of a systems behaviour and structural decomposition. A second problem with most traditional hardware description languages are that their semantics are at best described informally in a programmers manual or are embedded in related tools such as simulators. This provides a motive to focus on semantic issues, especially the *actional behaviour* and related transformation rules.

ASL is intended to be used as input to automatic synthesis, analysis, and verification tools. This requires that the language has a well defined semantics and that a set of rules are defined such that the specification and the implementation can be compared and shown equal in a formal sense. 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 (or reducing) syntactic transformations which enable deductive 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 and exhaustive verification techniques are made tractable.

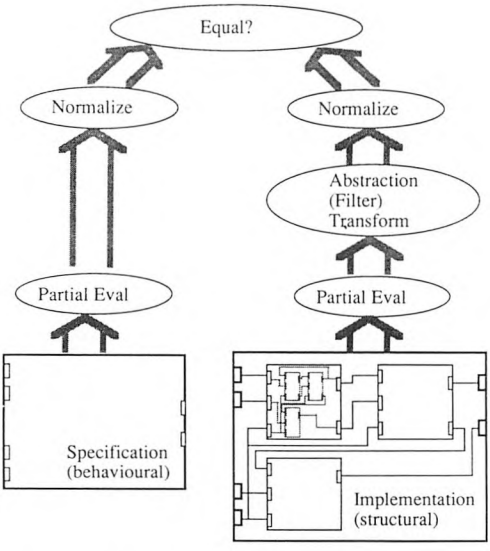


Figure 7.2. *Verification Utilizing Calculus, Internal Event and Port Reduction Rules.*

ASL, is designed to support specification of a system viewed as a module that can provide a set of functions to an external environment. To gain access to these functions, the module and its environment must agree upon a communication scheme. In a communication scheme, the ordering of a set of actions is described. These actions performed by agents, specify the interchange of information between the module and its environment. The

module communicates via ports and has no knowledge about the environment where it may be initialized. This implies that the actions performed are viewed as driven by events that are visual in the form of changes at the module ports. A port can be viewed as a connector to a net (or a set of nets) connecting a set of communicating modules. Events can have both global and local triggering effect, hence ASL can support design of both synchronous and asynchronous systems at different levels of abstraction. The encapsulation facilities of ASL provides for reusability of functions, interfaces (communication actions related to ports) as well as complete modules.

7.7 Synthesis of Behavioral Descriptions

The major results of the CAMAD project have been published in 6 papers. Here we will describe some most important results, which have been published in international journals and international conference proceedings.

In the first paper [4], we gives the formal definition of the extended timed Petri net (ETPN) model which consists of separate but related models of control and data parts. The application of this model for the description and synthesis of VLSI systems with asynchronous processing strategy is then discussed. The use of the ETPN model in the VLSI design process leads to the formalization of the control/data path allocation and module partitioning problem as an optimization problem. To solve this optimization problem, a set of design space exploration strategies and heuristic algorithms are then proposed.

The application of this design representation also leads to the effective use of CAD and automatic tools in the synthesis process. As a result, the CAMAD design aid system was developed based on the ETPN design representation and its synthesis methodology. Input to CAMAD is a high level behavioral specification which specifies what a system should be able to do without prescribing the physical structure of the implementation. This specification is first translated into the ETPN design representation. The ETPN representation is then analyzed, transformed, and finally partitioned into a set of processing modules which can be implemented independently. The major synthesis task in CAMAD is carried out by a set of semantics-preserving transformation algorithms which move a design step by step from an initial state to better ones and finally to an optimal or near optimal one. The overview of CAMAD is presented together with its major characteristics in the second paper [5].

In the following paper [6], the problem of how to automate the design of control structures for VLSI systems is addressed. We have developed design tools which can synthesize a control structure from a high level specification into microprograms as well as perform optimization based on some implementation constraints. The entry to such tools is a timed Petri nets with

restricted transition firing rules. This timed Petri net model is used to produce sequences of control signals to evoke operations of their associated data parts as defined in the ETPN design representation model. A set of algorithms for the control synthesis have been implemented and integrated into the CAMAD design aid system.

Finally in [7] we describe a systematic way to design asynchronous concurrent systems. Usually, it is very difficult for human designers to design such systems. One of the reasons is that such systems are inherently nondeterministic; therefore, it is almost impossible for designers to keep track of all activities of the designed system. Our approach to addressing this complicated problem is to start the design process with a behavioral specification of the system where the designers concentrate only on the desired system semantics. The designers can then utilize some design tools to exploit the design space so as to create an appropriate implementation structure or have the design system attempt to automatically provide an optimized solution based on a library of functional implementations. A general partitioning algorithm used to partition a VLSI description into a set of processing modules whose actions are coordinated to implement the specified behaviors is then proposed.

7.8 Simulation

The work on the simulation techniques applied for different architectures spread into three research subjects. In each subject, we used the programming language Occam as a basic simulation tool. The results of this work have been published as a three papers and an internal report.

A short description of the Occam language has been provided in the paper [1]. It summarizes some experiences (issues of design, testing and maintaining concurrent programs) gained during the ASAP simulator implementation in this language. The paper thus describes a register level simulator for a family of architectures based on asynchronous processes. Within this architecture we hope to avoid the usual bottlenecks of Von Neuman machines and at the same time avoid the problem of dynamically binding every operation as in dataflow machines.

Systolic architectures have proven to be cost effective and high-performance solutions to a variety of problems in, for example, signal and image processing. Usually, the development of a systolic solution to a problem is divided into three major steps: requirements definition, design and implementation. The design phase often consists of an ad hoc choice from a family of possible systolic solutions. To allow the designer to study and evaluate various trade-offs in different designs, we propose to use a high- level simulation package tailored for systolic architectures. The system, described in [2], will be

used for teaching under-graduate students basic design principles as well as for developing contemporary designs.

The simulation and performance evaluation of different tree computer architectures has been studied in the framework of the project. The internal report [3] summarizes some results obtained from this research work. The binary, completely linked, half ring, and full ring trees have been compared in respect to their relative performance for a random message exchange as well as for implementations of algorithms for Heapsort and adaptive numerical integration. For the completely linked tree, a foult-tolerant routing algorithm is tested.

7.9 Cooperation With Other Groups

CADLAB has cooperated with the Computer Systems Laboratory at Uppsala as well as with Piotr Dembinski in Gothenburg in applying the formal design techniques developed for description of control and processing structures and communications protocols to the description of integrated circuits. The CADLAB group, in November, presented the ASAP project for Microelectronic Center. Ideas presented during this meeting seems to be of great interest to the developed projects. Further, we have had contacts with Gunnar Carlstedt of HYLAB AB concerning the exchange of ideas for VLSI design. We are in the process of defining a closer cooperation with Dr. Gunnar Carlstedt; particularly in the relationship to his work on architecture and CAD tools sponsored by SICS (The Swedish Institute of Computer Science).

7.10 Industrial Significance

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

7.11 Other Related Activities

As a result of Professor Lawsons assistance to the Prime Minister of Malaysia Dr. Mahathir in planing a National Microelectronic Programme, a new institute MIMOS (Malaysian Institute of Microelectronic Systems) was established and inaugurated during 1985. Further contacts between MIMOS and various Swedish Institutes and the Ericsson Corporation have been pursued. MIMOS as of December 1986 has 48 employees. They have utilized Swedish CAD software from the Microwave Institute and Linköping University. Further, they completed 5 integrated circuits designs that were processed by NORDCHIP. We shall consider to accept a doctoral student from MIMOS during 1987.

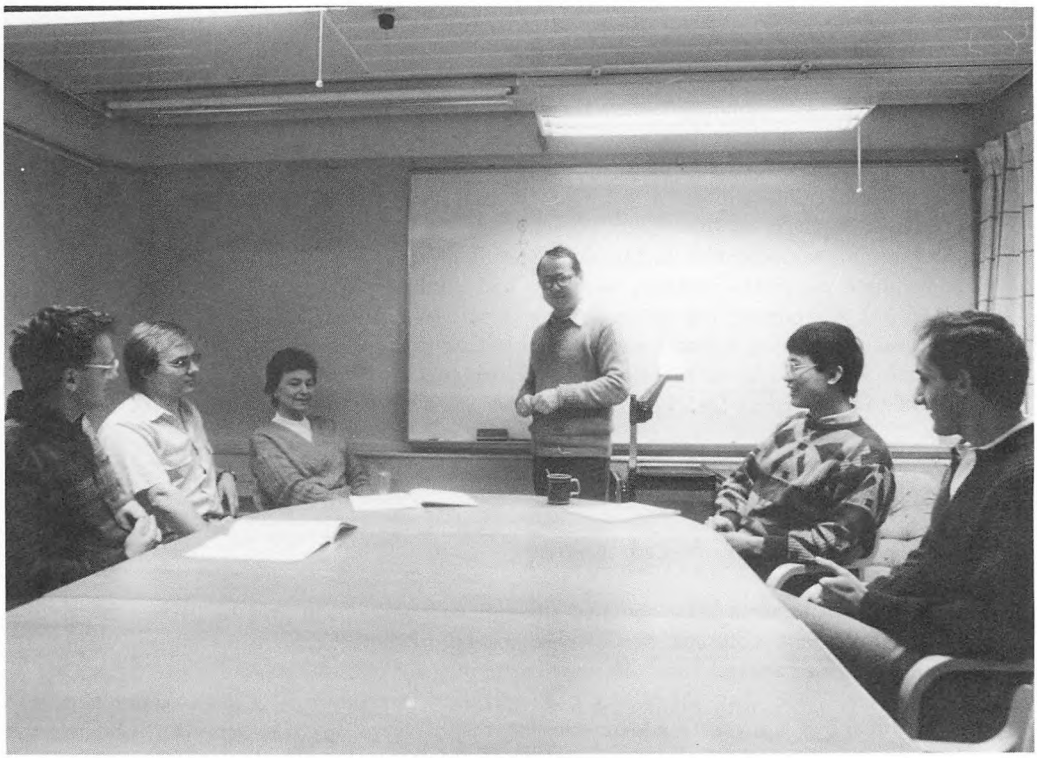


Figure 7.3. *CADLAB meeting with Tony Larsson, Björn Fjellborg, Britt-Marie Ahlenbäck, Krzysztof Kuchcinski, Zebo Peng and Naïl Kavak.*

7.12 Personnel

Professor Harold W. Lawson Jr., Ph.D.
Krzysztof Kuchcinski, Ph.D. (from September)
Britt-Marie Ahlenbäck, secr.
Björn Fjellborg, MSE
Nail Kavak, MSE
Tony Larsson, Tech.Lic
Mikael Patel, Tech.Lic.
Zebo Peng, Tech.Lic.

Professor Harold Lawson has been acting laboratory leader until September when Dr. Krzysztof Kuchcinski joined CADLAB. During this period Mikael Patel actively participated in the leadership of CADLAB. We expect that 2 new doctoral students will be added to the group during 1987.

7.13 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.

7.14 References

The following are the CADLAB publications for the year 1986 that are referenced in the text. For the full list of publications, please refer to the appendix.

1. J.Fagerström, *Experiences with Occam: A Simulator for Asynchronous Processes*, Proc. 19th Hawaii Int. Conf. on System Sciences, Hawaii, Jan.

1986, pp.95-102

2. J.Fagerström and M.Patel, *High-level Simulation of Systolic Architectures*, The First International Workshops on Systolic Architectures Oxford, 2-4 july 1986

3. B. Fjellborg, *A Simulation Study of Four Binary Tree Structures*, Report LiTH-IDA-R-86-19.

4. Z. Peng, *A Formal Approach to the Synthesis of VLSI Systems From Their Behavioral Descriptions*, Proc. 19th Annu. Hawaii Int. Symp. on System Sciences, Hawaii, Jan. 1986, pp.160-167

5. Z. Peng, *Synthesis of VLSI Systems with the CAMAD Design Aid*, Proc. 23rd ACM/IEEE Design Automation Conf., Las Vegas, Jun. 1986, pp.278-284

6. Z. Peng and K. Kuchcinski, *Synthesis of Control Structures From Petri Net Descriptions*, Microprocessing and Microprogramming, Vol.18, Nrs.1-5, 1986, pp.335-340

7. Z. Peng, *Construction of Asynchronous Concurrent Systems From Their Behavioral Specifications*, Proc. 10th World Computer Congress, Dublin, Ireland, Sept. 1986, pp.859-864

8.

LIBLAB

The Library and Information Science Research Laboratory

Roland Hjerppe

8.1 Introduction.

LIBLAB, a joint project of the Department of Computer and Information Science and the University Library, studies methods for access to documents in collections and the information contained in them. In the research program for LIBLAB are specified two major themes: Document Description and Representation, and Users and Library (Systems), and one minor: Networking, especially questions of central vs. local handling. For each of the themes there are also two subthemes defined.

The interest in document description and respresentation arises from the observation that all other types of improvements in systems for access to documents (catalogs or information retrieval systems) are of no avail if the basic data in the database do not adequately describe the document and its contents or if the representation hampers or prohibits the manipulations desired.

Appropriate data in a congruous database is, however, not enough. If flexible and convenient means for access to the information are lacking then the usage of the system will cease and information sought will never be found. Hence the interest in Users and Library (Systems).

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

No single source, system or service can alone cater to all the needs and demands of its users. The documents needed by an individual user are usually accessible in collections which mostly are arranged so that their size and the distance to them increases as the usage by the individual decreases. Documents needed often or regularly are available in the room, those needed less frequently are available in the room or department, those needed occasionally are available in the local library and those needed once or very seldom are available locally or in other libraries. This structure, which can easily be recognized once it exists, is troublesome to implement or cultivate as part of a plan since it is very difficult to predict the usage of an individual item. Problems of relations between and access to other collections, of networking, are therefore also a concern.

All of the themes and their subthemes were looked into during the first three years, 1983-1985, but the concentration has been 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 in which problem areas can be found that relate to all the themes, becoming the main focus of the activities of LIBLAB.

In 1986 the activities in LIBLAB have hence mostly been related to HYPERCATalog or been a continuation of earlier work on knowledge based systems and document description.

8.2 Project HYPERCATalog

The fundamental paradox of information retrieval:

The need to describe that which you do not know in order to find it

is usually solved by attempting to circumscribe the unknown by trying to describe that which is known. All operational systems are based on this approach.

A different solution, based on "I can't describe it, but I recognize it when I see it", and "I want more like this (but slightly modified)", which takes into consideration our perceptual and cognitive capabilities, is one of the bases for the HYPERCATalog.

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

- 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.

The main components of HYPERCATalog are depicted in Fig. 8.1 below.

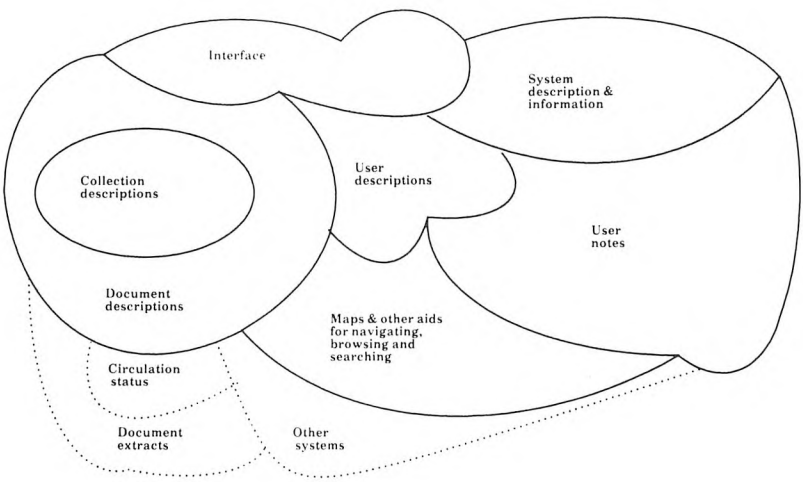


Figure 8.1. *HYPERCATalog components*

The HYPERCATAlog-project is planned to proceed through the following phases:

- Generation of design specification, functions and components
- Identification of problems inherent in specification
- Search for, and evaluation of, solutions to specific problems
- Elaboration of blueprint for construction
- Construction of a prototype

- Implementation and use
- Modifications

In the original plans the specification phase was to be concluded in 1986, with a design specification as a result. A full specification has, however, not been produced although the papers published are important parts of this specification. Some of the main reasons for the delay are i.a.

- questions on the form of the specification and its level of detail,
- the desire to gain experience of hypertexts through the usage of NoteCards, an application developed by Xerox for handling one type of hypertexts,
- the delay in the installation of the Xerox workstations that were received through their University Grant Program,
- more extensive studies of single components than originally planned.

Another important factor was the fact that inherent (and recognised from the beginning) even in the early visions of HYPERCATalog was also the need for an accompanying research program to study a number of problems and questions arising from the central conceptions. Some of these were listed in the previous Annual Research Report.

During 1986 preliminary work has been done on the concepts of filters, providing different views of the contents of the database and three fundamental types of views were identified, see Fig. 8.2 below (taken from the paper "HYPERCATalog and Three Meta-Schemata for Database Views: Knowledge Organizing, Collection Derived, and User Established Structures"),

The ambition in HYPERCATalog is to be able to cater to a very wide range of users, from first time library users, with perhaps no experience of libraries and catalogs, nor computers, to the researcher who uses the library on a daily basis. For all of these users are needed means for building mental models of the system, and these mental models have to be initially simple as well as extensible as experience and demands grow. A conscious use and provision of metaphors has been recognised as a potential, partial solution. An investigation of metaphors was thus initiated and a first result presented by Arja Vainio-Larsson in the paper "Metaphors as Communicators of Conceptual Ideas"

Preparatory work has also been done during 1986 on approaches to the handling of classification schedules as databases in their own right and on the the generation of "maps" from classification schedules.

The present outlook for the design specification phase is to have it conluded during 1987. The delays are expected to be more than compensated by the experiences gained internally and from work on Hypertexts at other places.

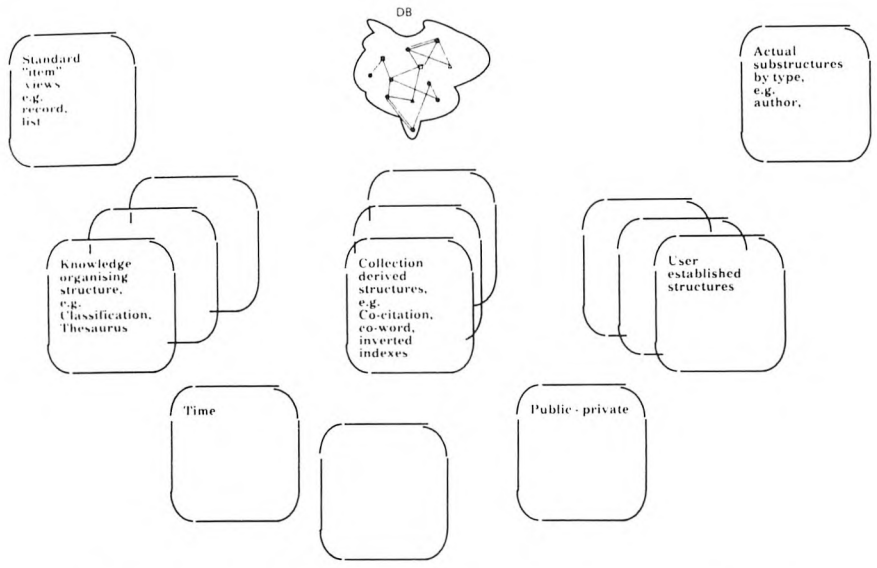


Figure 8.2. *The three fundamental views of databases, and additional filters .*

8.3 Cataloging and document description.

During the first months of 1986 a survey article titled "Electronic Publishing: Writing Machines and Machine Writings", was prepared by Roland Hjerppe for v. 21 of "Annual Review of Information Science and Technology". This survey is planned as the introduction to a discussion of the effects of modern information technology on 1) the change of and interaction between the concepts document/work/text, and 2) the concomitant impact on form, style and mode of literary expression.

Previous experiments with expert-systems for cataloging were written up for publication and a paper by Roland Hjerppe and Birgitta Olander titled "Cataloging and Expert Systems: AACR2 as Knowledge Base" has been accepted for publication during 1987 in Journal of the American Society for Information Science. A presentation of the project was made by Birgitta Olander at the session on "AI and Bibliographic Control" on 1968 Annual Meeting of the American Society for Information Science in Chicago in October 1986.

The mapping and graphic display of the structure of the set of rules in "Anglo-American Cataloging Rules. Second Edition" that was initiated (but not finalized) in 1985 - as a part of the ESSCAPE-project - was taken up again in November and concluded by the end of the year.

Preparatory work on formalization of bibliographic description and catalogs, using ideas and formalism from SGML, Standard Generalized Markup Languages and abstract editors, has also been done and the FORMEX (Formalised Exchange of Electronic Publications) document from the New Technologies Project Management of the Office for Official Publications of the European Communities is indicating an interesting possibility for Editing Systems with Cataloging Helper And Tagging OperatorS.

