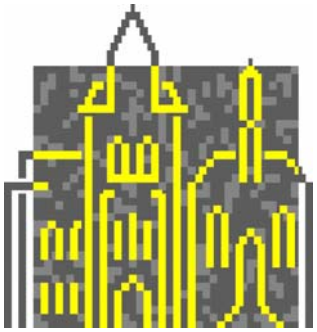

SCHLOSS DAGSTUHL

INTERNATIONAL
CONFERENCE
AND RESEARCH CENTER
FOR COMPUTER SCIENCE

Dagstuhl News

January - December 1999

Volume 2
2000

ISSN 1438-7581

Period: January - December 1999

Frequency: 1 per year

The International Conference and Research Center for Computer Science is operated by a non-profit organization. Its objective is to promote world-class research in computer science and to host research seminars which enable new ideas to be showcased, problems to be discussed and the course to be set for future development in this field.

Associates: Gesellschaft für Informatik e.V., Bonn
Technische Universität Darmstadt
Universität Frankfurt
Universität Kaiserslautern
Universität Karlsruhe
Universität Stuttgart
Universität Trier
Universität des Saarlandes

Funding: The state governments of Saarland and Rhineland Palatinate

Address: IBFI Schloß Dagstuhl
Octavieallee
D-66687 Wadern
Tel.: +49 - 6871 - 905127
Fax: +49 - 6871 - 905130
E-mail: service@dagstuhl.de
Internet: http://www.dagstuhl.de/

Welcome

Rather late in the year, you receive the second edition of the "Dagstuhl News", a publication for the members of the Foundation "Informatikzentrum Schloss Dagstuhl", the *Dagstuhl Foundation* for short. The delay was caused by the efforts taken by the preprocessing, the running, and the postprocessing of our 10th Anniversary Conference, "Informatics – 10 Years Back, 10 Years Ahead". The conference offered very interesting talks by leading scientists of our discipline. They leaned back, looked what has happened in their respective areas during the existence of Dagstuhl and tried to speculate what will happen in the near future. The proceedings of the conference will soon be published as LNCS volume 2000 by Springer Verlag.

The main part of this leaflet consists of collected resumees and other valuable information taken from the Dagstuhl-Seminar Reports. We hope that you will find this information valuable for your own work or informative as to what colleagues in other research areas of Computer Science are doing. The full reports for 1999 are on the Web under URL www.dagstuhl.de/DATA/Seminars/99.

The State and the Activities of the *Dagstuhl Foundation*

The foundation currently has 44 personal members and 9 institutional members.

In 1999, the foundation has supported a few guests with travel grants and a reduction of the Seminar fees. According to German law only the interests earned can be used to support the aims of a foundation.

Thanks

I would like to thank you for supporting Dagstuhl through your membership in the *Dagstuhl Foundation*. Thanks go to Fritz Müller for editing the resumees collected in this volume.

Reinhard Wilhelm (Scientific Director)

Contents

1 Decision Diagrams – Concepts and Applications 7

2 Software Engineering Research and Education: Seeking a new Agenda 9

3 Component-Based Programming under Different Paradigms 13

4 Deduction 17

5 Computational Geometry 18

6 Systems Integration 19

7 Unsupervised Learning 24

8 Program Analysis 26

9 Instruction-Level Parallelism and Parallelizing Compilation 35

10 High Level Parallel Programming: Applicability, Analysis and Performance 41

11 Mobile Multimedia Communication – Systems and Networks 45

12 Geometric Modelling 45

13 Graph Decompositions and Algorithmic Applications 48

14 Requirements Capture, Documentation, and Validation 49

15 Competitive Algorithms 50

16 Foundations for Information Integration 50

17 Agent Oriented Approaches in Distributed Modeling and Simulation: Challenges and Methodologies 51

18 Parallel and Distributed Algorithms 55

19 Computer Science in Astronomy 55

20 Linear Logic and Applications 56

21 Multimedia Database Support for Digital Libraries 59

22 Social Thinking – Software Practice 61

23 Declarative Data Access on the Web 64

24 Computational Cartography – Cartography meets Computational Geometry 67

25 Finite Model Theory, Databases, and Computer Aided Verification 69

26 Temporal Logics for Distributed Systems – Paradigms and Algorithms 71

27 Efficient Language Processing with High-level Grammar Formalisms 74

28 Scheduling in Computer and Manufacturing Systems 76

29 Complexity of Boolean Functions 77

30 Rigorous Analysis and Design for Software Intensive Systems 78

31 Computability and Complexity in Analysis 80

32 Symbolic-Algebraic Methods and Verification Methods – Theory and Applications 81

33 Content-Based Image and Video Retrieval 81

1 Decision Diagrams – Concepts and Applications

Seminar No. **99041** Report No. **229** Date **24.01.–29.01.1999**
Organizers: Bernd Becker, Christoph Meinel, Shin-Ichi Minato, Fabio Somenzi