8.3.1 Other projects and activities

Manny Jägerfeld, one of LIBLAB's doctoral students, has during the latter half of 1986 participated in a pre-study for WHO on the Establishment of a European Clearinghouse on Assessment of Health Technology in Linköping.

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. One of the goals in this project is first to design and provide three different approaches to decision support: Knowledge based systems, Hypertexts and Bibliographic databases, and then after testing prototypes attempt to integrate the approaches in one system. In Fig. 8.3 below is a typical screen display from LIMCONS, one of the decision support modules.

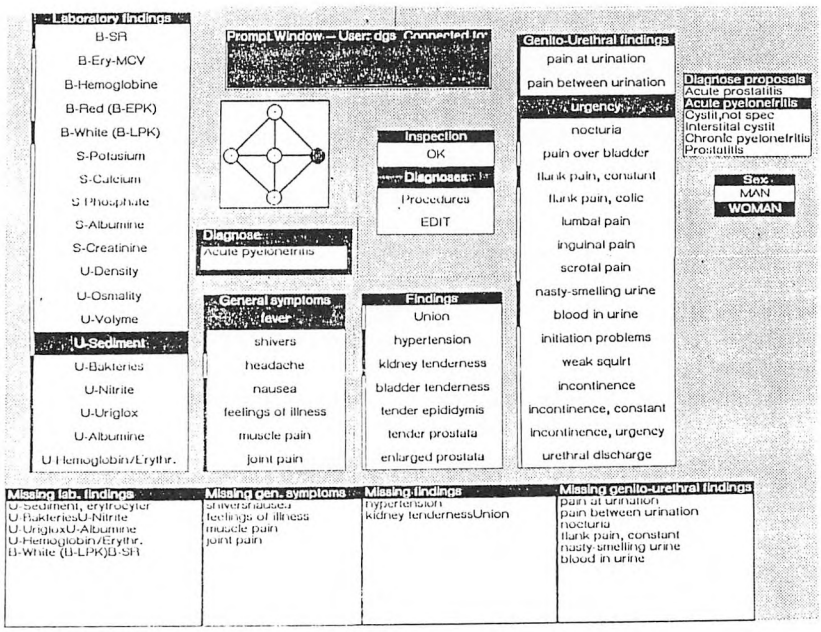


Figure 8.3. *LIMCONS interface*

During 1986 two papers were published that describe this work:

"Decision Support for General Practitioners; Design and Implementation by Integrating Paradigms: Hypertext, Knowledge Based Systems and Online Library." and
"The Need for Supplements to Traditional Expert Systems: Lessons from Designing Knowledge Based Systems for Primary Care."

Birgitta Olander and Roland Hjerppe participated in a task force plannning the development of an information network for primary health care, and Birgitta Olander, acting as the secretary wrote the final report, "HUGIN - Health University General Interactive Network". (The work was performed as a part of establishing a medical education program in Linköping that is called the Health University to indicate its commitment to health rather than disease, and that integrates education for people at various levels in medical and health care.)

8.4 Personnel

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

Roland Hjerppe, MSc, Laboratory leader, spends, apart from planning, coordination and administration etc., most of the time on the HYPERCATalog project.

Birgitta Olander, BA, MLS, Systems Librarian at Lund University, former head of the acquisitions department at Linköping University Library. BO is also pursuing doctorate studies at University of Toronto, Faculty of Library and Information Science and spent the summer of 1985 with LIBLAB in the HYPERCATalog project, before going back to finish her courses in Toronto during 1986.

Arja Vainio-Larsson, MA, former lecturer in psychology, began her doctoral studies at LIBLAB the fall of 1984. The main area of interest is the design of user interfaces. AVL has studied the use of metaphors and is now i.a. investigating the confluence of object oriented approaches with metaphors.

Lisbeth Björklund, B.Sc., library assistant in the interlending department of Linköping University Library, began her doctorate studies at LIBLAB the fall of 1985 and has mostly been taking graduate courses, i.a. in knowledge engineering.

Manny Jägerfeld, BA, who has had a research scholarship from DFI for studying computerization in libraries, has also started his doctorate studies at LIBLAB the fall of 1985 and was during the lattter half of 1986 engaged in the WHO-project on assessment of medical technology, focusing on the database

and information systems aspects.

Siv Söderlund who replaced **Anne-Marie Jacobson** from July 1986 as part time secretary for LIBLAB.

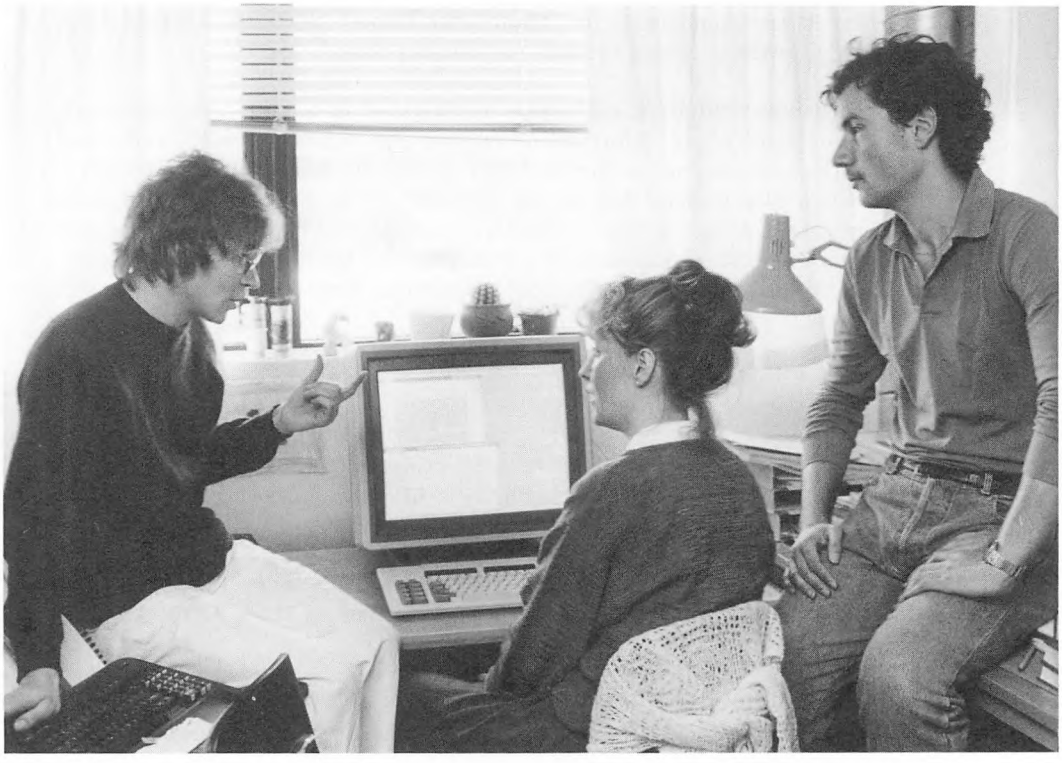


Figure 8.4. *Project discussion with Manny Jägerfeld, Arja Vainio-Larsson and Toomas Timpka.*

Associated people:

The following have various associations to LIBLAB:

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

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.

8.5 List of publications

Reports:

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

LiU-LIBLAB-R-1986:1
Hjerppe, R.: Electronic Publishing: Writing Machines and Machine Writings. The impact of computers on text. Mars 1986, 28+21p (Edited version published in "Annual Review of Information Science and Technology." vol. 21 1986. M. Williams. Ed. Knowledge Industry Publications Inc. pp.123-166)

LiU-LIBLAB-R-1986:2
Hjerppe, R.: (Reflections (Education (Library, Information, Computer (Science (Research))))) Mars 1986, 8p (Discussionstarter for the Anglo-Nordic Research Seminar "Training the Information Researcher for the Future", Lidingö, 9-11 April 1986)

LiU-LIBLAB-R-1986:3
Hjerppe, R.: HYPERCATalog and Three Meta-Schemata for Database Views: Knowledge Organizing, Collection Derived, and User Established Structures. (Published in "Online Public Access to Library Files: Second National Conference." J Kinsella Ed. Elsevier)

LiU-LIBLAB-R-1986:5
Malmberg, I-M.; Östberg, B-M.: LINS - LIBLABs Namnhantering-System. Testning och utvärdering. Maj 1986, 17p. (Also available as Specialarbete nr 108, 1985, Institutionen Bibliotekshögskolan, Högskolan i Borås)

LiU-LIBLAB-R-1986:6
Vainio-Larsson, A.: Metaphors as Communicators of Conceptual Ideas. August 1986, 14p. (Paper presented at "Ninth Scandinavian Research Seminar on Use and Development of Information Systems", Båstad, 19-22 August 1986)

Working papers:

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

1. *LiU-LIBLA-WP:33* Hultman, J.; Idberger, K.; Johansson, M.; Sisk, P.: Systemdokumentation. LINS: LIBLABs NamnhanteringsSystem. May 1986, 74+238p.
2. *LiU-LIBLA-WP:34* Olander, B.: Notes on Notecards. Augusti 1986, 5p.
3. *LiU-LIBLA-WP:35* Hjerppe, R.: Utkastet "DFIs VERKSAMHET - EN UTVÄRDERING" (Dnr 100/86-5) från Statskontoret, några synpunkter och kommentarer. September 1986, 6p.
4. *LiU-LIBLA-WP:36* Hjerppe, R.: A review of the dissertation "A Bibliometric Analysis and Evaluation of Research Writings by Indian Authors in the Field of Physics" by Paul Mohan Roy. Oktober 1986,

13+11p.

5. *LiU-LIBLA-WP:37* Hjerppe, R.: Informationssystem i kontor. Kursinformation. Oktober 1986, 3p.

6. *LiU-LIBLA-WP:38* Hjerppe, R.: LIBLAB. Verksamheten under 1986 och plan för 1987. Oktober 1986, 8+5p.

External papers:

(i.e. reports in other series or co-authored with people from other institutions)

1. (Olander, B.:) HUGIN (HälsoUniversitetets Generall Interaktiva Nätverk) - ett förslag till stöd för informationsförsörjningen vid hälsouniversitetet. Rapport för etapp 1 från Arbetsgruppen för delprojektet Kunskapsuppbyggnad och kunskapsspridning, Hälsouniversitetet i Östergötland. HälsoUniversitetet, rapport 1986:2, 32+2p.

2. Timpka, T.; Strömberg, D.; Möller, I.; Gill, H.; Bjurulf, P.,; Mattson, P. and Wigertz, O.: The Need for Supplements to Traditional Expert Systems: Lessons from Designing Knowledge Based Systems for Primary Care. Jan. 1986, 10+4p. (Paper presented at the conference "Expert Systems and Applications" in Avignon, April 28-30 1986)

3. Timpka, T.; Strömberg, D.; Möller, I.; Bjurulf, P.; Gill, H.; Mattson, P.; Olander, B. and Wigertz, O.: Decision Support for General Practitioners: Design and Implementation by Integrating Paradigms: Hypertext, Knowledge Based Systems and Online Library. (Paper presented at MEDINFO 86, 5th World Congress on Medical Informatics, in Washington, D.C. October 26-30 1986)

9.

LOGPRO
The Logic Programming Laboratory

Jan Maluszynski

9.1 Introduction

The Laboratory for Logic Programming was created in spring 1985 as a result of division of the former Group for Theoretical Computer Science into two independent research groups. (Until November 1986 LOGPRO had the status of a research group). The research concentrates on the foundations of logic programming systems and on the relation of logic programming to other computational paradigms.

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

9.2 Personnel and External Researchers

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

Jan Maluszynski, Ph.D., professor, *group leader*
Douglas Busch, Ph. D. *visiting researcher*
Wlodzimierz Drabent, Ph. D. *visiting researcher*
Staffan Bonnier, *graduate student*
Håkan Jakobsson, *graduate student* (at Stanford since Sept. 1986)
Simin Nadjm-Tehrani, *graduate student*
Ulf Nilsson, *graduate student*

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.

Some of the research was done in external cooperation with:

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

The main research activity concentrated around the project *"Research in Efficiency of Logic Programming"* funded by the National Swedish Board for Technical Development (grants STU-F 85-3166 and STU 86-3372).

9.3 Research Activities

9.3.1 The Background

We are searching for concepts that make it possible to improve efficiency of execution of logic programs and to facilitate logic programming. For this we study relationships between logic programming and other programming paradigms.

Research in 1986 was based on our previous results:

- a study of two-level grammars as a logic programming language: (Maluszynski,J., Towards a programming language based on the notion of two-level grammar, *Theoretical Computer Science* 28 (1984), 13-43);
- a formal comparison of logic programs and attribute grammars (Deransart,P. and Maluszynski,J.: Relating logic programs and attribute grammars *Journal of Logic Programming* 3, No. 2 (1985) 119-158).

The most important differences between logic programs and attribute grammars are the following:

- Attribute grammars use a many-sorted type discipline, while no concept of type is introduced in (traditional) logic programming.
- Attribute grammars refer to the concept of term evaluation, while the operational semantics of logic programs is based on resolution.
- Attribute evaluation process is conceptually separated from the parsing process that creates the unlabeled skeleton parse tree to be decorated, while the proof trees of logic programs are constructed together with their labels and no concept of unlabelled skeleton tree exists.

It is an interesting question whether the features of attribute grammars which distinguish them from logic programs possibly could be used in logic programming.

As a result of this research we obtained:

- a notion of data dependencies in logic programs;

- a notion of type for logic programs;
- an alternative view of Definite Clause Grammars.

9.3.2 The Results

The results obtained in 1986 are partially based on these notions. They are presented in the following papers:

Nilsson,U., AID: An alternative implementation of DCG's, *New Generation Computing* 4 (1986)

This paper describes an experimental implementation of Definite Clause Grammars based on conceptual separation of parsing and decorating of the context-free parse tree. This makes it possible to avoid unnecessary backtracking and to accept left-recursive grammatical rules. The implementation is done in DEC10 Prolog. It uses the SLR(1) parsing technique but it accepts any type of grammar. The conflicts in the parse table occurring for non-SLR(1) grammars are automatically and correctly handled by Prolog backtracking mechanism. In spite of its prototype nature the system is faster, at least on some examples, than the original implementation of DCG's provided by the Prolog system.

Drabent,W., and Maluszynski,J.: Inductive assertion method for logic programs, *Proc. of TAPSOFT 87* Pisa , March 1987

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.

In this paper, the inductive assertion method for proving partial correctness of logic programs is introduced and proved sound. 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.

Komorowski,J. and Maluszynski,J. Logic Programming and Rapid Prototyping, Report TR-01-86, Harvard University, Aiken Computation Lab. and LiTH-IDA-R-86-20 (to appear in *Science of Computer Programming*)

The paper discusses the usefulness of types and data-flow declarations in the process of systematic development of a logic program from an intuitive description of the problem. It outlines a methodology of logic programming based on these concepts. The methodology is introduced by means of an

example larger than those usually used to illustrate the advantages of logic programming. Starting with an informal specification of a structure-editor, it is shown how to formalize it into a directly executable prototype. The paper also introduces guidelines for validating logic programming code as implemented in Prolog.

9.3.3 Other Research

The group is currently engaged in the following activities:

- development of an improved model of AND/OR-parallel execution of logic programs (U. Nilsson). This is a continuation of the previous work. Some preliminary ideas are included in the examination thesis of H. Jakobsson. The aim of the work is to develop an AND/OR-parallel execution model without backtracking, based on a clear concept of data dependency. We plan to continue this work in cooperation with the Swedish Institute of Computer Science.
- experimental analysis of programming techniques used in logic programming (W.Drabent). 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. A draft paper is prepared for publication.

9.3.4 Future Research

The aim of our future research is to contribute to methodology of logic programming, both theoretically and by creating tools supporting the suggested methodologies. We plan to concentrate our efforts on the following important problems: 1. How to facilitate logic programming for beginners. 2. How to combine logic programs with external procedures without loosing their declarative nature.

We believe that a progress can be obtained by studying and developing formal concepts of data dependencies, types and inductive assertions for logic programs. We intend to develop methods for deducing types and data dependencies from logic programs. Such an analysis may allow to trace some inconsistencies in a program in compile time. We suggest to use additional (optional) declarations in logic programs to make the check more complete. The assertions will be used for proving partial correctness and/or for run-time check of the program. We also plan to use the information about types and

data dependencies for controlling debugging and for systematic generation of test data. Finally we have grounds to believe that this information opens for a disciplined use of calls of external (possibly higher-order) functional procedures as terms in logic programs without destroying their declarative reading.

9.4 Contacts within the Department

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

9.4.1 Courses for Graduate Students

The following courses were given in the academic years 1985/86 and 1986/87 by the members of the group or by visiting lecturers invited by the group:

Formal Language Theory (J. Maluszynski, D.Busch)

Logic for Artificial Intelligence (D.Busch)

Introduction to Logic Programming (J.Maluszynski, W.Drabent, U.Nilsson, S.Nadjm-Tehrani)

Algebraic Specifications (D.Busch, S.Bonnier)

CSP and CCS (J.Fagerström (PELAB), J.Maluszynski)

Logic Programming (research seminar)

9.4.2 Direct Contacts

The major area of interaction have been with PELAB (esp. regarding theory of programming languages, joint graduate corses, individual discussions, etc.).

9.5 External Contacts

9.5.1 External Cooperation

A great deal of the results has been obtained in cooperation with researchers abroad (see Section 9.3.). We hope to continue such a cooperation also in the future. We consider also a possibility of starting new joint projects. The laboratory remains in contact with Professor K. Futatsugi of Elechtrotechnical

Laboratory at Tsukuba (Japan). We received a prototype of the OBJ2 system developed by Professor Futatsugi in cooperation with Professor J.Goguen at SRI. The system was studied and used for preparation of laboratories in the course on algebraic specifications.

9.5.2 Conferences and Seminars

During 1986 the work of our group was presented at the following conferences:

The 4th Japanese-Swedish Workshop on Fifth Generation , Sigtuna, July 1986

The Workshop on Programming Environments and Programming Paradigms, Roskilde, Denmark, October 1986

We have regular contacts with the Swedish Institute of Computer Science in Stockholm. These contacts are supported by mutual presentations of the current research.

10.

NLPLAB
The Laboratory for Natural Language Processing

Lars Ahrenberg

The Laboratory for Natural Language Processing (NLPLAB) was formed from the research group working with Natural Language Processing within the then existing Artificial Intelligence Laboratory (AILAB), when this laboratory was split at the start of the new fiscal year 1986/87. The split of AILAB was a purely organizational change; the research carried out at NLPLAB is a continuation of that carried out by the former NLP research group.

The interests and competence of NLPLAB cover most aspects of the fields of Natural Language Processing and Computational Linguistics. Presently one application area is 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 a 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.

10.1 NLPLAB Personnel

Lars Ahrenberg, Ph.D., lab leader
Britt-Marie Ahlenbäck, secretary
Nils Dahlbäck, B.A.
Arne Jönsson, M.Sc., 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.

Figure 10.1. *Researchers in NLPLAB: Nils Dahlbäck, Arne Jönsson, Magnus Merkel, Mats Wirén and Lars Ahrenberg.*

10.2 A Short Overview of Current Research

The development of NLIs involves a number of research problems, many of which have a general theoretical interest as well. The tasks that have been of primary importance to us during 1986 are the following:

1. *The overall functioning and organization of general-purpose NLIs.* This task first of all involves the specification of appropriate behaviour for general-purpose NLIs and secondly the specification of built-in knowledge and design features that makes such behaviour possible. It is generally agreed that the knowledge required is of many different kinds. It does not only comprise linguistic knowledge in a narrow sense but also knowledge of the world (the domain of the background system) and knowledge about how to participate in a dialogue. For instance, in order to cope with a question such as

- *Kan jag få veta vilka dom är?*
- *Can you tell me which they are?*

the NLI has to know what objects are presently being talked about (to find out what is meant by *they*) and to have some knowledge about the supposed working of the background system and the supposed intentions of the user (to decide whether to answer by a simple *Yes* or *No* or a list of items). It is a difficult issue, not only to track down and give appropriate representation of the knowledge needed, but even more so to integrate it and making optimal use of it in one system employing a number of different knowledge sources.

2. *Parsing efficiency and grammar development.* 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.

3. *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. To perform such studies we have been developing tools for the analysis of dialogue and for the simulation of NLIs which will be used in a series of simulation experiments which will give us answers to questions such as the following: (a) What linguistic coverage do we need in a natural language interface? (b)How much of the language is specific for different domains? (c) What restrictions and limitations can we impose on the dialogue without it causing problems for the user?

10.3 LINLIN — a general-purpose NLI

Our research into the construction of NLIs is centered around the development of a general-purpose NLI, tentatively being called LINLIN (LInköping Natural Language INterface). By the term "general-purpose" we mean that the system should be adaptable to different kinds of background systems, such as databases, expert systems and instruction systems and have a general knowledge of natural language communication at different levels, including morphology, syntax, reference, speech acts and dialogue. To be usable a NLI must also be robust and time efficient.

During 1986 we have primarily been concerned with developing a suitable theoretical framework for LINLIN (Ahrenberg et al., 1986; see also Ahrenberg, 1987). Work with the first application — a group calendar — has started.

The dialogue capabilities of LINLIN are primarily designed for the following purposes: (a) to allow the user to build commands incrementally; (b) to provide help to the user in forming commands; (c) to ask the user for

clarifications and other information that the system might need; (d) to enable the user to make explicit and implicit cross-references between utterances, e.g. by the use of pronouns and elliptical constructions.

Whereas there exist fairly well developed techniques for the representation and invocation of grammars and lexicons in syntactic natural language parsing, the situation is much more unsettled in the case of assigning appropriate interpretations to expressions when uttered in a specific context. First of all, for many linguistic phenomena it is not even known what kind of knowledge is needed to make appropriate interpretations and hence not known how it can be represented. Second, since different types of knowledge (word knowledge, syntax, dialogue knowledge, knowledge of the world, etc.) are potentially relevant, there is also the problem of how to integrate these different knowledge sources in a working system. We may distinguish two fundamentally different ways of doing this. One is the sequential approach which applies different knowledge sources at different stages of the interpretation process, starting with a syntactic representation which is mapped onto a semantic representation which in turn is mapped into a more elaborate semantic structure and so on. The other is the parallel approach which in principle builds only one structure representing the interpretation and uses syntactic, semantic, pragmatic and world knowledge simultaneously in the process.

Both models have their draw-backs, but we favour the parallel model as it promises to reduce computation and seems to be more in line with human interpretation. The parallel model requires that the different types of knowledge sources can be seen under a common perspective. We find this common perspective in the object orientation of the system and especially in the notion of a *discourse object* (cf. Hayes, 1984). Virtually anything that can be individuated can be talked about and hence serve as a discourse object. Thus, the objects and facts of a database are discourse objects as well as the interface itself, the inputs it receives and the outputs it responds with as well as the commands that the user intends to be executed.

World knowledge is knowledge that relate directly to discourse objects, e.g., knowledge about what objects exist in the universe of discourse and what possible types of discourse objects there are. Linguistic knowledge is knowledge about how to talk about discourse objects, e.g., how we classify, measure, refer to and make statements about them. Dialogue knowledge is knowledge about how we interact with others in talking about discourse objects. The interpretation of an utterance is a discourse object description, in the simplest case a structure formed by an identified discourse object and a property ascribed to it in some appropriate mode of ascription, affirmative in the case of a statement, interrogative in the case of a question.

Object orientation require that discourse objects can be identified in the interpretation process. But the way we identify discourse objects and otherwise talk about them depends on their type. For instance, people tend to be identified by names whereas material objects tend to be identified by kind. To

a certain degree natural language can be viewed as a union of sublanguages of this kind, which are appropriate for different types of discourse objects. It follows that a general-purpose NLI should have knowledge about such sublanguages at least for the most common types of objects. As part of our first application Magnus Merkel is developing a system which handles temporal reference in Swedish.

Of course it is necessary even in a parallel system to impose some order on the interpretation process. This is a serious problem that we have only started working with and which will be in the fore-front in the year to come.

10.4 Parsing and Grammar Development

This project is being carried out as two separate but communicating subprojects, one directed towards finding efficient control strategies in parsing and the other directed towards development of unification-based grammars for Swedish.

10.4.1 Parsing Efficiency

An important motivation to this subproject stems from a tendency in natural-language processing during the last several years towards *declarative* grammar formalisms. This tendency has manifested itself with respect to linguistic *tools*, perhaps seen most clearly in the evolution from ATNs, with their strongly procedural grammars, to PATR in its various incarnations (Shieber et al. 1983, Karttunen 1986), and to logic-based formalisms such as DCG (Pereira & Warren 1980). It has also manifested itself in linguistic *theories*, where there has been a development from systems employing sequential derivations in the analysis of sentence structures to systems like LFG and GPSG which establish relations among the elements of a sentence in a non-sequential and inherently direction-independent way. E.g., things like rule ordering simply do not arise in these theories.

Declarative grammar formalisms describe linguistic structure in a non-procedural and, in principle, processor-independent way. Procedural formalisms, although possibly highly standardized (like Woods' ATN formalism), typically make references to an (abstract) machine.

The promise of declarative formalisms is to remove from the grammar writer the burden of expressing linguistic descriptions in a way which allows for efficient execution by the processor. On the other hand, the designer of the processor then has to find an overall "optimal" control strategy for processing grammars. In particular, this brings the *rule-invocation strategy* into critical focus: to gain maximal processing efficiency, one has to determine the optimal way of putting the rules to use. The potentially very large number of rules in

realistic natural-language systems makes this issue all the more important.

In a practical sense, the motivation for this subproject to some extent arised out of the grammar-development project: it was found that the execution times of the D-PATR system (Karttunen 1986) — which can be regarded as a prototypical unification-based parser — ought to be significantly reduced given a more fine-tuned (less predictive) rule-invocation strategy.

D-PATR employs rule invocation top-down (through an Earley-style chart parser). One soon realizes that a pure top-down strategy is far too predictive, postulating all the lower-level constituents that can immediately follow in a phrase. It seems that by employing a bottom-up or a mixed top-down/bottom-up strategy, it would be possible to significantly reduce the number of edges (items) produced, and, hence, parsing times and memory demands. But the question of precisely what strategy to be preferred is more or less open.

This forms the background for work by Wirén, who, starting off with a context-free chart parser in the manner of Ritchie & Thompson (1984) and Thompson (1981), has made systematic variations to the rule-invocation strategy. In an experiment reported in Wirén (forthcoming), six alternative strategies were compared in terms of the number of edges produced and overall parsing times.

The most efficient strategy turned out to be top-down filtering (a mixed top-down/bottom-up approach) which outperformed the standard top-down strategy with a factor of between three to five. Another well-behaved candidate was the elegant bottom-up strategy by Kilbury (1985).

The plan is to draw further upon these experiences in exploring how unification-based parsing might be realized more efficiently. In particular, it is expected that Kilbury's strategy, with its low overhead, will then turn out to be very useful.

10.4.2 Grammar Development

The above-mentioned D-PATR system has been available to the laboratory since January 1986. D-PATR is an environment for unification-based grammars on Xerox 1100 workstations and it was developed at SRI International (Karttunen 1986). Previous work has been carried out within the laboratory on developing grammars in unification-based formalisms; Ahrenberg has written a Swedish grammar within the lexical-functional grammar framework (Ahrenberg 1986) which uses unification as one of the primitive operations to constrain the parsing procedure.The main difference between D-PATR and other unification-based systems is that D-PATR is a general grammar development environment whereas other systems are developed to support one specific formalism. It is therefore possible to develop and test various types of grammars in D-PATR, from simple phrase structure

grammars without feature augmentations to more complex grammars, such as lexical-functional grammar (LFG) and functional unification grammar (FUG).

We have used D-PATR to develop a grammar for a subset of Swedish (Merkel 1986) and thereby tested it on a variety of Swedish constructions. Unification as a means of constraining linguistic analysis seems indeed to be a sound and promising device, but the question is whether it is feasible, or even preferable, to restrict oneself to a single primitive operation. Other questions that arise are existing drawbacks in the D-PATR system, e. g. the lack of optionality in the rules and in the lexicon.

10.5 Studying Human-Computer Dialogues

To a large extent, the work on natural language understanding in AI and computational linguistics, is motivated by the need to build more user-friendly interfaces. At the same time, there is a continuing discussion on the pros and cons of natural language interfaces as compared with artificial (query) languages. In our opinion, this discussion is based on some questionable assumptions on the nature of language and communication. Assumptions which tend to influence work in natural language, and also to some extent the work on natural language interfaces, in an unhealthy way.

Therefore we intend to do empirical studies of human-computer dialogues using a simulated natural language interface. (Dahlbäck & Jönsson 1986). The experimental situation is shown in figure 10.1 below. To the left we have the subject, in the middle we have the intelligent natural language interface, ARNE (Almost Realistic Natural language Equipment), and to the right is the background system, a data base, an expert system etc. The subjects are *not* told anything about the simulation. They all think they are using a real natural language interface that we want to test.

The experiment is conducted by placing the subjects in one room, allowing them to use a regular terminal on which they type in their questions. When the subject enters a question to the database, ARNE receives the question and if possible transforms it to an appropriate command in the query language which is then sent to the data base system. If the question is inappropriate, (it could for instance be about a completely wrong subject, or too vague a question to ask the data base system), then ARNE asks the subject for clarification, more information or gives information about the system being used, i.e. the background system.

The background system, upon receiving a query from ARNE, responds with an answer which is passed back in one way or the other to the subject. Otherwise, if there is something wrong or no answer can be given etc, then ARNE asks for or gives some additional information. In this fashion ARNE conducts a dialogue with the subjects.

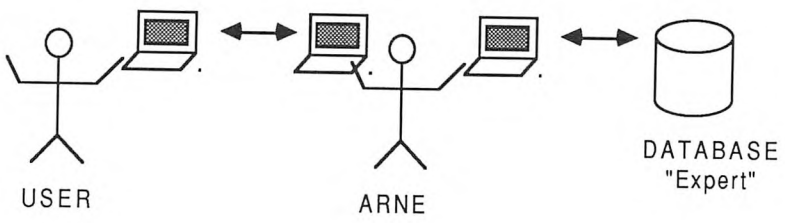


Figure 10.2. *Experimental situation*

There are, however, some problems when conducting an experiment using such a simulated natural language equipment. These includes the need to control the variation of the responses from the "system" as well as the need to achieve a reasonable response time and appropriate response quality. Short response times are necessary for achieving a continuity in the dialogue. Furthermore, since we want to study if the output from the system effects the language used by the user, we want to be able to vary the quality of the responses, sometimes using standardized output of a "canned text" quality, sometimes using a more human-like stylistic variation. To achive these goals, we have for the first background system used developed a parser that can handle most of the initial questions in a dialogue (Jönsson, 1986). We have also developed a set of standardized responses for this domain.

The environment described above is tuned up for one background system, a data base system. However, we intend to do series of experiments using different background systems, not only data base systems, but also various expert systems and CAI systems.

This means that we must reconfigurate the screen so that the menus reflect the current background system. It will also mean that we have to write some other elementary parser, based on simple pattern matching like AV-OM. Therefore it is *not* an easy task to conduct a dialogue simulation using some other background system. Of course, one could always use the system without all these refinements, but as pointed out earlier, such a system will probably be too slow and could thus lose the coherence of the dialogue.

For the analysis of the data we have developed a tagging system for dialogues. The system is interactive, and uses window techniques and pop up menues etc.

The system can easily be reconfigured for any aspect of structure that the user may be interested in.

10.6 References

Ahrenberg, Lars (1986). *Lexikalisk-Funktionell Grammatik på svenska.* Forskningsrapport LiTH-IDA-R-86-07. Also in Karlsson, Fred (ed.) *Papers from the Fifth Scandinavian Conference on Computational Linguistics*, Helsinki, Department of General Linguistics.

Ahrenberg, Lars (1987). *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, Dahlbäck, Nils, Jönsson, Arne, Merkel, Magnus och Wirén, Mats (1986). *Mot ett dialogsystem för svenska.* NLPLAB Memo 86-01.

Dahlbäck, Nils & Jönsson, Arne, *A System for Studying Human Computer Dialogues in Natural Language*, Research Report, LiTH-IDA-R-86-42, 1986.

Hayes, Philip J. (1984). *Entity-Oriented Parsing.* Technical Report CMU-CS-84-138, Carnegie-Mellon University.

Jönsson, Arne (1986). *AV-OM: ett hjälpmedel för dialogsimulering.* NLPLAB Memo 86-02.

Karttunen, Lauri (1986). *D-PATR: A Development Environment for Unification-Based Grammars. Proc. 11th COLING*, Bonn, 74—80.

Kilbury, James (1985). *Chart Parsing and the Earley Algorithm.* KIT-Report 24, Projektgruppe Künstliche Intelligenz und Textverstehen, Technische Universität Berlin.

Merkel, Magnus. *A Swedish Grammar in D-PATR — Experiences of working with D-PATR.* Research Report LiTH-IDA-R-86-31.

Pereira, Fernando C. N. and David H. D. Warren (1980). *Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks.* Artificial Intelligence 13(3), 231—278.

Shieber, Stuart M., Hans Uszkoreit, Fernando C. N. Pereira, Jane J. Robinson, and M. Tyson (1983). *The Formalism and Implementation of PATR-II.* In: Barbara Grosz and M. Stickel, eds., *Research on Interactive Acquisition and Use of Knowledge*, SRI Final Report 1894, SRI International, Menlo Park, Cal.

Thompson, Henry and Graeme Ritchie (1984). *Implementing Natural Language Parsers.* In: Tim O'Shea & Marc Eisenstadt: *Artificial Intelligence: Tools,*

Techniques, and Applications. Harper & Row.

Thompson, Henry (1981). *Chart Parsing and Rule Schemata in GPSG.* Research Paper No. 165, Department of Artificial Intelligence, University of Edinburgh.

Wirén, Mats (forthcoming). *A Comparison of Different Rule-Invocation Strategies in Context-Free Chart Parsing.*

11.

PELAB
The Programming Environments Laboratory

Bengt Lennartsson

Programming Environments is a research area in evolution. Four cornerstones, or points of reference, have been: Interlisp for its power, Unix for its simplicity, Mentor for its formal foundation, and Smalltalk for its integrated window and dialogue system. During the years a number of research projects have been carried out at several universities and research institutes: *Cedar* at Xerox PARC, *Cornell Program Synthesizer* and *The Synthesizer Generator* at Cornell University, *Pecan* and *Garden* at Brown University, *Gandalf* at CMU, and *Pathcal* and *DICE* at Linköping University, to mention a few.

The research projects have, in general, not been aiming at full size complete production quality systems, but rather at investigating consequences of certain ideas and design decisions. A general criticism from people in the real world has been that programming environment research is focusing on *programming in the small* while *programming in the large* is the real issue. The distinction *programming in the small vs programming in the large* has several interpretations. In general *programming in the small* means software design, coding, integration and testing made by the individual software specialist or by a small group of people. *Programming in the large* on the other hand puts emphasis on the management of hundreds of thousands or even millions of lines of code, hundreds of people, multiple versions, and configuration management.

Programming Environment research in general is aiming at supporting creativity and productivity for the individual software designer, and sometimes at investigating new architectures for software systems. It may be true that in some of the biggest industrial software projects, the software design and coding is a small fraction of the total effort.

However, in order to achieve the best total result from the software process, it may be a good idea to choose the best possible tools and techniques for the

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

individual software designers and then tune the management and administration to the techniques used rather than vice versa. No doubt, better support for the individuals means increased productivity and fewer individuals to manage and control for a specific task. *Programming in the small* and *programming in the large* represent two views on how to handle the software crises: better technology or better administration and management.

Experiences from Xerox Corporation, Rational, Apple, and other companies have evidenced that ideas from programming environment research can be scaled up from the original toy implementations to commercial competitive systems.

11.1 Short Summary of the Activities During 1986

In this paragraph the PELAB work during 1986 is presented very briefly. The different projects are described in more detail under **Project Presentations** later on.

11.1.1 PELAB Research Projects 1986

The idea in PELAB is to have a kernel project with a number of individual thesis projects attached to it.

DICE, *Distributed Incremental Compiling Environment*, was the kernel project during 1982-85. Peter Fritzson, the DICE architect, has been with SUN Micro Systems from April 1985 to July 1986. After return to Linköping he has been working part time on transforming the DICE prototype, implemented in Interlisp on DEC-20, to a demo system implemented in Pascal on SUN-3. Peter Fritzson has presented some new papers on DICE related work. DICE, as a kernel project, is essentially completed, but a couple of individual thesis projects started in the DICE context still exist. Rober Bilos has presented his first paper on a token based editor, TOSSED, and so have Nahid Shahmehri and Mariam Kamkar on their work on Runtime Dependent Program Flow Analysis.

Ola Strömfors has presented his licentiate thesis and two more papers on his work on the ED3 editor. One of the IDA related spin-off companies, Programsystem, has developed ED3 to a software product now available for a number of computers and operating systems.

Kristina Ernstsson's licentiate project is inactive, since she is on leave of absence during the academic year 1986-87.

The new kernel project PEPSy, *Programming Environment for Parallel Systems*, is gathering speed. The definition of the problems has been sharpened, and one report and one paper have been written. Johan Fagerström, licentiate, and three more graduate students are currently

involved in the work.

11.1.2 PELAB Personnel 1986

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

Supervisors:

Peter Fritzson, PhD	At SUN March 1985 - July 1986
Anders Haraldsson, PhD	

Employed graduate students:

Rober Bilos, MSE	
Kristina Ernstsson, MSE	Not 1986/87
Johan Fagerström, licentiate	
Bengt Karlstrand, BSc	From Aug. 1986
Mariam Kamkar, BSc	
Yngve Larsson, MSE	
Nahid Shahmehri, BSc	
Lars Strömberg, MSE	
Ola Strömfors, licentiate	

Associated persons:

Johnny Eckerland, licentiate, Epitec AB
Kenth Ericson, Softlab AB
Pär Emanuelson, PhD, Epitec AB
Azadeh Ghaemi, undergraduate student
Tommy Olsson, MSE, IDA
Lennart Lövstrand, undergraduate student
Jerker Wilander, Softlab AB

11.1.3 Publications 1986

Rober Bilos: *A Token-Based Syntax Sensitive Editor.* Proceedings of the workshop on Programming Environments - Programming Paradigms. Roskilde, Denmark. October, 1986.

Johan Fagerström, Yngve Larsson, Lars Strömberg: *Debugging Techniques for Distributed Environments.* Proceedings of the workshop on Compiler Compilers and Incremental Compilation. Bautzen, GDR. October, 1986.

Johan Fagerström, Yngve Larsson, Lars Strömberg: *Distributed Debugging - - collected ideas.* LITH-IDA-R-86-21. June, 1986.

Peter Fritzson: *A Multi-Language High-Level Common Intermediate Representation.* Proceedings of the workshop on Compiler Compilers and Incremental Compilation. Bautzen, GDR. October, 1986.

Peter Fritzson: *Systems and Tools for Exploratory Programming - Overview and Examples.* Proceedings of the workshop on Programming Environments - Programming Paradigms. Roskilde, Denmark. October, 1986.

Mariam Kamkar, Nahid Shahmehri: *Runtime Dependent Program Flow Analysis.* Proceedings of the workshop on Programming Environments - Programming Paradigms. Roskilde, Denmark. October, 1986.

Bengt Lennartsson: *Programming Environments and Paradigms - Some Reflections.* Proceedings of the workshop on Programming Environments - Programming Paradigms. Roskilde, Denmark. October, 1986.

Bengt Lennartsson, Anne-Marie Fransson, Jerker Wilander: *Future Techniques for Software Systems Design* (in Swedish). Sveriges Mekanförbund. Mekanresultat 86004.

Ola Strömfors: *A Structure Editor for Documents and Programs.* Licentiate Thesis No 73. Linköping Studies in Science and Technology. Linköping 1986.

Ola Strömfors: *Editing Large Programs Using a Structure-Oriented Text Editor.* Proceedings of International Workshop on Advanced Programming Environments. Trondheim, Norway. June 1986.

Ola Strömfors: *A Structure Editor as a Template for Programming Environment Functions.* Proceedings of the workshop on Programming Environments - Programming Paradigms. Roskilde, Denmark. October, 1986.

11.2 Project Presentations

The current research in PELAB is a continuation of the work in program manipulation projects, that was done during the seventies, and of the INTERLISP experience in general. PELAB was established as a separate research group in 1980. Since then four PhD dissertations and two licentiate theses have been presented. The work has covered a large area, from partial evaluation and its application to Pattern Matching and to Specification of an Abstract Prolog Machine, to Compiler Writing Systems and Incremental Compilation. A very successful project, DICE (*Distributed Incremental Compiling Environment*), is approaching termination, and a new kernel project, PEPSy (*Programming Environment for Parallel Systems*) has been set up.

In the future it is quite likely that we will return to program analysis and program transformations again. 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 coding very large systems in a standardized general purpose programming language. We are just about to start preliminary studies on new architectures for large software systems, and the kernel project to follow PEPSy in a few years will, potentially, be on Multi-Level Software Architectures. In such a situation general analysis and transformation tools will be very useful.

11.2.1 The DICE Project

Peter Fritzson, Johnny Eckerland, Mariam Kamkar, Ralf Nilsson, Nahid Shahmehri, Ola Strömfors

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

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

A prototype of DICE was first implemented. The tools were running on a DEC-20, the host, where also all the information of the developed program was saved. The host was connected to a target, a PDP-11, where the developed program was executed. Among the results should be mentioned that:

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

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

- **Remote debugging and maintenance** is easy to achieve with the DICE system configuration.
- An incrementally compiling system like DICE which has a program data base is especially suitable for the **development of big programs**, on the order of 20 000 to 100 000 lines of code. Compiled code gives fast execution and incrementality gives fast program update and powerful debugging facilities.
- **Separability** - the compiled program is separated from the source code so that it can execute outside of the program development system.
- **Connectivity** - the DICE system can be connected to a malfunctioning production program in order to debug it or to correct it. This can be done after the error has occurred and need not be planned in advance.