The fifth workshop **Decision Diagrams – Concepts and Applications** in the series *Computer Aided Design and Test* at the IBFI Schloß Dagstuhl was organized by Bernd Becker (Univ. Freiburg), Christoph Meinel (Univ. Trier), Shin-Ichi Minato (NTT Optical Network, Japan), and Fabio Somenzi (Univ. of Colorado). It was attended by 31 scientists.

Decision Diagrams (DDs) have found widespread use in computer-aided design of digital circuits. They form the heart of many tools for formal verification and are also commonly used in logic synthesis, circuit testing and in the verification of communication protocols. With increasing number of applications, also in non–CAD areas, classical methods to handle DDs are being improved and new questions and problems evolve and have to be solved.

The organizers took the opportunity to bring together researchers from different areas in computer science, electrical engineering and industry. The common aim of all researchers is to deepen the understanding of DDs as a data structure, to improve existing techniques and to explore new fields of application. At the workshop, 23 lectures were presented covering different topics of DD research among them being:

- Potential and limitations of DDs, complexity of algorithms for (Boolean) function manipulation
- Minimization and approximation of Binary DDs (BDDs)
- Formal verification of sequential circuits with BDD based methods
- Extensions beyond Boolean functions to represent and manipulate word-level circuit functions
- Applications in synthesis, design and test of real-time systems, state/event systems

There were many discussions concerning challenging open questions — at universities and in industry as well — and future directions of research in the DD area.

The detailed program including the abstracts and some full papers can be found on the WWW-page:

`http://ira.informatik.uni-freiburg.de/events/design_and_test_99/program.html`

An interesting abstract selected by the Dagstuhl News editor:

Nonapproximability Results for OBDD- and FBDD-Minimization

Detlef Sieling, Universität Dortmund

The variable ordering problem for OBDDs is the problem to compute a variable ordering minimizing the OBDD size of a function given by an OBDD. Its complexity is of theoretical and practical importance because the choice of the variable ordering can decide between polynomial or exponential OBDD size and between success or failure of an application. The known NP-hardness results do not exclude polynomial time approximation algorithms for the variable ordering problem, i.e. algorithms which guarantee to obtain variable orderings for which the OBDD size is larger than the optimum by at most some constant factor. The main result is that the existence of such an algorithm implies P=NP. Hence, we get a justification to use heuristics and to give up the search for approximation algorithms for the variable ordering problem.

Besides OBDDs, Free BDDs (FBDDs) can be used as a data structure for Boolean functions. The manipulation of FBDDs is efficient if only FBDDs respecting a fixed graph ordering are used. Graph orderings are a generalization of variable orderings. Hence, we get the problem of minimizing FBDDs and of optimizing graph orderings for a function given by an FBDD. The other main result is that polynomial time approximation schemes for these problems imply P=NP or ZPP=NP, respectively. Approximation schemes are algorithms whose result is larger than the optimum by a factor of at most $1 + \epsilon$ where $\epsilon > 0$ is part of the input. Again, we get a justification to use heuristics.

2 Software Engineering Research and Education: Seeking a new Agenda

Seminar No. **99071** Report No. **230** Date **14.02.–19.02.1999**
Organizers: Ernst Denert, Daniel Hoffmann, Jochen Ludewig, David Parnas

Taking Stock of Software Engineering Research and Education What do we know? What should we know?

Introduction

Software Engineering should address, and solve, existing problems.

Software Engineering as a branch of computer science emerged from discussions and conferences in the late sixties. Its goal was to apply knowledge and techniques from traditional engineering to the construction and maintenance of software.

Now, as the end of the century draws near, many people apply concepts, techniques, and notations created in, and around, the field of software engineering. But we are far away from a common understanding of what the important problems are and which approaches are appropriate. We have many conferences and magazines of questionable value, and little agreement about what should be taught in universities, and which topics should be further investigated.

These ideas in mind, four people, namely Dan Hoffman and David L. Parnas in Canada, Ernst Denert and Jochen Ludewig in Germany, organized a Dagstuhl workshop. Stefan Krauß served as a secretary before, during, and after the event.