The DICE prototype, implemented in Interlisp on the DEC-20, has been machine translated to Pascal, and when the translated system is in operation on a SUN-3, the project as such will terminate. However, the results, the experiences, and the people will be necessary when implementations and experimental research become important in the PEPSy project.

11.2.2 Runtime Dependent Program Flow Analysis

Mariam Kamkar, Nahid Shahmehri

The long term goal in this project is to investigate how results from static program flow analysis can be combined with dynamic state information in an executing system to support interactive and incremental updating, testing and debugging.

Interactive access to static analysis information is a powerful facility which is often far superior to manual inspection of program text. Static analysis can be performed with varying degree of precision. We identify three classes of analysis with successively increasing precision:

- Cross-reference analysis is the first class. Examples of queries which can be handled by such a tool are:

  Which procedures/functions call a named procedure/function ?

  Which objects are declared/referenced/modified by a named procedure/function ?

  Such queries can be answered without doing any data or control flow analysis of a program. A well-known example of such a tool is the Masterscope facility in Interlisp.

- Static analysis of programs including control and data flow analysis is the second class. Other kinds of queries which may be answered through this analysis are:

  Where may a variable have been assigned its current value ? or

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

  Show the data flow from variable x to variable y.

- The last class, and our proposed addition to the two categories above, is runtime dependent program flow analysis, *i.e.* control and data flow analysis which depends on the current runtime state. This kind of analysis copes with the same queries as in the previous category, but with greater precision.

To solve the query problems, we first deal with the control and data flow analysis problems. This problem area may be divided into two subproblems, intraprocedural and interprocedural flow analysis.

Intraprocedural data flow analysis is performed on the basic blocks of a procedure body. Having access to the resulting information, we are able to solve two common data flow problems, *i.e. live variables* and *"reaching definitions*, see [Hecht-77]. These results are used to establish *definition-use chaining* for the relevant variables. Some extra computation will be needed during a query to get to the right instances of a variable within a basic block.

Interprocedural analysis determines the effects of procedure calls on the variables of a program, and how to associate this information with call statements. In the case of optimization, one usually is interested in the direct effect of a variable on another. Here we are also interested in the indirect effects. We have started the implementation of a tool which will perform runtime dependent program flow analysis. This tool will be integrated with an existing debugger which provides access to the runtime state.

For interprocedural data flow analysis we are going to use the algorithms described by [Banning-79].

We are using an abstract syntax tree as the internal representation of programs. The abstract syntax tree has been augmented with control flow information, represented as arcs between basic blocks. Global information will be computed once and stored for further use. Some local information will be recomputed every time it is needed.

The algorithms described in [Hecht-77] for *live variables* and *reaching definitions*, may be replaced by the incremental variants given by [Ghodsi-83], who also has incremental variants of Banning's interprocedural algorithms.

References:

[Banning-79] J.P. Banning. *An efficient way to find the side effects of procedure calls and the aliases of variables.* Conf. record of the Sixth Annual ACM SIGPLAN/SIGACT Symposium on POPL. January, 1979. pp.29-41.

[Ghodsi-83] V. Ghodsi. *Incremental analysis of programs.* PhD thesis. University of Central Florida, 1983.

[Hecht-77] M.S. Hecht. *Flow analysis of computer programs.* Elsevier North-Holland, New York, 1977.

[Horwitz-85] S. Horwitz, T. Teitelbaum. *Relations and attributes: A symbiotic basis for editing environments.* Proc. of the ACM SIGPLAN 85 Symp. on Lang. Issues in Programming Environments. Seattle. June, 1985. pp.93-106.

[Tischler-83] R. Tischler, R. Schaufler, Ch. Payne. *Static analysis of programs as an aid to debugging.* Proc. of the SIGSOFT/SIGPLAN Software Engineering Symposium on High-Level Debugging. March, 1983. pp.155-158.

11.2.3 TOSSED, A Token-Based Syntax Sensitive Editor

Rober Bilos

One observation in the work on both ED3 and DICE is that re-scanning is an expensive operation in an incremental environment. The TOSSED project is aiming at investigating token based editing. The program is represented as a sequence of tokens. Text is conventionally entered character by character and parsed token by token. The main idea in TOSSED is to scan the user input as it is entered and internally represent the program as a token sequence. This avoids time consuming re-scanning.

In the implementation of TOSSED some of the functions normally available in syntax-directed editors have been included such as template instantiation, automatic indentation and prettyprinting, lexical and syntactic error handling. At present TOSSED supports editing of Pascal programs. As the parser, prettyprinter, and the syntactic error recovery mechanism are table-driven, editors supporting other languages can easily be generated.

The token type has a key role in TOSSED. As soon as a token is recognized, the scanner checks its type and takes appropriate action. All legal tokens entered by the user are inserted into the token sequence. Each token type is associated with certain prettyprinting information.

In TOSSED a lexical error is indicated when a token is entered, and syntax errors are indicated on the *syntax check command.* A program in text form with lexical or syntactic errors can be loaded into the editor.

The program can be parsed on command whenever the user wishes. Automatic parsing can be turned on and off by the user. The parse state is saved for each

interval of 20 lines (screen page length). TOSSED is implemented in Pascal on DEC-20. For efficient memory usage it has its own garbage collection. The parser has been generated using an LALR(1) parser generator developed at the department. The syntactic error recovery routine was implemented already in 1983 for batch use. It will be modified to work in an interactive environment. If an error is not repairable the cursor is placed on the erroneous token and the user is asked to correct the error.

We plan to integrate this editor as an alternative front end to the DICE system. In order to do this, the editor has to record where in the token sequence updates have been introduced during an editing session. The memory requirements and the performance of TOSSED will be compared with conventional text based editors as well as parse tree based ones.

11.2.4 A Structure-Oriented Text Editor for Large Programs

Ola Strömfors

The ED3 editor designed and implemented by Ola Strömfors has been in use for structured documents in general for several years. It has recently been used as a template for a powerful language oriented editor-prettyprinter-syntax checker. The language oriented features are available for Ada and Pascal. The editor has now evolved to a software product marketed by Programsystem AB, one the spin-off companies around.

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

The structure is *not* used to build parse trees. Instead the user is free to build any tree he wants. Often the user wants to divide a long sequence of procedures on the same level into different groups. As *head* of each tree node he can put a comment describing this group of procedures. This is useful for programming languages without nested procedure declarations, such as *C* or even assembly languages. Different nodes can even be written in different programming languages. Dividing the program into text nodes, with for instance a procedure in each, gives the user a good overview of his program. The tree structure given by the author may also help others to understand the program.

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

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

The ED3 editor is implemented in Pascal and Ada on DEC-20, VAX-11 VMS, and several UNIX systems as SUN-3, Gould Power Node, NCR Tower, and also on IBM PC.

11.2.5 The PEPSy Project

Johan Fagerström, Bengt Karlstrand, Yngve Larsson, Lars Strömberg

The PEPSy project is a five-year project aiming at understanding and constructing tools and methods for software development for distributed systems. During the first phase, ending June 1987, existing methods, tools, and notions will be investigated.

For our purposes, a distributed system consists of a number of geographically separated processors that communicate via messages. We do not consider systems based on shared memory implementations of communication. Critical applications for this kind of systems can be found in areas such as process control, telecommunications, defense systems, and office automation.

Distributed, or parallel, systems introduce a number of new dimensions to system design and programming. For example, designers, and sometimes users, must take time delays and limited bandwidth into account. This extra complexity compared to sequential systems must be handled if robust, reliable, and correct programs are to be constructed.

A goal in PEPSy is to abstract away from system specific details, allowing the programmer to think in terms of the problem. However, it must be realized that advanced users will use knowledge of system structures to improve the performance of their programs. We intend to provide advanced tools for such expert users. The PEPSy programming environment will include traditional tools, *i.e.*

- editors
- compilers and interpreters
- linkers and loaders
- runtime support tools

together with tools specially constructed for parallel systems, *e.g.*

- **demon editor and linker**, a tool for specifying and invoking watch-dog processes that monitor and influence some part of an application program on a particular node.
- **static analyzers**, providing extra functionality compared to traditional tools especially for concurrent processing *i.e.* deadlock and livelock

detection

- **interface generator**, a tool generating a view of a system, since the complete distributed system will not be visible at one time. This tool will also act as a metaphor for tool integration
- **system browser**, a tool for browsing through static information in the system, such as program databases
- **name server**, a tool for keeping track of dynamic objects in the system, such as processes and logical communication links, and
- **tracing facilities**

Distributed systems introduce a number of new problem sources as well as new types of bugs. Tracing is complicated by non-determinism and lack of observability of program states. Debugging is further complicated by the non-reproducibility of events. In the immediate future, our research will be focused on the following areas:

- **Programming paradigms for distributed systems.** Traditional languages such as Pascal and Lisp can be extended with primitives supporting concurrency. Another approach is to construct new languages designed for distributed systems such as Argus, Cell and Conic.

  During the first part of this project we have studied various languages and problems related to language design for distributed system. In our future research we will concentrate on languages with explicit processes and communication primitives, *e.g.*, Conic, Mesa, or Ada.

- **Tools supporting debugging.** A number of prototype tools based on our research and experience with existing parallel systems will be implemented. For example, a trace tool together with intelligent inspection routines, based on knowledge of data and control dependencies and system timing of events, will aid the programmer in detecting subtle bugs introduced by varying time delays.

- **Management of static and dynamic objects.** Naming and addressing of objects are traditionally difficult in distributed system. The programmer should be allowed to refer to object by name without knowing their exact location within the system. At the same time he must be allowed to specify the object's whereabouts for performance reasons. Tools supporting browsing of static and dynamic information are essential for both program development, testing, and system maintenance.

- **User interaction and programming support.** A consistent and supportive user interface is of utmost importance due to large system sizes, complex states and dynamic behavior in distributed systems. Therefore, extra efforts must be put into developing a uniform programming tool interface, based on modern bit-mapped displays and pointing devices.

11.3 Industry Related Activities

It is not the purpose of PELAB to participate in the development of software products for industry. However, we are looking upon our role as a communication channel between industry and international programming environment research. Results from PELAB and other groups are available in the IDA Knowledge Transfer Program, presented elsewhere in this report.

In the PELAB area there are also two spin-off companies: Softlab and Programsystem. We are working in close contact with these companies and also with Epitec in the AI area.

During the year we have also actively made two more specific contributions to the knowledge transfer to industry. The first is a project Future Techniques for Software Design funded by *Mekanförbundet*, an association of national industry in the mechanical, electrical, and electronics fields. The second contribution is the organization, jointly with ASLAB, of the Software Environment Week at the department. The components of the week were:

- SOFT4 - a two day tutorial on Software Development Environment,
- Environments in Use, a one day Ada-in-Sweden seminar
- Ada Europe Environment Working Group, a two day meeting
- exhibition and social activities at the department during the evenings

During this week there were participants from ten different countries.

12.

RKLLAB

The Laboratory for Representation of Knowledge in Logic

Erik Sandewall
Professor of computer science

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

12.1 Researchers and Projects.

12.1.1 Activities.

The activities in RKLLAB during 1986 have been in the following, overlapping and interacting areas:

Non-standard logics and their implementations, in particular:
- non-monotonic logic and reason maintenance (NML-RM)
- fuzzy logic (FL)
- constraint programming systems (CPS)

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

Office systems, in particular:
- theories of office software (ThOS)
- the LINCKS project

Representation of knowledge about machinery and processes, in particular:
- theoretical analysis of action structures (AS)
- laboratory system for intelligent, adaptive machinery (IAM)
- geometrical reasoning (GR)

12.1.2 Laboratory members.

The following researchers have been members of RKLLAB during 1986 (or a part of the year):

	Activities or main interest
Laboratory leadership	
Erik Sandewall	AS
Lillemor Wallgren, secr.	
Ann-Marie Jacobson, secr.	
Project leaders and senior graduate students:	
Dimiter Driankov	FL, LINCKS
Jim Goodwin	NML-RM
Jalal Maleki	CPS
Lin Padgham	LINCKS
Michael Reinfrank	NML-RM
Ralph Rönnquist	ThOS, LINCKS, AS
Graduate students and masters thesis students:	
Johan Andersson	LINCKS
Christer Bäckström	GR
Patrick Doherty	NML-RM
Peter Haneclou	NML-RM
Johan Hultman	IAM
Stefan Wrammerfors	ThOS
Peter Åberg	LINCKS

12.1.3 Main current achievements.

The major achievements during 1986, in terms of finished research results, have been:

a) identification of the greatest lower bound operation with respect to the relation of "contains more information" between networks (Rönnquist)

b) development of a constraint programming system (Maleki)

c) a formal characterization of action-plans for machinery and the conditions on 'correct' plans (Sandewall and Rönnquist)

d) completion of work on analysis of recurring cycles and 'pipelining' of plans (Sandewall)

e) a formal characterization of reason maintenance operations in terms of four-valued logic (Haneclou)

f) a characterization, using logic, of admissible states and the effects of movement operations, in a simple geometrical pseudo-world (Bäckström)

g) implementation and first demonstrations of a high-level programming system for controlling discrete events in mechanical systems (Hultman)

Also, the on-going and larger LINCKS project has produced several published reports regarding various aspects of the system being developed (Padgham et al).

In view of the length of this list, and the space constraints in the present volume, we shall go into detail for some of the projects only. The publication list at the end of the chapter gives the references to publications for all the results.

12.2 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 world" 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.

12.2.1 Non-monotonic logic and reason maintenance.

One important and specific application of non-monotonic logic is for reasoning about property inheritance with exceptions. Suppose we have a number of concepts, such as "vehicles", "personal cars", "buses", "vehicles manufactured in Sweden", "boats", "diesel driven vehicles", "vehicles with the driver's seat to the left", etc. There is an obvious specialization ordering between the concepts, for example "Saabs are front- wheel drive cars". This *isa* relation is a partial order, where one concept may be a specialization of more than one other concept.

When practical application domains are encoded as such a structure, there is a frequent need to also admit exceptions in the hierarchy, and to interpret the *isa* relation more loosely as "most A's are B's" or "an A is generally a B". For example, one would then represent "Volvo cars are in general gasoline driven" (using the *isa* relation), but also to have more specialized concepts which *isa* Volvo-cars but which do not *isa* gasoline- driven.

The logic of the *isa* relation with exceptions has a number of strange aspects. In a work which was mostly done during 1985, but which was further extended in 1986, Erik Sandewall defined a partial semantics and a set of inference rules for that relation (ref. 11). One of the surprising aspects of this logic is that the validity of a conclusion does not only depend on whether certain assumptions are satisfied, but it also depends on whether those assumptions are logically unrelated or not. - The work was based on a 'functional' approach to non-monotonic logic which was developed last year (ref. 18).

It is generally agreed that non-monotonic logics can only be implemented as reason-maintenance systems (RMS), i.e. systems which keep track of the logical support for each conclusion, and which is able to retract and re-establish conclusions as their support is lost and re-gained. Jim Goodwin's Ph.D. thesis work in the area of reason-maintenance systems has continued during 1986.

Also, Peter Haneclou has extended the 'functional' approach to non-monotonic logic into a characterization of reason maintenance (ref. 17). His key idea is to view a proof as a tree-like structure rather than as a sequence. Suppose we have the specialized inference rules

$a \Rightarrow b$
$b \Rightarrow c$
$a \Rightarrow d$
$c,d \Rightarrow e$

and we start from {a} as a set of axioms or a belief set. In the previous work, we considered proof sequences like the ones shown in figure 12.1, where each step "upwards" in the sequence represents the use of one more inference rule. Haneclou considers instead proof structures like in figure 12.2, where inference rules are applied at the bottom-most position which is possible. The conclusion set from a proof-"tree" is then the join of the leaf nodes in the tree.

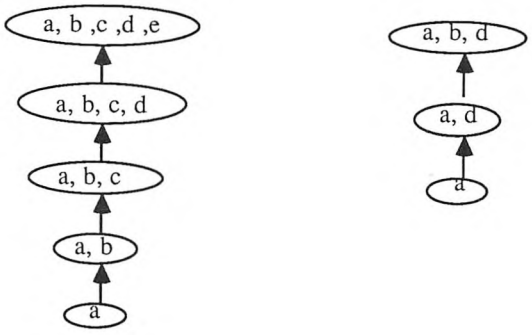


Figure 12.1. *Example of proof sequence.*

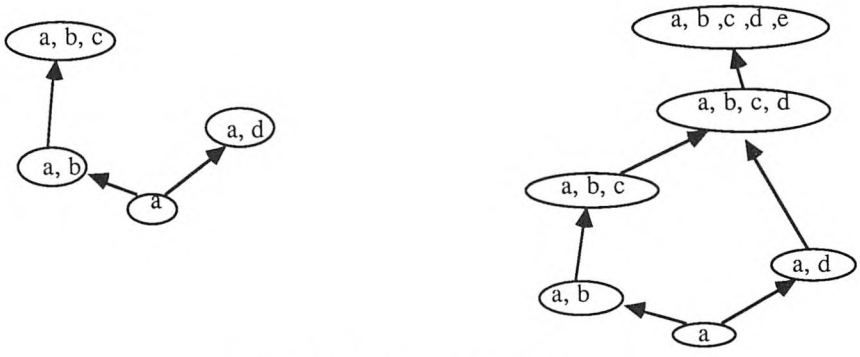


Figure 12.2. *Haneclou's proof structures.*

The abstract description of the inference "engine" is now that it gradually extends a proof "tree". Like a reason maintenance system, the proof tree retains the dependency information for the successive conclusions. If non-monotonic inference rules are used (like in Reiter's default logic), a contradiction may develop in the conclusion set. The appropriate action is then to remove one branch from the proof tree before proceeding, which is the formal counterpart of backtracking in the RMS.

Haneclou shows that a well-formed proof tree (where all conclusions are drawn at a point which is as low as possible in the tree) exists if and only if there is a proof sequence. He also shows that there are well-defined "break-points" which define how to prune the tree when a contradiction has occurred.

During the fall semester, Michael Reinfrank has given a course on non-monotonic logic and reason maintenance, and prepared lecture notes for the course. A follow-up course will be given during the spring 1987. He also advised students who are interested in the area, and continued his own

research on rule-based non-monotonic deduction systems and related theories. A first prototype will be available early 1987 (ref. 9).

In October 1986, Michael Reinfrank presented an invited survey paper at eth Workshop on Truth Maintenance in Berlin, West-Germany (ref. 10) With the assistance of Christer Bäckström, he organized a Workshop on the Frame Problem on January 19-20, 1987.

12.2.2 Fuzzy logic.

Dimiter Driankov has continued his research in fuzzy logic during the year, and has produced several publications (ref. 1-4). We plan to include an account of his work in next year's progress report.

12.2.3 Constraint programming systems.

Jalal Maleki has completed his work constraint programming systems, which is being presented as a licentiate thesis (ref. 19; the licentiate is intermediate between M.Sc. and Ph.D.) The work includes an implementation of the constraint system "engine" and a graphical user interface which makes it easy to design and understand constraint structures. It also includes a formal characterization of the constraint "engine's" operation. (Also reported in ref. 5)

12.3 Office systems.

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

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

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

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

- development of a theory which (in an empirical sense) accounts for observed phenomena in actual office software.

The first three of these goals are obviously in line with the overall focus of RKLLAB; the fourth goal is a necessary complement in order to understand office systems.

During 1986, most of the effort was directed into an experimental implementation project, the LINCKS project. There was also some theoretical work on representation issues.

12.3.1 Office systems vs. end-user operating systems.

The term 'office system' refers to a particular type of usage environment and a particular market. From the software engineering point of view, we feel that a more appropriate term would be "end user operating system". We then think of an operating system as the software which serves the user with a top level command language, a facility for storage of data (such as the file system), and the support for navigating between and for coordinating a number of specific 'application' programs. Instead we de-emphasize the role of the operating system for administrating time-sharing of the resources in and around the computer.

Classical operating systems (also including Unix) have been designed with the needs of the programmer and the conventional programming language in mind. It is interesting to consider how one should design an operating system in order for it to be a good substrate for fourth-generation software, expert systems, and other modern software technology, and in order to be as convenient as possible for the end user. For example, one might prefer to have a system-wide data base system (an "information repository") instead of a conventional, directory-oriented file system. The work that we do on office systems, along both theoretical and practical lines, attempts to answer also such questions.

12.3.2 Theories of office software.

Ralph Rönnquist has continued his work on network structures, a network being a collection of nodes, and labeled arcs from nodes to nodes. One network is said to "contain more information than" another network if there is a way to increase the information contents of the latter network and obtain the former one.

Addition of more nodes or arcs to a network is one way to increase its information contents. In other words, a network contains at least as much information as any of its sub-structures. Another way to inrease the information is by identification of nodes. When a network is created for expressing some amount of information, it may happen that the same "thing" in the application, is represented by two or more, distinct nodes in the network. The addition of information that identifies two nodes as referring to the same object, is an information increase. Operationally, it results in a network contraction, i.e. the identified nodes are brought together into a single node while retaining all their arcs.

Taken together, the extension and contraction operations form a transitive relation between networks. It defines equivalence classes between networks; each class consists of networks which are equivalent with respect to information contents. For example, the networks in figure 12.3 are equivalent since each may be obtained from the other by operations that "increase" information (in the sense of s). Also, the networks in figure 12.4 are equivalent.

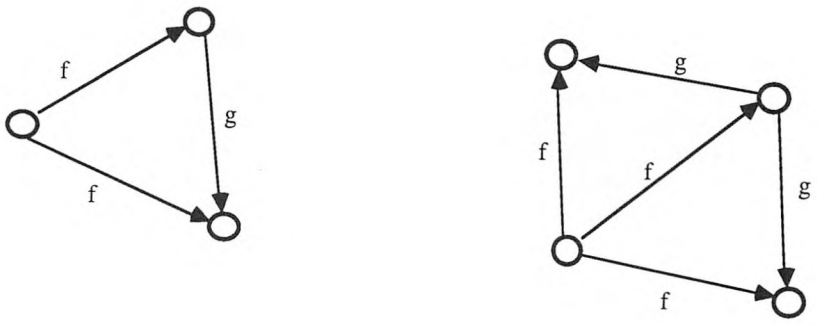


Figure 12.3. *Equivalent networks*

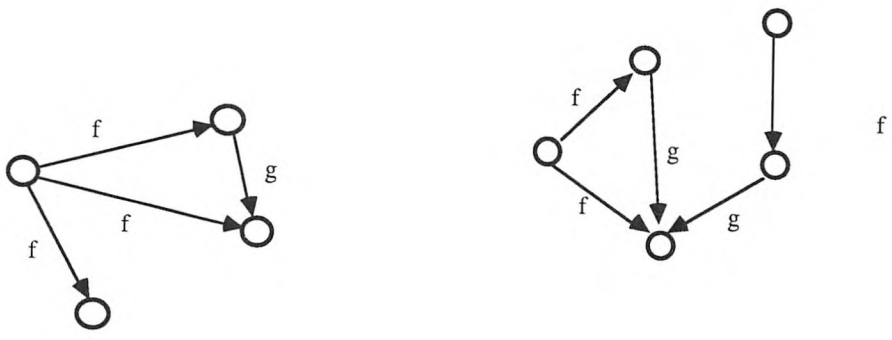


Figure 12.4. *Equivalent networks*

Rönnquist has proved that there is a unique member of each equivalence class which is maximally contracted, and which is then a suitable canonical element. He has studied the "contains less information" relation between such equivalence classes, and proved that it defines a lattice. The join operation in the lattice is simple: it is obtained by putting two networks side by side (without connecting them with any arc), and then contracting the resulting, unconnected network maximally in order to obtain the canonical element.

The meet operation a $\sqcap$ b for networks is more complex, but has now been found. It is obtained by forming a new network by a Cartesian-like operation: each pair of a node in a and a node in b, constitutes a node in a $\sqcap$ b. Two nodes in a $\sqcap$ b are connected by an arc iff there are arcs between their respective components in a and in b. This is a significant result.

These results are reported in ref. 14.

12.3.3 The LINCKS project.

The "Linköping Intelligent Knowledge Communication System" is being developed as a prototype next-generation office system or (for the reasons that were discussed in section 12.3.1) a prototype next-generation end-user operating system. The project started in late 1985, and is led by Lin Padgham. The other project members during 1986 have been Johan Andersson, Dimiter Driankov, Ralph Rönnquist, and Peter Åberg.

The first half of 1986 was devoted to design work, both global design principles and specification of the basic parts of the implementation. Actual implementation has dominated since September 1986, and is expected to continue so until the end of March (1987). After that we expect to go back to more global design issues.

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

The goal of the LINCKS project is now to build a system which supports communication in such a way that some of the low-level communication acts can be automated, using the technology of knowledge-based systems. This goal determines a number of crucial features in the system:

a. The goal structure, where higher level actions are decomposed into lower level actions, must be explicitly stored in the computer system. Only then can the built-in "expert system" have a chance to "understand" the context of what they are supposed to do, and apply whatever "common sense" they have been endowed with.

b. The structure for representing information in the system must be sufficiently rich, so that the user can conveniently express himself in it. At the same time, it must be sufficiently uniform and clear to allow "expert system" type software in LINCKS to manipulate the data base.

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

c. In line with the spirit of both knowledge based systems and conventional data bases, all information that is stored in LINCKS should be as transparent as possible and allow "multiple use". For example, the goal structure described in point (a) should not be a special-purpose structure. Instead, there should be a global facility in the system for storing and manipulating action plans, i.e. partially ordered sets of event descriptions. This facility could be used by the user as a planning tool in his or her work, personally or in the organization. The same facility is also used for internal purposes by the LINCKS system in organizing its work. This design results in a desirable integration between the user's work-plan and the computer system's work-plan.

d. A communication tool must be available whenever its user wants to communicate. It can therefore not be restricted to a "workstation" in the user's office; it must be portable and able to travel. This becomes practical through the present rapid development of truly portable computers. Consequently, the information repository in LINCKS is designed as a distributed system from the basic level up.

We expect that the first, small-scale demonstrations of the LINCKS system shall be made during 1987. Parts of the work which has been done so far have been reported in ref:ces 6-8.

12.4 Representation of knowledge about machinery and processes.

Our department has done research on office information systems for almost ten years. More recently, we have also started to work on the application area of "machinery", with the following concrete examples in mind:

- machines that do manufacturing operations automatically (NC-machines, industrial robots, etc)

- machines that move around - unmanned vehicles or vehicles with computer assistance to the driver

- machines that lift and move cargoes (cranes, trucks, some uses of industrial robots)

From a computer science point of view, some of the crucial theoretical issues in such applications are in action planning and in geometrical reasoning. Also, the design of high-level programming systems, for use in the control of machinery with distinct actions, is an open research problem. We are working on all of those areas.

12.4.1 Theoretical analysis of action structures.

The definition of the behavior of a machine consists of two levels. Globally, the plan specifies which distinct actions are to be performed, and what is required about the order of those actions in time. The global plan is often best described by graphical means, for example as in figure 12.5.

Figure 12.5. *Graphical description of action plan.*

Locally, each constituent action in the plan also has to be defined. Sometimes it is appropriate to use nesting of plans, so that an action is defined as a sub-plan, but at some points one comes to atomic actions. In the engineering context, atomic actions are often characerized by dependencies between a number of quantities, and very often by dependencies which achieve a feed-back in order to control the behavior of the machine and keep it within the desired bounds. - The use of quantitative dependencies is relatively unfamiliar in A.I., with its roots in logic, but has of course beën carefully studied in control theory.

Most of our work during 1986 has been concerned with the global level, although we have also started to study the very crucial issue of how these two levels can be made to work together. When we use the term "action plan" we will refer to the global level of distinct actions.

One concrete example of where such action plans are needed, is for programming automatic manufacturing cells. Such a cell typically receives a flow of work-pieces, performs operations on each work-piece, and generates an outgoing flow of processed work-pieces. There are a few machines which perform distinct processing steps, and an industrial robot or other agent which moves workpieces from one point to the next. (Figure 12.6) Typically the time taken in each of the processing machines is larger than the time required for moving a workpiece, so that it is natural to "pipeline" the cell and allow several workpieces, in successive stages of completion, to be in the cell at once.

In one paper (ref. 11), Sandewall and Rönnquist have developed a formal characterization of action plans, like the ones needed for such a cell. The environment of the cell is characterized by a number of "features" with distinct values. For each action, it is specified what values the features must have at

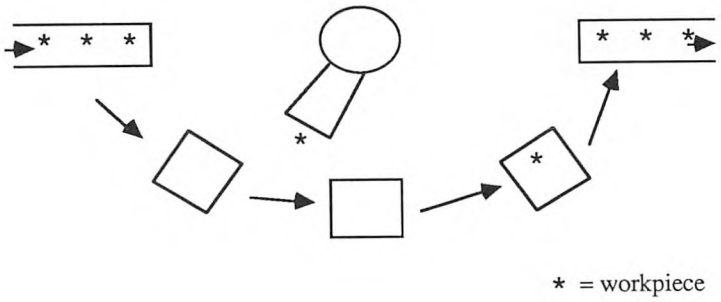


Figure 12.6. *Moving work-pieces through the cell.*

the beginning and at the end of the action (for those features whose value is changed) or throughout the action (for those features whose value must remain constant). The paper specifies the criteria which must be imposed on the total plan, in order to guarantee that the conditions for each action are satisfied.

In a later paper (ref. 20), Sandewall uses those results for defining the "pipelining" operation on the cell's program. In other words, given a program for the manufacturing operation as seen from the individual work-piece, and a proposed program for the cell that allows it to keep several work-pieces going at once, the paper defines the criteria for when the proposed cell program correctly implements the given workpiece program.

12.4.2 Laboratory system for intelligent, adaptive machinery.

The COPPS system (ref. 16) is being developed by Johan Hultman, and is a prototype software system for controlling actual machinery. Using COPPS, the intended behavior of the mechanical system is described on the two levels that were defined in the previous section. On the upper level, there is a collection of actions, and a token-passing mechanism which is used both for administrating the 'resources' that the actions perform, and for obtaining the correct temporal ordering between the actions.

On the lower description level, on the other hand, each action is characterized as a number of connections for continuous update. For example, if an action is performed as a simple feed- back loop, where the current value of an actuator is a function of the current value of a sensor, and the given norm value, then the lower description level will be as in figure 12.7: there is one data object for the sensor, one data object for the effector, one data object that represents the action. The arrows in figure 12.7 represent paths for data-streams, where successive values are fed from one field to the next, and may be numerically transformed and combined on the way.

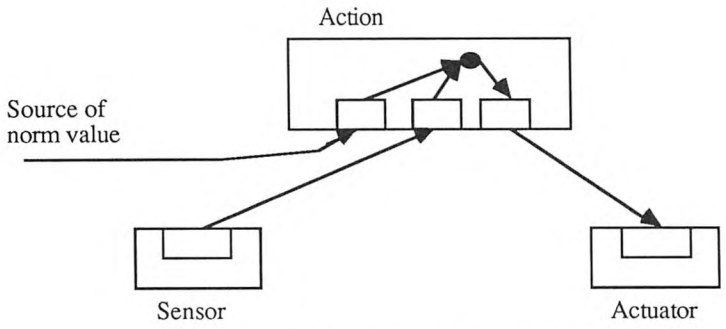


Figure 12.7. *Low level action description*

The COPPS system has been implemented on an Apple MacIntosh Computer, and is presently used for controlling a toy model of an industrial robot. The immediate plans are to extend the model into a more complex one, and to build a programming environment which supports the design, analysis, and manipulation of the mechanical programs.

12.4.3 Geometrical reasoning.

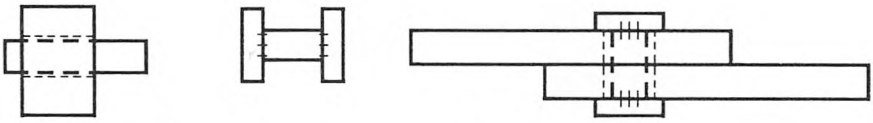
Geometry is well known to be a hard problem for knowledge based systems. Like for programming of machinery, geometrical reasoning requires a synthesis of qualitative and quantitative operation. Very little work has been done so far on geometrical reasoning.

For a first attempt to come to grips with this difficult area, Christer Bäckström defined a very simplified geometrical world, and has studied its properties using the techniques of logic. His model world is essentially two-dimensional, with an exception that will soon be described, and it is populated by rectangles whose sides are parallel to the coordinate axes. Thus each rectangle can be characterized by two vectors, viz. the position of its lower left corner, and the diagonal vector.

If two rectangles are positioned side by side, the two adjacent sides can be "glued together" along the adjacent sides. The operation of moving a rectangle is defined, in such a way that if two rectangles are glued together and one of them is moved, the other ones move with it.

Rectangles may have holes, which again must have rectangular shape. A smaller rectangle may be inserted, fully or partly, into a hole in a larger rectangle. Figure 12.8 show some examples of configurations which may be formed in this simple model world. Crossbarred lines (++++++) mean that sides have been glued; broken lines (- - - -) are used to indicate the sides of holes.

In his paper (ref. 15), Christer Bäckström presents axioms which characterize the static properties of this world in first-order loigc, as well as the axioms



A body through a hole in another body. A bolt Representation of a bolt junction.

Figure 12.8. *2-D representation of geometrical objects.*

which characterize the move operation in the world, using a variant of dynamic logic. The conclusion of the work is that formalization was possible using this approach, but some of the resulting axioms are a bit messy. One of the objectives of the continued work will be to find an alternative approach which allows 'neater' axioms.

12.5 References.

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

Papers in international journals or conference proceedings, published or accepted since the beginning of 1986.

1. **Dimiter Driankov**, An outline of a fuzzy sets approach to decision-making with interdependent goals, *Proc. of the First IFSA Congress,* Palma de Mallorca July, 1985. To appear in *Fuzzy Sets and Systems, An Int. Journal.*

2. **Dimiter Driankov**, Inference with single fuzzy conditional proposition, to appear in *Fuzzy Sets and Systems*, (1987).

3. **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 1986.

4. **Dimiter Driankov**, Many-valued logic for belief-intervals: The logical lattice. To appear in *Proc. of the Second World Congress of the Int. Fuzzy Sets Association,* Tokyo, July 20-25, 1987.

5. **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.

6. **Lin Padgham:** LINCKS Linköpings Intelligent Knowledge Communication System. (Revised Version). In *Proc of I.F.I.P. Conference on Methods and Tools for Office Systems*, Pisa, Italy, October 22-24, 1986.

7. **Lin Padgham, Ralph Rönnquist**: An Imperative Object Oriented System. *Proc.*

of the 20th Hawaii International Conference on System Sciences, 1987, vol 1, p 516.

8. **Lin Padgham, Ralph Rönnquist:** From a Technical to a Humane Environment: A Software System Supporting Co-operative Work. *Proc. of the GDI International Conference on USER INTERFACES*, Rüschlikon, Switzerland, 20-21 Oct, 1986.
9. **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 (to appear.)
10. **Michael Reinfrank:** Reason Maintenance Systems. In Herbert Stoyan (ed.) *Proc. of the Workshop on Truth Maintenance*, Berlin, 1986. Springer Verlag (to appear.)
11. **Erik Sandewall, Ralph Rönnquist:** A Representation of Action Structures. In *Proc. of the 5th National Conf. on Artificial Intelligence, AAAI-86, Philadelphia, 1986.*
12. **Erik Sandewall:** Non Monotonic Inference Rules for Inheritance with Exception. In *Proc. of the IEEE, Special Issue on Knowledge Representation.* 1986.
13. **Erik Sandewall:** Specification Environments for Information Management Systems. Panel position paper in *Proc. IFIP Congress 1986.*

Departmental reports:

14. **Ralph Rönnquist:** The Information Lattice of Networks Used for Knowledge Representation. LiTH-IDA-R-86-02
15. **Christer Bäckström:** Logical Modelling of Simplified Geometrical Objects and Mechanical Assembley Processes. LiTH-IDA-R-87-05
16. **Johan Hultman:** COPPS - A Software System for Defining and Controlling Actions in a Mechanical System. LiTH-IDA-R-87-06
17. **Peter Haneclou:** A Formal Approach to Reason-maintenance Based on Abstract Domains. LiTH-IDA-R-87-07

Other papers referenced above:

18. **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.
19. **Jalal Maleki:** ICONStraint, a Dependency Directed Constraint Maintenance System. Licentiate Thesis No. 71, Linköping University (forthcoming).
20. **Erik Sandewall:** The Pipelining Transformation on Plans for Manufacturing Cells with Robots. Manuscript.

13.

ADP Administrative Data Processing

Göran Goldkuhl

13.1 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.

13.2 Research activities.

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

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

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

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

13.3 Personnel:

Göran Goldkuhl, PhD, senior lecturer
Eva-Chris Svensson, MSc, director of undergraduate studies
Carina Björkman, secretary
Siv Söderlund, secretary

Johan Eltes, assistant
Hans Holmgren, MScE
Rolf Nilsson, BSc, lecturer
Torbjörn Näslund, assistant
Lise-Lotte Raunio, lecturer
Annie Röstlinger, BSc , lecturer
Dan Strömberg, MScE, lecturer (now at Foa)
Roger Zollner, lecturer
Elisabeth Zsiga, 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 takes decisions about 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 Barbara Ekman as secretary, is responsible for the organization of undergraduate courses and continuing education for industry. Most of the 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

Department of Computer and Information Science

Organization:

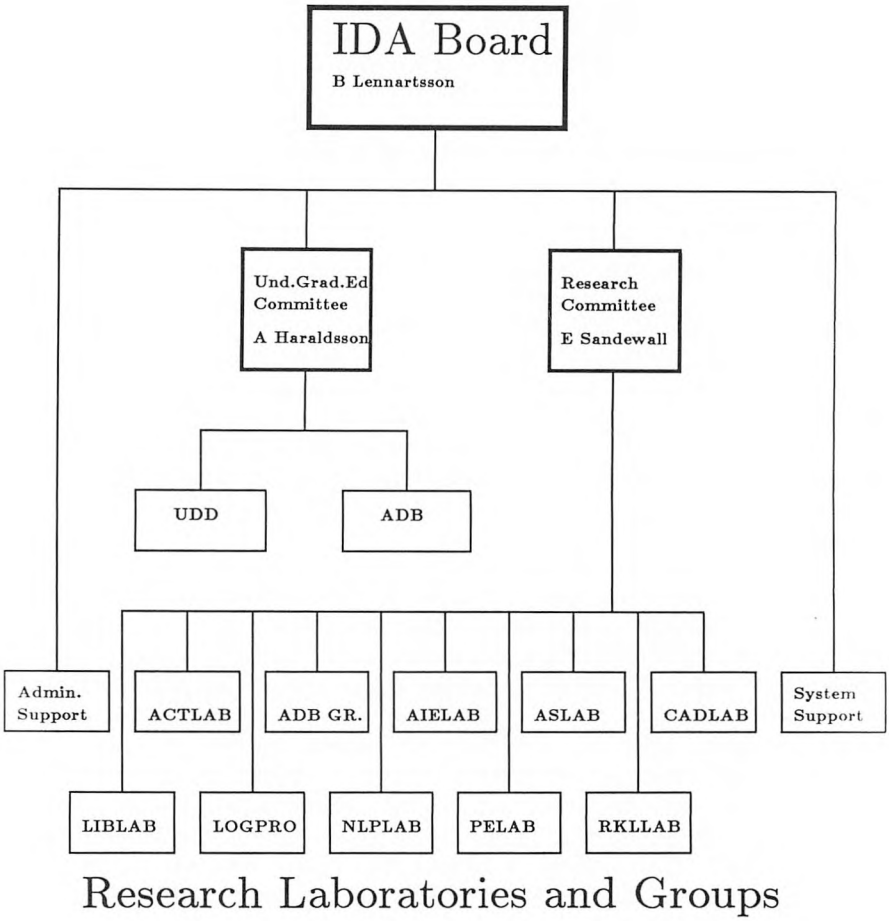


Figure A.1. *Administrative organization of the department.*

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 Eva-Chris Svensson for those in the School of Arts and Science.

Formally all significant administrative decisions 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 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 1986/87 balances at 23.1 MSEK. (One MSEK is at present approximately 0.15 USD.) Of this sum, the resources for undergraduate education supplied by the university amount to 10.4 MSEK, and corresponding resources for research and graduate education are 3.0 MSEK. The research activities are thus heavily dependent on external sources, where the Swedish Board for Technical Development, STU, is the main contributor (86/87: 5.4 MSEK). Additional funds are provided by the Delegation for Technical and Scientific Information Supply, DFI, (86/87: 0.9 MSEK) and the Natural Science Research Council, NFR, (86/87: 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 3 MSEK. Commissioned education programmes for industry are budgeted at about 1 MSEK 1986/87.

Costs for office space and investment in equipment are not included in the above figures. During 1986 the department installed computer equipment at an approximate value of 12 MSEK. Thus for instance, the value of the AI workstations grant from Rank-Xerox Corporation was estimated to be in the order of 8 MSEK.

Department leadership:

Bengt Lennartsson, department chairman

Erik Sandewall, research committee chairman
Anders Haraldsson, undergraduate education committee chairman
Inger Emanuelson, administrative manager

Administrative office:

Inger Emanuelson, administrative manager
Lillemor Wallgren, secretary of graduate education

Lena Wigh, office assistant

Technical services:

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

Leif Finmo, research engineer
Dimitrios Fotiadis, research engineer
Arne Fäldt, senior research engineer
Claes Illergård, 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.