This workshop attempted to reach an agreement about these questions, for those who participated, but hopefully also with some effect on our colleagues who did not. By discussing our ability to solve those problems which actually occur in software engineering, we identified what should be in the curriculum and in the research agenda.

As part of the announcement and invitation, a list of eleven suggested tasks was distributed, ranging from "Analyse intended application, write requirements document." to "Revise and enhance software systems." Those who intended to participate were asked to submit a position paper

on at least one of the subjects, and to judge the current state of all the subjects.

At the beginning of the workshop, participants reduced and modified the topics for various reasons, until nine topics for discussion and elaboration were selected. The topics and the most fundamental questions related to these topics are listed below; no attempt is made in this summary to discuss the results in detail; full information is provided in the report.

For each topic, one of the participants was selected as the "secretary" (see below). Most attendees participated in two of the subgroups. Plenary meetings were held once or twice the day, allowing for critical discussions and feedback.

All results were (almost) uniformly cast into tables, using criteria and attributes that had been agreed in the plenary meetings. These tables contain judgements like this one (about the state of the art in statistical testing):

Effectiveness:	low
Affordability:	low
Teachability:	medium
Penetration:	none
Research potential:	low

These schematic judgements were often complemented by remarks.

The Topics

1. Requirements (Joanne Atlee)

How can we analyze the intended application to determine the requirements that must be satisfied? How should we record those requirements in a precise, well-organized and easily-used document?

In practice, this goal is rarely achieved. In most projects, a significant number of software development errors can be traced back to incomplete or misunderstood requirements.

We need to improve the state of requirements engineering by improving our application of existing practices and techniques, evaluating the strengths and weaknesses of the existing practices and techniques, and

developing new practices and techniques where the existing ones do not suffice.

2. Software Design (Johannes Siedersleben)

How can we design the basic structures of the software, evaluate competing design alternatives, and reuse existing designs whenever possible?

Participants felt that the design task is solved at the level of individual modules, but not solved at the level of system architecture. Designing system architectures, the known techniques are not sufficient. Another important question is how to relate different levels of abstraction in a large system.

We need a collection of architectural patterns, i.e. proven architectural building blocks aimed at particular problems, possibly domain-specific.

3. Implementation (Peter Knoke)

The problem is that the quality of the software implementation (i.e. the code) is generally not as good as it could be, and should be. How can this situation be improved by various means, including SE teaching, SE research, or other means?

The participants agreed in the importance of this field which is often neglected in higher education. Good programming is not that easy; it should be taught, and it should be generally recognized.

There is little need for research in this area, but education should be greatly improved, emphasizing good craftsmanship.

4. COTS (Commercial-Off-The-Shelf) (Paul Strooper)

The use of COTS is where a portion of an application is not custom-developed , but instead provided by a COTS product. This is the problem we addressed here: Do we know how we can integrate COTS products into software systems?

There are significant problems with the integration of COTS software and there is a lack of documented solutions and experience reports dealing with these problems. It was therefore concluded that the problem of integrating COTS products into software products is mainly unsolved, except for small-scale COTS products, such as GUI and component libraries.

5. Test (Dan Hoffman)

How can we perform systematic or statistical testing of the software and the integrated computer system?

Many companies do no systematic testing. Some companies do perform systematic testing; it is often reasonably effective but very expensive in effort and calendar time. Unit testing is usually ad hoc, if performed at all. Test automation is primitive. Test education, especially in universities, is poor.

Given the importance of testing to industry, there is relatively little research done.

6. Families (David Weiss)

This section is concerned with identifying the tasks involved in identifying (and documenting) defined families.

In general, few techniques exist for defining families, i.e., for performing the analysis needed to identify a defined family. There are a few techniques that have become commercially available in the last 2-3 years, but they are just starting to be tried by the early adopters in industry. As a result, the problem was rated as partially solved.

7. Maintenance (Andreas Zeller)

How can we maintain a product's integrity during its evolution?

This problem is partially solved. There is a number of well-understood solutions that help in software maintenance. However, there is a need and a potential for better solutions, especially in the areas of reverse engineering and reengineering.

8. Measurement (Motei Azuma)

Specifically, when dealing with software using metrics, we can assess metrics with the following basic questions: Are we able to describe and forecast process and product characteristics by metrics? Are we able to control process and product using metrics, possibly continuously?

Some metrics are widely known and sometimes used in practice. Yet there are many metrics to be developed, validated and taught for practical use. Therefore the general problem is only partially solved.