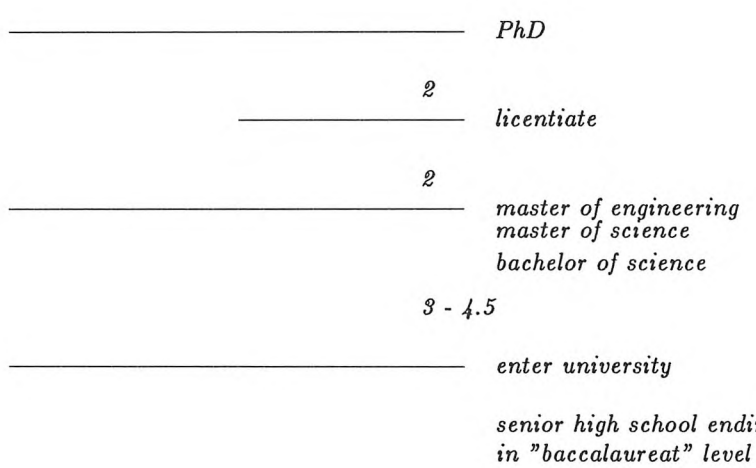
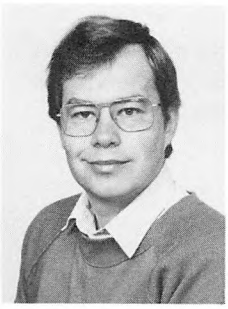


Fig B.1. *Levels of degrees*

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 3.

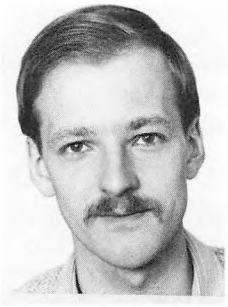
Graduate studies in the department of Computer and Information Science are organized as a program consisting of courses and project participation. The course program is organized at the department level and consists of *basic courses*, each of which is given every third year (if possible), and *occasional 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 engaged in graduate study program.

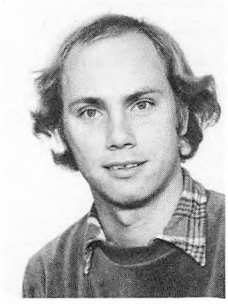


Ahrenberg, Lars, BA. PhD spring 1987, (previous affiliation Uppsala and Göteborg) researcher. Group leader, NLPLAB. Natural language processing, computational linguistics, user interfaces.

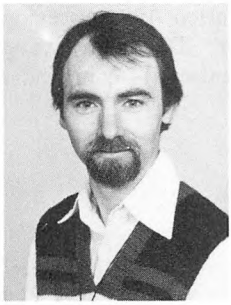
Douglas Busch, PhD (Rockefeller 1973, associate professor in logic and theoretical computer science. Previous affiliation Mcquarie University, Sydney, Australia). Application of theories from formal logic to problems in theoretical computer sience and artificial intelligence; algebraic specification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



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



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

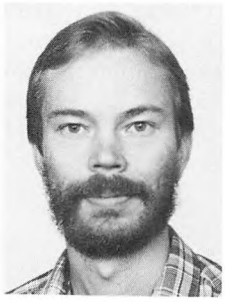


Göran Goldkuhl, PhD (Stockholm 1980, previous affiliation Göteborg), senior lecturer. Group leader in ADP research. Information requirement analysis, behavioral aspects of information systems, research methodologies, information systems and quality of working life.

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

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

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



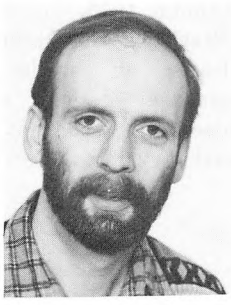
Rolf Karlsson, PhD (Waterloo 1984, previous affiliation Lund), researcher. Data structures, algorithm analysis, computational complexity, computational geometry.

Krzysztof, Kuchcinski, PhD (Gdansk 1984, on leave from Institute of Computer Science, Politechnika Gdanska), researcher. Group leader, CADLAB. Computer architecture, CAD, real-time operating systems, system testing.

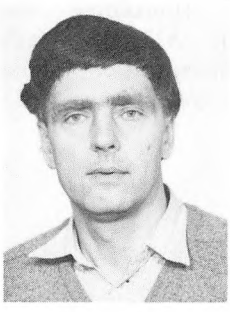
Harold W. Lawson Jr., PhD (Stockholm, several previous affiliations, also in industry), professor of telecommunication and computer systems. Computer architecture, VLSI, Computer-aided design, methodology of computer-related education and training.

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

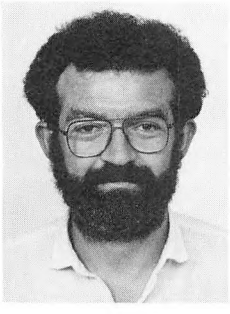
Christos Levcopulos, PhD (Linköping 1987), researcher. Computational geometry, analysis of algorithms, data structures.



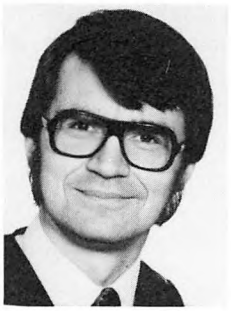
Andrzej Lingas, PhD (Linköping 1983, previous affiliation Warszawa and MIT), docent. Group leader in geometric complexity. Complexity theory, analysis of algorithms, geometric complexity, graph algorithms, logic programming, VLSI theory.



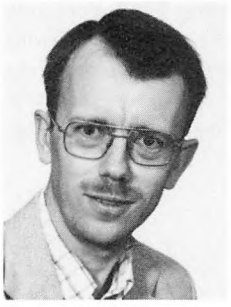
Jan Maluszynski, PhD (Warszawa 1973, several previous affiliations), acting professor and group leader in theoretical computer science. Logic programming, software specification methods.



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



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



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

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

Graduate Study Course Program 1985-86

Basic and Occasional Graduate Courses:

Theory of systems development (Göran Goldkuhl)

Non-Standard Logics for Artificial Intelligence (Erik Tengvald)

Software Engineering (Benny Odenteg, Kevin Ryan)

Search structures (Rolf Karlsson)

Computer Architecture / VLSI (Harold W Lawson)

Change analysis (Göran Goldkuhl)

Knowledge organization (Roland Hjerppe)

Analysis and Complexity of Parallel Algorithms (Andrzej Lingas)

Research-Related Courses and Seminars:

Debugging of programs with parallel processes (Bengt Lennartsson)

The AIM project (Erik Tengvald)

The HYPERCATalog project (Roland Hjerppe)

Authority control (Roland Hjerppe)

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

AI and software engineering (Spring) (Sture Hägglund)

Statistical Information Systems (Bo Sundgren)

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

Future CAD Systems (Harold W Lawson)

Logic Programming (Jan Maluszynski)

Complexity of algorithms (Andrzej Lingas)

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

Special Courses for the Knowledge Transfer Program:

Knowledge Engineering with EMYCIN. (Kristian Sandahl)

Introduction to KEE. (Roland Rehmnert)

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)

A Selection of Seminars 1986

General seminars spring 1986

14/1 Bertil Rolf, Lund. Ur logikens historia

20/1 Håkan Jacobsson, IDA. AND/OR parallelism in logic programs without backtracking

21/1 Seif Haridi, SICS. A constructive theorem prover and its application in logic programming

30/1 Kevin Ryan, Ida. Report from ToolUse

4/2 Göran Hagert, Uppsala. AI och kognitiv modellering

13/2 Karl-Erik Årzen, LTH. Expertsystem för reglerteknik

18/2 Johan Fagerström, IDA. Simulation and Evaluation of an Architecture based on Asynchronous Processes

25/2 Jonas Barklund, UPMAIL. En reviderad abstrakt prologmaskin och dess implementation

3/3 Ulf Nilsson, LiTH. AID, an alternative implementation of DCG

5/3 Vojin Plavsic, FOA3. Optical Computing

11/3 Mark Overmars, Utrecht. Range Searching on a Grid

18/3 Boris Magnusson, Lund. IPDS: Interactive Program Development

21/3 Torbjörn Molin, UPMAIL. Icke-monotona resonemang eller Tweetys hämnd

8/4 Paul O Fredricksson. Highly Parallel Supercomputer

10/4 Jozef Olenski, Warszawa. Statistiska informationsspråk och thesaurus-baserad utveckling av integrerade metadatabassystem

15/4 Thomas Strothotte, Univ. Stuttgart. Structural Properties of the Heap

21/4 Marek Kubale, Univ of Gdansk. Approximate Scheduling Independent Two-Processor Tasks with dedicated Processors

21/4 Håkan Jakobsson, IDA. An implementation of AND parallelism

22/4 John McDermott. Where will knowledge acquisition tools lead

22/4 Anna Hart. Dealing with uncertainty - Can we cope?

13/5 Toomas Käer, Bengt Lennartsson, IDA. Programutvecklingssystemet PUS-80

23/5 Tony Larsson, IDA. On the Specification and Verification of VLSI Systems

27/5 Henning Christiansen, Roskilde Univ. Context Sensitive Parsing in full Prolog

29/5 Wlodek Grudzinski, Univ. Warszawa. A Database Support System for Prolog

9/6 Henryk Jan Komorowski, Harvard Univ. Declarative Programming Environment

13/6 Peter D Holmes, former TRW Defense Systems. A Method for Explanation Truth Maintenance and Rule Activation within the ROSIE Programming Environment

24/6 Vincenzo Ambriola, Pisa. Semantics-directed Compilation using

General seminars fall 1986

12/8 Zbigniew Michalewicz, Wellington, New Zealand. Are Statistical Databases safe?

19/8 Ludwik Czaja, Computer Science Dept., Univ. of Warsaw, Poland Cause - Effect Structures.

21/8 Z Ras, Univ. of Knoxville, Tennessee. On Methodologies for Knowledge Representation and KNowledge Acquisition

4/9 Per-Erik Håll, Rank Xerox. Kontorssystemet Viewpoint

9/9 Peter Fritzson, IDA. Ett och ett halvt år på SUN - Vad händer nu och i framtiden

9/9 Erik Sandewall, IDA. AAAI '86 and NMR Theory

19/9 Christos Levcopoulos, IDA. New Results about the Approximation Behavior of the Greedy Triangulation

22/9 Ola Strömfors, IDA. A Structure Editor for Documents and Programs

23/9 Bo Dahlbom, Göteborgs och Umeå Univ. Datorer, förnuft och etik

30/9 Marek Karpinski, Univ of Bonn and Math. Science Res. Inst., Berkeley What are the Limits of Superfast Parallelization?

1/10 Marek Karpinski, On Probabilistic Space Complexity

9/10 Douglas Busch, IDA. Combining Logic and Functional Programming

14/10 Brian Clark, Firma Carl Lamm. Unixsystemet Pyramid och en RISC-maskin

21/10 Ken Kahn. Vulcan: Logical Concurrent Objects

29/10 K.G. Wigander. Programators systemutvecklingsmodell ROS

30/10 Keith Geddes, Univ of Waterloo. The MAPLE Computer Algebra System

11/11 Jon-Olof Hugozon, Samhällsvetenskapliga Inst. Hur fungerar en bra arbetsgrupp

14/11 Göran Goldkuhl, IDA. Farväl till systemansats och V-grafer

19/11 Östen Dahl, Stockholms Univ. Problem i syntaktisk och semantisk parsning

24/11 Ian Munro, Univ. of Waterloo. Implicit data Structures

27/11 Jörgen Gustavsson, Roger Larsson, IDA. Ada in distributed Systems

3/12 Jerker Wilander, Softlab. Distribuerade databaser, speciellt frågeoptimering

11/12 Carl-Martin Allwood, Göteborgs och Linköpings Univ. Effekter av användarnas bakgrundskunskaper vid användning av applikationsprogram

17/12 Per Svensson, Foa. En systemarkitektur för effektiv utvärdering av statistiska frågor

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. A first 2 year course has started during 1985 for 20 students at Ericsson, Stockholm.

Courses. During 86/87 IDA will give a total of approx 75 (70) different courses. In parentheses the figures from 85/86. In the engineering study programs IDA gives 51 (50) courses with a total of 3300 (3500) students, 10 (10) single-subject courses, and about 14 (10) external courses for industry with about 500 (400) 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. This explains the decrease in number of students. 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.

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 86/87 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. 6 teachers from other subjects and from industry
c. 40 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, B.A. (PhD, spring 87)
Rober Bilos, MSc
Douglas Busch, PhD
Nils Dahlbäck, B.A.
Patrick Doherty
Wlodek Drabent, PhD
Johan Fagerström, MSc
Björn Fjellborg, MSc
Göran Goldkuhl, PhD
Christian Gnosspelius
Anders Haraldsson, PhD
Sture Hägglund, PhD
Arne Jönsson, MSc
Rolf Karlsson, PhD
Christian Krysander, MSc
Krzysztof Kuchcinski, PhD
Bengt Lennartsson, PhD
Peter Loborg
Jalal Maleki, MSc
Jan Maluszynski, PhD
Magnus Merkel, B.A.
Henrik Nordin. MSc
Kerstin Olsson, MSc

Tommy Olsson, MSc
Mikael Patel, MSc
Zebo Peng, MSc
Ivan Rankin, BSc
Roland Rehmnert, MSc
Kevin Ryan, PhD
Erik Sandewall, PhD
Nahid Shahmehri, MSc
Ola Strömfors, MSc
Katarina Sunnerud, MSc
Eva-Chris Svensson, BSc
Lars Wikstrand, BSc
Olle Willen, BSc
Mats Wiren, MSc
Per Övernäs, BSc

Listing of Undergraduate Course Program 1986/87

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)	Göran Goldkuhl
AI-programmering (C4)	Jalal Maleki
Logik, grundkurs (C1, D4)	Erik Sandewall
Psykologi grundkurs (C2)	Kjell Olsson, Dept of Education and Psychology
Naturligt språkbehandling (C4)	Mats Wiren
Operativsystem (D3tk, D4, Y4, I4)	Ola Strömfors
Algoritm och komplexitetsteori (C4)	Rolf Karlsson
Programmerung Y, fortsättningskurs (Y3, Y4)	Kristian Ernstsson
Kompilatorer och interpretatorer (Y4, I4)	Nahid Shahmehri
Programmeringsteori II (C4)	Douglas Busch
Administrativ databehandling (Y4, I4)	Eva-Chris Svensson
Cobol (Y4, I4)	Per Övernäs
Logikprogrammering (C3, D4)	Jan Maluszynski
Programmeringsteori (C3)	Wlodek Drabent
Processprogrammering (D3pv, D4, Md4, Y4, I4)	Mikalel Patel
Operativsystemteori (C3, D3pv, M4d)	Ola Strömfors
Programmering och projektarbete i Pascal (C1, D1)	Peter Loborg
Lagringssstrukturer (C2, D2, Md3)	Kristina Ernstsson
Assemblyprogrammering (C2, D2, Md3, I4)	Rober Bilos
Programutvecklingsmetodik (D2, Md3)	Christian Krysander
Programutvecklingsmetodik och programmeringsprojekt D (D3)	Christian Krysander
Programutvecklingsmetodik M (Md3)	Olle Willen
Programutvecklingsmetodik och programmeringsprojekt C (C3)	Henrik Nordin
Data och programstrukturer D (D3)	Anders Haraldsson
Data och programstrukturer C (C2)	Roland Rehmnert
Distribuerad problemlösning (D4, C4)	Johan Fagerström

Datorer och datorutrustning (I1)	Christian Gnosspelius
Fördjupningskurs i komplexitetsteori (D4)	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)	Arne Jönsson
Datalingvistik (C2)	Lars Ahrenberg
Programmering i inkrementellt system (C1)	Anders Haraldsson
Programmering i inkrementellt system (D1)	Anders Haraldsson
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)	Krzysztof Kuchinski
Datorstödd elektronikkonstruktion (D4, Y4, I4)	Bengt Magnhagen, DIGSIM AB
Datalogi 1 - baskurs (enstaka kurs Linköping)	Kerstin Olsson
Datalogi 2 - Programmeringsprinciper (enstaka kurs Linköping)	Tommy Olsson
Programmering i Ada (enstaka kurs Norrköping)	Olle Willen
Programmering i LISP(enstaka kurs Linköping)	Patrick Doherty
Artificiell intelligens(enstaka kurs Linköping)	Ivan Rankin

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.

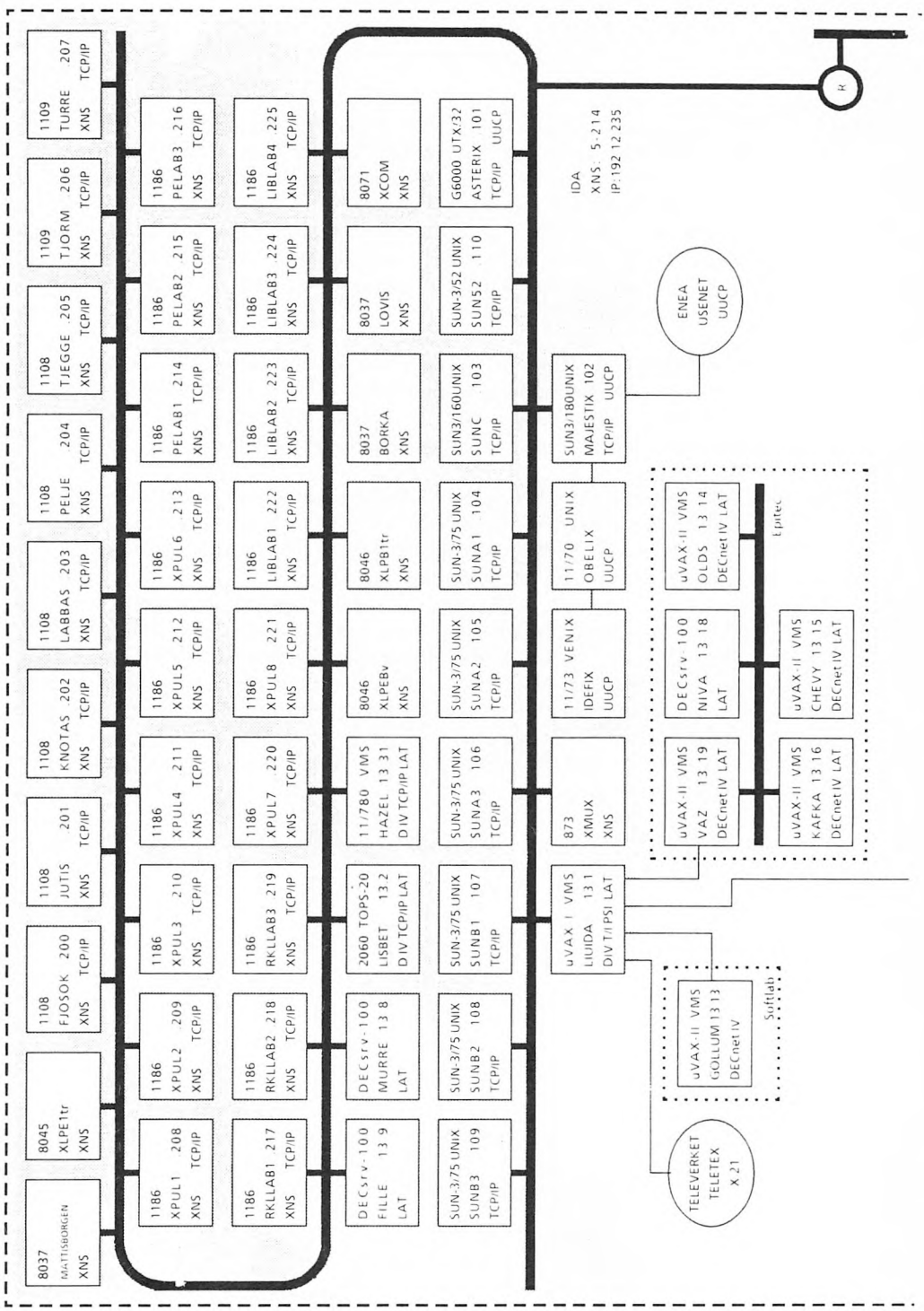
The department has a policy of giving high priority to the supply of appropriate computing resources for research and education. We have also during the years been able to modernize and keep in pace with the rapid development in the area, e.g. regarding the emergence of powerful workstations with high-resolution 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) and a Xerox Ethernet with twentysix 1108/1109/1186 Lisp Machines, file servers and laser printers.

We have also recently acquired seven SUN 3 workstations. 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.

A large 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 861219
8037 MATTISHORGEN XNS
8045 XLPE1tr XNS
1108 FJOSOK 200 XNS TCP/IP
1108 JUTIS 201 XNS TCP/IP
1108 KNOTAS 202 XNS TCP/IP
1108 LABBAS 203 XNS TCP/IP
1108 PELJE 204 XNS TCP/IP
1108 TJEGGE 205 XNS TCP/IP
1109 TJORM 206 XNS TCP/IP
1109 TURRE 207 XNS TCP/IP
1186 XPUL1 208 XNS TCP/IP
1186 XPUL2 209 XNS TCP/IP
1186 XPUL3 210 XNS TCP/IP
1186 XPUL4 211 XNS TCP/IP
1186 XPUL5 212 XNS TCP/IP
1186 XPUL6 213 XNS TCP/IP
1186 PELAB1 214 XNS TCP/IP
1186 PELAB2 215 XNS TCP/IP
1186 PELAB3 216 XNS TCP/IP
1186 RKLLAB1 217 XNS TCP/IP
1186 RKLLAB2 218 XNS TCP/IP
1186 RKLLAB3 219 XNS TCP/IP
1186 XPUL7 220 XNS TCP/IP
1186 XPUL8 221 XNS TCP/IP
1186 LIBLAB1 222 XNS TCP/IP
1186 LIBLAB2 223 XNS TCP/IP
1186 LIBLAB3 224 XNS TCP/IP
1186 LIBLAB4 225 XNS TCP/IP
DECsrv-100 FILLE 139 LAT
DECsrv-100 MURRE 138 LAT
2060 TOPS-20 LISBET 132 DIV TCP/IP LAT
11/780 VMS HAZEL 1331 DIV TCP/IP LAT
8046 XLPEBv XNS
8046 XLPB1tr XNS
8037 BORKA XNS
8037 LOVIS XNS
8071 XCOM XNS
SUN-3/75 UNIX SUNB3 109 TCP/IP
SUN-3/75 UNIX SUNB2 108 TCP/IP
SUN-3/75 UNIX SUNB1 107 TCP/IP
SUN-3/75 UNIX SUNA3 106 TCP/IP
SUN-3/75 UNIX SUNA2 105 TCP/IP
SUN-3/75 UNIX SUNA1 104 TCP/IP
SUN3/160UNIX SUNC 103 TCP/IP
SUN-3/52 UNIX SUN52 110 TCP/IP
G6000 UTX/32 ASTERIX 101 TCP/IP UUCP
uVAX I VMS LIUIDA 131 DIV T/I PSI LAT
873 XMUX XNS
11/73 VENIX IDEFIX UUCP
11/70 UNIX OBELIX UUCP
SUN3/180UNIX MAJESTIX 102 TCP/IP UUCP
IDA XNS 5-214 IP:192 12 235
ENEA USENET UUCP
uVAX-II VMS VAZ 13 19 DECnet IV LAT
DECsrv 100 NIVA 13 18 LAT
uVAX-II VMS OLDS 13 14 DECnet IV LAT
uVAX-II VMS KAFKA 13 16 DECnet IV LAT
uVAX-II VMS CHEVY 13 15 DECnet IV LAT
uVAX-II VMS GOLLUM 13 13 DECnet IV
Softlab
TELEVERKET TELETEX X.21

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.

(*Thesis 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, Mickael Patel:** An Interactive Technique for Communicating and Realizing Algorithms. 1984.

No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator. 1984.

No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching. 1985.

No 52 **Zebo Peng:** Steps towards the Formalization of VLSI Design Systems, 1985.

No 60 **Johan Fagerström:** Simulation and Evaluation of an Architecture based on Asynchronous Processes. 1986.

No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems. 1986.

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.

EXTERNAL PUBLICATIONS 1980-

(*Papers published in books, journals or international conference proceedings.*)

1. **Lars Ahrenberg:** Lexikalisk-Funktionell Grammatik på svenska. In *Papers from the Fifth Scandinavian Conference of Computational Linguistics,* University of Helsinki, Dept of General Linguistics pp. 1-12, 1986.
2. **Rober Bilos:** A Token-Based Syntax Sensitive Editor. *Proc of the Workshop on Programming Environments - Programming Paradigms.* Roskilde, 1986.
3. **Shamsul Chowdhury:** Expert System Aid in Statistical Analysis and Interpretation of Data. In *Proc. of the Society of Reliability Engineers,* Outaniemi, 1986.
4. **James A. Dean, Andrzej Lingas and Jörg R. Sack:** Recognizing Polygons or How to Eavesdrop. In *Proc. of the Allerton Conference on Communication, Control, and Computing,* Urbana, Illinois, 1986.
5. **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.
6. **Pierre Deransart, Jan Maluszynski:** Relating Logic Programs and Attribute Grammars. *Journal of Logic Programming, vol 3,* No. 2, pp 119-158, 1985.
7. **Pierre Deransart and Jan Maluszynski:** Modelling Data Dependencies in Logic Programs by Attribute Schemata. Published as *INRIA, Report RR323,* 1984.
8. **Wlodzimierz Drabent, Jan Maluszynski:** Proving Run-Time Properties of Logic Programs. To appear in *Proc. of TAPSOFT 87,* Pisa, 1987.
9. **Dimiter Driankov,** An outline of a fuzzy sets approach to decision-making with interdependent goals, *Proc. of the First IFSA Congress,* Palma de Mallorca July, 1985. To appear in *Fuzzy Sets and Systems, An Int. Journal.*
10. **Dimiter Driankov,** Inference with single fuzzy conditional proposition, to appear in *Fuzzy Sets and Systems,* (1987).
11. **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 1986.
12. **Dimiter Driankov,** Many-valued logic for belief-intervals: The logical lattice. To

appear in *Proc. of the Second World Congress of the Int. Fuzzy Sets Association*, Tokyo, July 20-25, 1987.

13. **Johan Elfström, Jan Gillqvist, Hans Holmgren, Sture Hägglund, Olle Rosin, Ove Wigertz:** A Customized Programming Environment for Patient Management Simulations. *Proc. of the 3rd World Conf. on Medical Informatics*, Tokyo, 1980.
14. **Pär Emanuelson, Anders Haraldsson:** On Compiling Embedded Languages in Lisp. *Proc. of the 1980 LISP Conf.*, Stanford, Calif, 1980.
15. **Pär Emanuelson:** From Abstract Model to Efficient Compilation of Patterns. *Proc. of the 5th Int. Conf. on Programming*, Turin, 1982. Revised version accepted for publication in *Science of Computer Programming.*
16. **Johan Fagerström:** Experiences with Occam: A Simulator for Asynchronous Processes. *Proc 19th Hawaii Int. Conf. on System Sciences, Hawaii,* Jan. 1986, pp. 95-102.
17. **Johan Fagerström:** Tradeoffs in an Architecture based on Asynchronous Processes. In *Proc 2nd Nordic Symposium on VLSI in Computers and Communications*, 1986.
18. **Johan Fagerström, Mikael R.K. Patel:** High-level Simulation of Systolic Architectures. In *Proc of the International Workshop on Systolic Arrays,* Oxford, 2-4 July, 1986.
19. **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.
20. **Peter Fritzson:** A Systematic Approach to Advanced Debugging through Incremental Compilation. *Proc of the ACM SIGSOFT/SIGPLAN Symposium on High-Level Debugging,* Pacific Grove, CA., March 1983.
21. **Peter Fritzson:** Symbolic Debugging through Incremental Compilation in an Integrated Environment. *The Journal of Systems and Software* 3, 285-294, (1983).
22. **Peter Fritzson:** Preliminary Experience from the DICE System - A Distributed Incremental Compiling Environment. *Proc. of the ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments,* Pittsburgh, PA. April 1984.
23. **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.
24. **Peter Fritzon:** 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.
25. **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.
26. **James W. Goodwin:** Why Programming Environments Need Dynamic Data Types. *IEEE Trans. Software Eng., vol SE-7,* no 5, 1981. Also in Barstow et al. (eds.) *Interactive Programming Environments*, McGraw-Hill, 1984.
27. **James W. Goodwin and Uwe Hein:** Artificial Intelligence and the Study of Language. *Journal of Pragmatics, 6,* pp 241-280, North-Holland, 1982.
28. **James W. Goodwin:** WATSON - A Dependency Directed Inference System, In *Proc. of the AAAI Workshop on Non-Monotonic Reasoning*, New Palz, NY, 1984.
29. **James W. Goodwin:** A Process Theory of Non-Monotonic Inference. in *Proc. of the Int. Joint Conf. on Artificial Intelligence, IJCAI,* 1985.
30. **Sture Hägglund:** Dialogue Models for Human-Computer Communication. A Practitioner's View. in *Proc. of the Workshop on Models of Dialogue: Theory and Application.* Linköping 1981.
31. **Sture Hägglund, Johan Elfström, Hans Holmgren, Olle Rosin, Ove Wigertz:** Specifying Control and Data in the Design of Educational Software. *Computers & Education, vol 6,* no 1, 1982.

32. **Sture Hägglund, and Roland Tibell:** Multi-Style Dialogues and Control Independence in Interactive Software. In Green et al. (eds.) *The Psychology of Computer Use*, Academic Press, 1983. Previous version in *Proc. of the 1st European Conf. on Cognitive Engineering*, Amsterdam, 1982.
33. **Sture Hägglund:** On the Design of a Query Environment for Office Use. *Proc. of the 2nd Scandinavian Seminar on Information Modelling and Database Management*, Tampere, 1983.
34. **Uwe Hein:** Interruptions in Dialogue. Also in D. Metzing (ed), *Dialogmuster und Dialogprozesse*. Hamburg, Buske, 1981.
35. **Uwe Hein:** Natural and Artificial Communications. - Some Reflections -. in *Proc. of the Workshop on Models of Dialogue: Theory and Application*. Linköping 1981.
36. **Uwe Hein:** Constraints and Event Sequences. *Proc of the NATO symp. on Artificial Intelligence*, Lawrence Erlbaum, 1982.
37. **Uwe Hein:** PAUL - A Programming Language for Knowledge Engineering Applications. *Proc. of the International Conference on Artificial Intelligence*, Leningrad, October 1983.
38. **Roland Hjerppe:** What artificial intelligence can, could, and can't, do for libraries and information services. *Proc. 7th IOLIM,* Learned Information Ltd. London. December 1983.
39. **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 Conference,* Chicago, 18-24 August, 1985.
40. **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.*Proc. IRFIS 6 Conference* (International Research Forum in Information Science), Frascati, Italy, 15-18 Sept. 1985, Elsevier Science Publishers B.V. (North-Holland), 1986. pp.211-232)
41. **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.
42. **Roland Hjerppe:** Knowledge Organizing, Collection Derived, and User Established Structures. Published in Online Public Access to Library Files:*Second National Conference*. J Kinsella Ed. Elsevier International Bulletins, Oxford 1986 pp. 101-110).
43. **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.
44. **Hans Karlsson, Roland Lindvall, Olle Rosin, Erik Sandewall, Henrik Sörensen and Ove Wigertz:** Experience from Computer Supported Prototyping for Information Flow in Hospitals. *Proc. of the ACM SIGSOFT Second Software Engineering Symposium: Workshop on Rapid Prototyping,* Columbia, Maryland, April 19-21, 1982.
45. **Hans (Karlsson) Gill, Bertil Kågedahl, Erik Sandewall, Henrik Sörensen, Lennart Tegler, and Ove Wigertz:** A Notation for Information Flow Models Supporting Interactive System Development. *Proc of the 6th Annual Symposium on Computer Applications in Medical Care*, Washington DC, nov 1982.
46. **Rolf G. Karlsson, Ian Munro,** Proximity on a Grid, in the *Proc. of 2nd Symposium on Theoretical Aspects of Computer Science* (1985), Springer-Verlag Lecture Notes on Computer Science 182, 187-196.
47. **Rolf G. Karlsson, Ian Munro, Ed Robertson,** The Nearest Neighbor Problem on Bounded Domains, in the *Proc. of 12th Int. Colloquium on Automata, Languages and Programming* (1985), Springer-Verlag Lecture Notes on Computer Science 194, pp 318-327.
48. **Rolf G. Karlsson:** Point Location in Discrete Computational Geometry. *Proc. 6th Brazilian Congress on Computing,* 1986, 561-569.
49. **H Jan Komorowski, Jan Maluszynski:** Logic Programming and Rapid

Prototyping. Also Published as report TR-01-86, Harward University, Aiken Computation Laboratory. To appear in *Science of Computer Programming.*

50. **H. Jan Komorowski:** QLOG - The Software for Prolog and the Logic Programming. *Proc. of the Logic Programming Workshop*, Debrecen, Hungary, 1980. Also in Clark, Tärnlund (eds.) *Logic Programming*, Academic Press, 1982.
51. **H. Jan Komorowski:** Partial evaluation as a means for inferencing data structures in an applicative language: a theory and implementation in the case of Prolog. *Proc of the Symp. on Principles of Programming Languages*, Albuquerque, 1982.
52. **H. Jan Komorowski:** An Abstract Prolog Machine. *Proc. of the European Conf. on Integrated Interactive Computing Systems*, Stresa, 1982.
53. **H. Jan Komorowski:** A Prototype Compiler for Prolog. Poster version presented at *the 6th Int. Conf. on Software Engineering*, Tokyo, 1982.
54. **Krzysztof Kuchcinski and Zebo Peng:** Microprogramming Implementation of Timed Petri Nets, *Proc. 2nd Nordic Symp. on VLSI in Computers and Communications,* Linköping, Sweden, June 1986.
55. **Tony Larsson:** Semantics of a Hardware Specification Language and Related Transformation Rules *Proc. 2nd Nordic Symp. on VLSI in Computers and Communications,* Linköping, Sweden, June 1986.
56. **Tony Larsson:** Semantics of a Hardware Specification Language, *Microprocessing and Microprogramming,* Vol.18, Nos 1-5, 1986, pp. 335-340.
57. **Harold W. Lawson Jr.:** New Directions in Micro- and System Architecture in the 1980's, in *Proc. of the National Computer Conference, NCC-81,* Chicago, Ill., 1981.
58. **Harold W. Lawson Jr.:** An Approach to Improving Computer Literacy, in *Teaching Informatics Courses: Guidelines for Trainers and Educationalists*, (ed. by A.L.W. Jackson), North-Holland, 1982.
59. **Harold W. Lawson Jr.:** The Holistic Approach in Introducing Computer Systems, in *The Computing Teacher,* vol 10, no 7, October 1982. Also in Japanese translation in Nikkei-Computer, Niekkei-McGraw-Hill, Tokyo, 1982.
60. **Harold W. Lawson Jr.:** An Architecture-Based Strategy for Improving Computer Education, in *Proc. of the Euromicro 82 Symposium*, Brussels, September 1982.
61. **Harold W. Lawson Jr.:** Some Consequences of Tomorrows Electronics CAD Systems, in *Proc. of Mantech 83, Discoveries Int. Symp.*, London, 1983.
62. **Harold W. Lawson Jr.:** Computer architecture education, a chapter in Tiberghien (Ed.): *New Computer Architectures*, pp 224-285, Academic Press, 1984.
63. **Harold W. Lawson Jr.:** Impact of CAD and Integrated Circuit Developments on Telecommunication. *Proc. of the EUTECO Conference,* Oct 1983, Varese, Italy.
64. **Harold W. Lawson Jr.:** Ingrediants and Implications of Tomorrows CAD Systems. *Integrated Circuit Seminar,* July 18-22 1983, Singapore.
65. **Harold W. Lawson Jr.:** Architecting VLSI Systems. *Integrated Circuit Seminar,* July 18-22 1983, Singapore.
66. **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.
67. **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.
68. **Harold W. Lawson, Bryan Lyles:** An Architectural Strategy for Asynchronous Processing. *IFIP Workshop*, March 26-28, 1984, Bristol.
69. **Bengt Lennartsson:** Programming Environments and Paradigms - Some Reflections. In *Proc. of the Workshop on Programming Environments - Programming Paradigms*, Roskilde, Denmark, October 1986.
70. **Christos Levcopoulos, Andrzej Lingas:** Covering Polygons with Minimum Number of Rectangles, *Proc. of the STACS Symposium,* Paris (1984), *Lectures Notes in Computer Science,* vol 166, Springer Verlag.
71. **Christos Levcopoulos:** On Covering Regions with Minimum Number of Rectangles, *Proc. of the Workshop on Parallel Computing and VLSI*, Amalfi, Italy, (1984) North-Holland Publ. Co.

72. **Christos Levcopoulos, Andrzej Lingas:** Bounds on the Length of Convex Partitions of Polygons, in the *Proc. of the 4th FST-TCS Conference*, Bangalore, India, (1984), *Lectures Notes in Computer Science,* vol 181, Springer Verlag.
73. **Christos Levcopoulos,** Minimum Length and "Thickest-First" Rectangular Partitions of Polygons, in the *Proc. of the 23rd Allerton Conf. on Comm., Control and Computing,* Illinois, October 1985.
74. **Christos Levcopoulos,** A Fast Heuristic for Covering Polygons with Rectangles, in the *Proc. of 5th Int. Conf. on Foundations of Computation Theory*, GDR, (1985), *Lectures Notes in Computer Science,* vol 199, Springer Verlag.
75. **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.
76. **Christos Levcopoulos:** An Omega (square root(n)) Lower Bound for the Non-Optimality of the Greedy Triangulation. To appear in *Information Processing Letters,* (1987).
77. **Christos Levcopoulos, Andrzej Lingas:** On the Approximation Behavior of the Greedy Triangulation for Convex Polygons. To appear in *Algorithmica,* 1987
78. **Andrzej Lingas:** Heuristics for Minimum Edge Length Rectangular Partitions of Rectangular Partitions of Rectilinear Figures, *Proc of 6th GI Conference on Theoretical Computer Science*, Dortmund (1983), *Lectures Notes in Computer Science*, Springer Verlag.
79. **Andrzej Lingas:** An Application of Maximum Bipartite C-Matching to Subtree Isomorphism, *Proc. of the 8th Colloquium on Trees in Algebra and Programming*, L'Aquila (1983). *Lectures Notes in Computer Science,* vol 159, Springer Verlag.
80. **Andrzej Lingas:** A Note on Complexity of Logic Programs, *Proc. of the Logic Programming Workshop*, Aldeia das Acoteias, Portugal (1983).
81. **Andrzej Lingas:** The Greedy and Delauney Triangulations are not bad in the average case and Minimum Weight Geometric Triangulation of Multi-Connected Polygons is NP-complete, *Proc. of the International Conference on Foundations of Computation Theory*, Borgholm (1983), *Lecture Notes in Computer Science,* vol 158, Springer Verlag. See also *Information Processing Letters,* vol 22, pp 25-31, (1986).
82. **Andrzej Lingas:** A Linear-Time Heuristic for Minimum Weight Triangulation of Convex Polygons. *Proc. of the Allerton Conference on Communication, Control, and Computing*, Urbana, Illinois 1985.
83. **Andrzej Lingas:** Subgraph Isomorphism for Easily Separable Graphs of Bounded Valence. *Proc. of the 11th Int. Workshop on Graphtheoretic Concepts in Computer Science*, Castle Schwanberg, Wuerzburg, Germany, June, 1985.
84. **Andrzej Lingas:** On Partitioning Polygons. *Proc of the 1st ACM Symposium on Computational Geometry*, Baltimore, Maryland, June 1985.
85. **Andrzej Lingas,** Subgraph Isomorphism for Biconnected Outerplanar Graphs in Cubic Time, in the *Proc. of 3rd Symposium on Theoretical Aspects of Computer Science,* January, 1986, Orsay, France, *Lecture Notes in Computer Science,* vol 210, Springer Verlag.
86. **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.
87. **J. Bryan Lyles:** CAD Approaches for an Asynchronous Architecture. In *Proc. of the Nordic Symposium on VLSI in Computers and Communications*, June 13-15, 1984, Tampere, Finland.
88. **J. Bryan Lyles, Zebo Peng, Johan Fagerström:** Naming Services in a Distributed Computer Architecture. In *Proc. of the Nordic Symposium on VLSI in Computers and Communications,* June 13-15 1984, Tampere, Finland.
89. **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.
90. **Jan Maluszynski, Jorgen Fischer Nilsson:** A Comparison of the Logic Programming Language Prolog with Two-Level Grammars. *Proc. of the 1st Logic Programming Conference*, Marseille-Luminy, 1982.

91. **Jan Maluszynski, Jorgen Fischer Nilsson:** A version of Prolog based on the notion of two-level grammar. *Proc. of the Prolog Programming Environments Workshop*, Linköping, 1982.
92. **Jan Maluszynski, Jorgen Fischer Nilsson:** Grammatical Unification. *Information Processing Letters,* vol 15 pp 150-158, (1982).
93. **Jan Maluszynski:** Towards a Programming Language based on the Notion of Two-Level Grammar. *Theoretical Computer Science,* vol 28, pp 13-43, North-Holland (1984).
94. **Jan Maluszynski, H. Jan Komorowski:** Unification-Free Execution of Logic Programs, in *Proc of the IEEE Symposium on Logic Programming,* Boston 1985.
95. **Minton, Carbonell, Knoblock, Kuokka and Henrik Nordin,** Improving the Effectiveness of Explanation-based Learning, in *Proc. of the Workshop on Knowledge Compilation*, Sept. 24-26, Oregon State University, 1986.
96. **Ulf Nilsson:** AID: An Alternative Implementation of DCGs. *New Generation Computing, vol 4*, No 4 pp 383-399, 1986.
97. **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.
98. **Ludmila Ohlsson:** A Computer Model for Domain Dependent Systems. *Proc of 7th Int. ALLC Symp. on Computers in Literary and Linguistic Research*, Pisa, 1982 (North-Holland).
99. **Lin Padgham:** LINCKS Linköpings Intelligent Knowledge Communication System. (Revised Version). In *Proc of I.F.I.P. Conference on Methods and Tools for Office Systems*, Pisa, Italy, October 22-24, 1986.
100. **Lin Padgham, Ralph Rönnquist:** An Imperative Object Oriented System. *Proc. of the 20th Hawaii International Conference on System Sciences*, 1987, vol 1, p 516.
101. **Lin Padgham, Ralph Rönnquist:** From a Technical to a Humane Environment: A Software System Supporting Co-operative Work. *Proc. of the GDI International Conference on USER INTERFACES*, Rüschlikon, Switzerland, 20-21 Oct, 1986.
102. **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.
103. **Mikael Patel:** A Threaded Interpretive Language Supporting Programming in the Large. *Proc. of the 6th Rochester Forth Conference*, June 11-14, 1986, Univ of Rochester, Rochester, N Y.
104. **Zebo Peng:** A Formal Approach to the Synthesis of VLSI Systems from their Behavioral Descriptions,*Proc 19th Annual Hawaii Int. Symp. on System Sciences, Hawaii,* Jan 1986, pp 160-167.
105. **Zebo Peng:** Synthesis of VLSI Systems With The CAMAD Design Aid. In *Proc. of the 23rd ACM/IEEE Design Automation Conference*, Las Vegas, June 1986.
106. **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.
107. **Zebo Peng and K Kuchcinski:** Synthesis of Control Structures From Petri Net Descriptions, *Microprocessing and Microprogramming,* Vol.18, Nrs 1-5, 1986, pp. 335-340.
108. **Zebo Peng:** Construction of Asynchronous Concurrent Systems From Their Behavioral Specifications, *Proc. 10th World Computer Congress IFIP-86,* Dublin, Ireland, Sept. 1986, pp.859-864.
109. **Ivan Rankin:** SMORF - An Implementation of Hellberg's Morphology System. In Papers from the *Fifth Scandinavian Conference of Computational Linguistics,* University of Helsinki, Dept of General Linguistics, pp 161-172.
110. **Günter Riedewald, Jan Maluszynski, Piotr Dembinski:** *Formale Beschreibung von Programmiersprachen*, R. Oldenburg Verlag, Munchen, Wien, (1983).
111. **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.
112. **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 (to appear.)

113. **Michael Reinfrank:** Reason Maintenance Systems. In Herbert Stoyan (ed.) *Proc. of the Workshop on Truth Maintenance*, Berlin, 1986. Springer Verlag (to appear.)
114. **Olle Rosin, Hans Holmgren, Sture Hägglund,** Implementing Tuning and Feedback Facilities in a System for Patient Management Simulations, *Proc. 3rd Congress on Medical Informatics Europe*, Toulouse, 1981.
115. **Piotr Rudnicki, Wlodzimierz Drabent:** Proving Properties of Pascal Programs in MIZAR 2, *Acta Informatica,* vol 22, pp 311-331, 1985.
116. **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.
117. **Kristian Sandahl:** The Migration of Expert Systems into Production Environments. *Proc. Nord-Info Seminar on Knowledge Engineering*, Köpenhamn, 1986.
118. **Erik Sandewall et al:** Provisions for Flexibility in the Linköping Office Information System, *Proc. of the National Comp. Conf.*, Los Angeles, 1980.
119. **Erik Sandewall, Claes Strömberg, Henrik Sörensen:** Software Architecture Based on Communicating Residential Environments. *Proc. of the 5th Int. Conf. on Software Engineering*, San Diego, 1981. Also in Barstow et al. (eds.) *Interactive Programming Environments*, McGraw-Hill, 1984.
120. **Erik Sandewall, Henrik Sörensen, Claes Strömberg:** A System of Communicating Residential Environments. *Proc. of the 1980 LISP Conf.*, Stanford, Calif, 1980
121. **Erik Sandewall:** Unified Dialogue Management in the Carousel System. *Proc. of the ACM Conference on Principles of Programming Languages*, Albuquerque, NM, 1982. Appeared in print in *N. Naffah (ed.) Office Information Systems,* North Holland, 1982.
122. **Erik Sandewall:** An Environment for Development and Use of Executable Application Models. Presented at the *seminar "Software factory experiences",* Capri, May 3-7, 1982.
123. **Erik Sandewall, Sture Hägglund, Christian Gustafsson, Lennart Jonesjö, Ola Strömfors:** Stepwise Structuring - A Style of Life for Flexible Software. *Proc. of the National Computer Conference,* Anaheim, 1983.
124. **Erik Sandewall:** Formal Specification and Implementation of Operations in Information Management Systems. In: Jan Heering and Paul Klint (eds.), *Colloquium Programmeeromgevingen*, MC Syllabus, Mathematisch Centrum, Amsterdam 1983.
125. **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.
126. **Erik Sandewall, Ralph Rönnquist:** A Representation of Action Structures. In *Proc. of the 5th National Conf. on Artificial Intelligence, AAAI-86, Philadelphia, 1986.*
127. **Erik Sandewall:** Non Monotonic Inference Rules for Inheritance with Exception. In *Proc. of the IEEE, Special Issue on Knowledge Representation.* 1986.
128. **Erik Sandewall:** Specification Environments for Information Management Systems. Panel position paper in *Proc. IFIP Congress 1986.*
129. **Piotr Siemienski:** A specialized VLSI CAD DATABASE. *Nordic Symposium on VLSI in Computers and Communications*, June 13-15 1984, Tampere, Finland.
130. **Dan Strömberg, Peter Fritzon:** Transfer of Programs from Development to Runtime Environments. *BIT,* vol 20, no 4, 1980.
131. **Ola Strömfors, Lennart Jonesjö:** The Implementation and Experiences of a Structure-Oriented Text Editor. *Proc of the ACM SIGPLAN/SIGOA Symposium on Text Manipulation*, Portland, Oregon, June 8-10, (SIGPLAN NOTICES, vol 16, no 6) 1981.
132. **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.

133. **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.
134. **Bo Sundgren:** How to Satisfy a Statistical Agency's Need for General Survey Processing Programs. *Proc. of the 45th Session of the International Statistical Institute*, Amsterdam, Aug 12-22, 1985.
135. **Erik Tengvald**, Reducing Design Complexity, or Why does AI Work, *Proc. of the AIMSA-84 Conf.*, Varna, Bulgaria, (1984).
136. **Ove Wigertz, Johan Elfström, Sture Hägglund and Olle Rosin:** Computer-Assisted Training in Patient Management and Clinical Decision Making. in Pages et al. (eds.) *Meeting the Challenge: Informatics and Medical Education*, North-Holland, 1983.
137. **Jerker Wilander:** An interactive programming system for Pascal. *BIT*, vol 20, 2, 1980. Also in Barstow et al. (eds.) *Interactive Programming Environments*, McGraw-Hill, 1984.
138. **Yoshikazu Yamamoto, Mats Lenngren:** Graphic Model Building System, in *Proc of the 16th Annual Simulation Symposium*, Tamppa, Fla., 1983, and in *Proc of the IMACS Symp. on Simulation in Engineering Sciences*, North-Holland, Nantes, 1983.

DEPARTMENTAL REPORTS 1986-

(Reports in the series LiU-LIBLAB-R-86- are listed in chapter 8)

LiTH-IDA-R-86-01 **Christos Levcopoulos:** Minimum Length and "Thickest-First" Rectangular Partitions of Polygons. Also in *Proc.of the 23rd Annual Allerton Conference on Communication, Control and Computing*, Monticello, Illinois, October 1985.

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

LiTH-IDA-R-86-03 **Christos Levcopoulos:** A fast Heuristic for Covering Polygons by Rectangles. Also in *Proc. of Int. Conf. on Fundamentals of Computation Theory (FCT'85)*, Cottbus, GDR, September 1985 and *Lecture Notes* in Computer Science Nr 199.

LiTH-IDA-R-86-04 **Arne Jönsson Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms. Also in *Proc. of the 19th Annual Hawaii International Conference on System Science*, Jan. 8-10, 1986.

LiTH-IDA-R-86-06 **Harold W. Lawson, Jr:** The DATASAAB Flexible Central Processing Unit.

LiTH-IDA-R-86-07 **Lars Ahrenberg:** Lexikalisk-Funktionell Grammatik på svenska. Finns i proc från *Föredrag vid de nordiska datalingvistikdagarna 1985*, Helsingfors universitet, inst för lingvistik.

LiTH-IDA-R-86-08 **Christos Levcopoulos.** Fast Heuristic for Minimum Length Rectangular Partitions of Polygons. Also in *Proc of the Second ACM Symposium on Computational Geometry*, June 2-4, 1986, Yorktown Heights, New York.

LiTH-IDA-R-86-09 **Harold W. Lawson:** An Asynchronous Approach to Microprogramming.

LiTH-IDA-R-86-10 **Andrzej Lingas:** Subgraph Isomorphism for Biconnected Outerplanar Graphs in Cubic Time. Also in *Proc. of the 3rd Symposium on Theoretical Aspects of Computer Science*, January 1986, Orsay, France and *Lecture Notes* in Computer Science, Springer Verlag.

LiTH-IDA-R-86-11 **Andrzej Lingas:** The Greedy Trianagulation Heuristic for Minimum Weight Triangulation of Convex Polygons Approxiamtes the Optimum. Also in *Proc of the 2nd ACM Symposium on Computational*

Geometry, Yorktown Heights, New York, June 1986.

LiTH-IDA-R-86-12 **Dimiter Driankov:** Inference with Consistent Probabilities in Expert Systems.

LiTH-IDA-R-86-14 **Dimiter Driankov:** Uncertainty Calcus with Verbally Defined Belief-Intervals. Also in *International Journal of Intelligent Systems.*

LiTH-IDA-R-86-15 **Dimiter Driankov:** Inference with a Single Fuzzy Conditional Proposition. Also in *International Journal for Fuzzy Sets and Systems.*

LiTH-IDA-R-86-16 **Dimiter Driankov:** An Outline of Fuzzy Sets Approach to Decision Making with Independent Goals. Also in *International Journal for Fuzzy Sets and Systems.*

LiTH-IDA-R-86-17 **Dimiter Driankov:** A Calculus for Belief-Intervals Representation of Uncertainty. Also in *International Conference on Information Processing and Management of Uncertainty in Expert Systems*, Paris 30 June - 4 July, 1986.

LiTH-IDA-R-86-18 **Lin Padgham:** LINCKS Linköpings Intelligent Knowledge Communication System. (Revised Version). In *Proc of I.F.I.P. Conference on Methods and Tools for Office Systems*, Pisa, Italy, October 22-24, 1986. Also presented at *Interaktiva Administrativa System Konferens*, Åre, 14-16 April 1986.

LiTH-IDA-R-86-19 **Björn Fjellborg:** A Simulation Study of Four Binary Tree Structures.

LiTH-IDA-R-86-20 **Henryk Jan Komorowski, Jan Maluszynski:** Logic Programming and Rapid Prototyping. Also published as report TR-01-86, Harward University, Aiken Computation Laboratory.

LiTH-IDA-R-86-21 **Johan Fagerström, Yngve Larsson, Lars Strömberg:** Distributed Debugging - Collected Ideas.

LiTH-IDA-R-86-22 **Johan Fagerström:** Tradeoffs in an Architecture based on Asynchronous Processes. Accepted for *2nd Nordic Symposium on VLSI in Computers and Communications*, 1986.

LiTH-IDA-R-86-23 **Wlodzimierz Drabent, Jan Maluszynski:** Proving Run-Time Properties of Logic Programs.

LiTH-IDA-R-86-24 **Johan Fagerström, Mikael R.K. Patel:** High-level Simulation of Systolic Architecture. Also in *Proc of the International Workshop on Systolic Arrays,* Oxford, 2-4 July, 1986.

LiTH-IDA-R-86-25 **Rolf G. Karlsson:** Greedy Matching on a Grid.

LiTH-IDA-R-86-26 **Rolf G. Karlsson:** Point Location in Discrete Computational Geometry. A preliminary version appeared in *"Proc. 6th Brazilian Congress on Computing"*, July 1986.

LiTH-IDA-R-86-27 **Rolf G. Karlsson, J. Ian Munro:** Proximity on a Grid under L_1 and L_∞ Metrics. A preliminary version appeared in *"2nd Symposium on Theoretical Aspects of Computer Science"*, 1985, Springer-Verlag, LNCS, 182, 187-196.

LiTH-IDA-R-86-28 **Zebo Peng:** Synthesis of VLSI Systems with the CAMAD Design Aid. Also published as part of *the Proc of the 23rd ACM/IEEE Design Automation Conference*, Las Vegas, June 29 - July 2, 1986.

LiTH-IDA-R-86-29 **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.

LiTH-IDA-R-86-30 **Rolf G. Karlsson, Mark H. Overmars:** Scanline Algorithms on a Grid.

LiTH-IDA-R-86-31 **Magnus Merkel:** A Swedish Grammar in D-PATR. Experiences of working with D-PATR.

LiTH-IDA-R-86-32 **Bengt Lennartsson:** Programming Environments and Paradigms - Some Reflections. Appeared at the *Workshop on Programming Environments - Programming Paradigms"*, Roskilde, Denmark, October 1986.

LiTH-IDA-R-86-33 **Ivan Rankin;** SMORF User's Guide.

LiTH-IDA-R-86-34 **Ivan Rankin;** SMORF - an Implementation of Hellberg's Morphology

System.

LiTH-IDA-R-86-35 **Johan Fagerström, Yngve Larsson and Lars Strömberg:** Debugging Techniques for Distributed Environments. Accepted for the *Workshop on Compiler and Incremental Compilation* in Bautzen, East Germany, October 11-18, 1986 and the *Workshop on Programming Environments - Programming Paradigms*, Roskilde University Centre, Denmark, October 22-24, 1986.

LiTH-IDA-R-86-36 **Peter Fritzson:** Systems and Tools for Exploratory Programming. Overview and Examples.

LiTH-IDA-R-86-37 **Rolf G Karlsson, Mark H Overmars:** Normalized Divide and Conquer: A Scaling Technique for Solving Multi-Dimensional Problems.

LiTH-IDA-R-86-38 **Peter Fritzson:** A Common Intermediate Representation for C, Pascal, Modula-2 and Fortran-77. Also presented at the *Workshop on Compiler Compilers and Incremental Compilation*, Bautzen, DDR, October 12-17, 1986.

LiTH-IDA-R-86-39 **Ola Strömfors:** A Structure Editor as a Template for Programming Environment Functions. This paper was presented at a *Workshop on Programming Environments - Programming Paradigms*, at Roskilde University Centre, Denmark, October 22-24, 1986.

LiTH-IDA-R-86-40 **Mariam Kamkar, Nahid Shahmehri:** Runtime Dependent Program Flow Analysis. This is a revised version of paper presented at a *Workshop on Programming Environments - Programming Paradigms*, at Roskilde University Centre, Denmark, October 22-24, 1986.

LiTH-IDA-R-86-41 **James A. Dean, Andrzej Lingas, Jörg-R. Sack:** On Recognizing Polygons, or how to Eavesdrop. Also in *Proc of the Allerton Conference on Communication, Control, and Computing*, Urbana, Illinois, 1986.

LiTH-IDA-R-86-42 **Nils Dahlbäck, Arne Jönsson:** A Method for Studying Human-Computer Dialogues in Natural Language.

LiTH-IDA-R-86-43 **Ola Strömfors:** Editing Large Programs Using a Structure-Oriented Text Editor. Also presented at *the International Workshop on Advanced Programming Environments*, Trondheim, Norway,June 16-18, 1986.

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. Also presented at the *Workshop on Programming Environments - Programming Paradigms*, Roskilde, Denmark, October 22-24, 1986.

LiTH-IDA-R-87-03 **Mikael R.K. Patel:** A Threaded Interpretive Language Supporting Programming in the Large. Also in *Proc. of The sixth Rochester Forth Conference*, University of Rochester, Rochester, New York, June 11-14, 1986.

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.

Diskussion & Debatt:

LiTH-IDA-R-87-04 **Arja Vainio-Larsson:** Datavetenskap: Teknik och Vetenskap.

FURTHER INFORMATION

LABORATORY LEADERS. Secretaries names are given in italics.

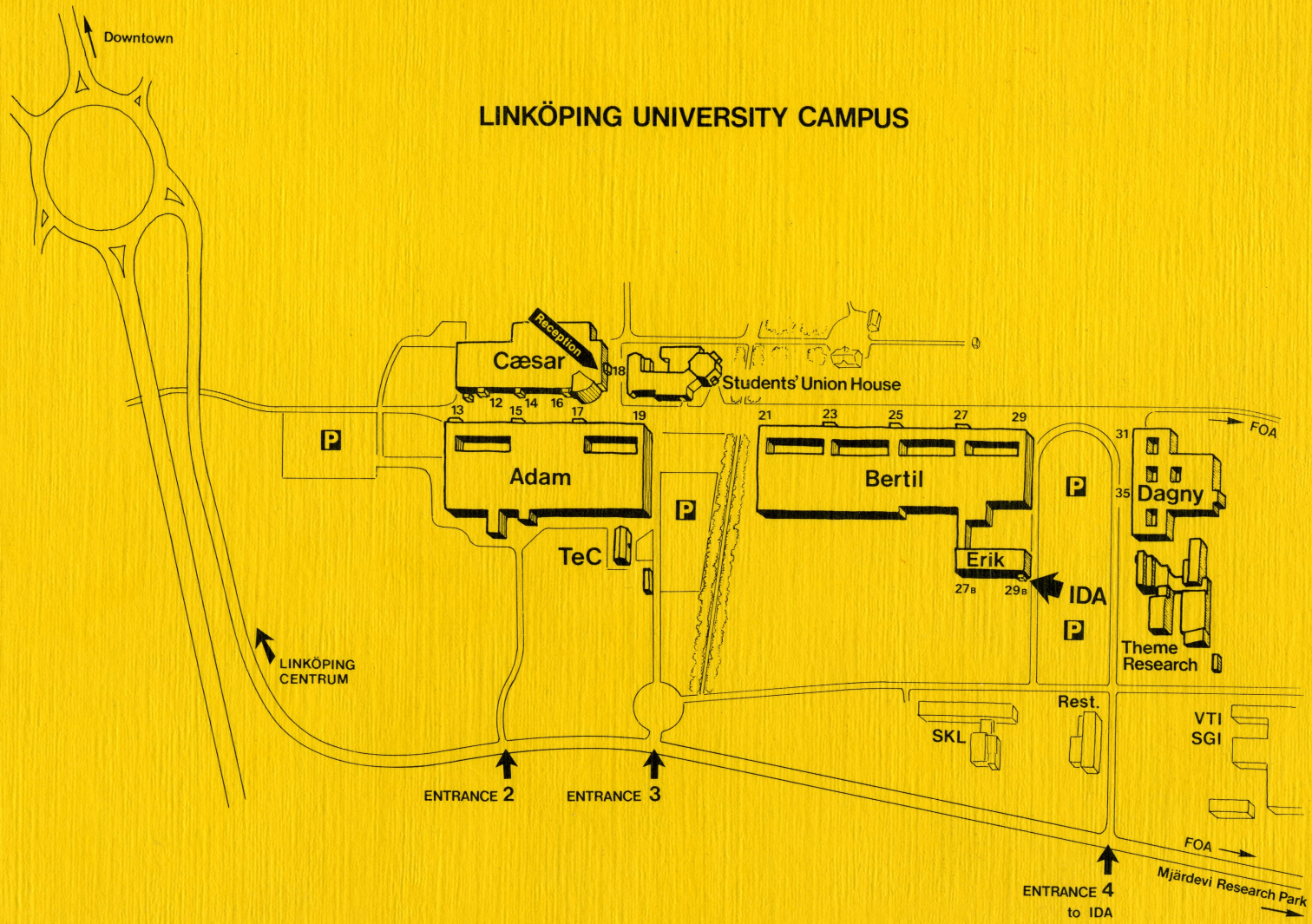
Unit	*Leaders*	
Graduate Studies Programme, general	Erik Sandewall	281408
	Lillemor Wallgren	281480
Undergradutate Studies Programme, general	Anders Harldsson	281403
	Barbara Ekman	281410
	Lena Wigh	282492
Laboratories:		
Administrative Data Processing Group	Göran Goldkuhl	281452
	Carina Björkman	281458
AI Environments	Erik Tengvald	281470
	Lisbeth Linge	281472
Application Systems	Sture Hägglund	281431
	Gunilla Lingenhult	282297
CAD of Digital System	Harold Lawson	281314
	Britt-Marie Ahlenbäck	281318
Complexity of Algorithms	Andrzej Lingas	281938
	Bodil Mattson Kihlström	281652
Knowledge Representation in Logic	Erik Sandewall	281408
	Anne-Marie Jacobson	281975
	Lillemor Wallgren	281480
Library and Information Science	Roland Hjerppe	281965
	Siv Söderlund	281426
Logic Programming	Jan Maluszynski	281483
	Bodil Mattsson Kihlström	281652
Natural Language Processing	Lars Ahrenberg	282427
	Britt-Marie Ahlenbäck	281318
Programming Environments	Bengt Lennartsson	281427
	Gunilla Lingenhult	282297
Admittance of foreign graduate students	Jan Maluszynski	281483
Administration	Lillemor Wallgren	281480

MAILING ADDRESS

Department of Computer and Information Science
Linköping University
S-581 83 Linköping
SWEDEN

TELEPHONE 013 - 281000 or directly to the numbers above
TELEFAX int+46 13 14 22 31
TELEX 8155076 LIUIDA S

LINKÖPING UNIVERSITY CAMPUS



Klintland Grafiska, Linköping 1987