The leading role of some standards organizations was pointed out (ISO/IEC JTC1/SC7, JTC1/SC7/WG6).

9. Software Configuration Management (Walter Tichy)

How can we manage and control the evolution of a product?

This problem is solved for standard situations. Software configuration management is a well-understood discipline that offers a number of solutions for managing software evolution.

Distributed software configuration management still requires more attention. Also, composition of complex systems from versioned components brings problems with configuration consistency.

Conclusions

Several participants have expressed their interest in a permanent activity along the lines initiated at Dagstuhl; we will try to keep the spirit alive by presenting our results in conferences and magazines, hopefully stimulating responses from those who were missing. Software Engineering needs definitely more work like this.

3 Component-Based Programming under Different Paradigms

Seminar No. **99081** Report No. **231** Date **21.02–26.02.1999**
Organizers: Philip Wadler, Karsten Weihe

Motivation

Throughout the last decades, much research has focussed on object-oriented, template-oriented, and functional programming techniques. However, there is not much interaction between these research communities. Although there is a high overlap of fundamental ideas and concepts, ideas are expressed in terms of sharply different language features. Worse, the public discussion in each of these communities seems to be dominated by a "purist" viewpoint, which regards the other paradigms as strongly inferior.

Recently, new threads of research have been initiated that try to find practical combinations of different programming styles in mainstream programming languages. This research is centered around Java and C++. Java has turned out to be too restricted for many applications. Consequently, a number of extensions to Java have been proposed and implemented, to add parametric and functional features. On the other hand, the full power of the generic features of C++ and the possibility to simulate other features from the functional realm have been discovered only recently. Since the C++ standard library - and many other

recent libraries - is designed according to these principles, there is a practical need for further research on combinations of generic and functional techniques with an object-oriented programming style.

The notion of components, or component-based programming, seems to be a useful fundament for this kind of research. The meaning of this word is intuitive: programs are broken down into primitive building blocks, which may be flexibly "plugged together" according to well-defined protocols. In fact, each of the above-mentioned programming paradigms may be viewed as an attempt to realize such a component-based programming style, however, the definition of components and the techniques for combining them varies significantly. Hence, analyzing these differences is crucial for a deeper understanding of the problem.

Workshop Experience

This workshop was an experiment. Researchers from different software–engineering and programming language camps were brought together under the leitmotif of components. In the motivation text we defined components as "primitive building blocks, which may be flexibly 'plugged together' according to well-defined protocols," and we called this definition "intuitive." It was clear beforehand that the "big camps"—functional, object–oriented, generative, etc.,—would probably agree on this definition but interpret it quite differently. An early outcome of the workshop was the experience how incompatible these interpretations actually are, and that an agreement is not even reachable inside each camp separately.

The scientific and practical backgrounds of the participants were very different, and there was no evidence that people would easily understand each other and have fruitful discussions. Fortunately, the ice was broken right at the beginning of the workshop, within a quarter of an hour. The opening talk was not regularly finished but resulted almost immediately in a lively, highly controversary discussion (loosely moderated by the speaker), in which the majority of all particants were actively involved. This had set the tone for the rest of the workshop, and so it became one of the most lively workshops we have ever seen.

The workshop was a real success, but the success was not of the kind we expected and aimed at. The motivation text specified three concrete goals, however, our experience is that concrete results are still too ambitious. The goal to understand each other and to establish fruitful paths of

communication across the camps has turned out to be ambitious enough. These paths were the basis for many minor, unanticipated success stories.

A striking example of these minor success stories was a session that was jointly organized by a functional–programming guy and a C++/template–programming guy. They proved convincingly that C++ can be viewed as a real (though not perfectly purist) functional language. This path of communication is in great contrast to the public discussion across the camps in newsgroups and other media, which is all too often frustrating and sometimes even hostile and offending.

What is a component? Beyond the intuitive, informal definition from our motivation text, there is probably no satisfactory, commonly agreeable answer to this question at all. The workshop brought to our mind that our definition left many crucial details open. It does not even specify whether components should be defined to be types or objects, and in fact, we did not come up with an agreement on this point.

Another point of major disagreement was the question of static type safety. Of course, plug and play relies on the assumption that the "plug" and the "socket" fit perfectly together. Translated into this metaphor, static type safety means that a plug does not fit into a socket unless it makes sense. We learned about very good pros and cons—and again, no chance for an agreement.

We also learned a lot about further properties and characteristics that may or may not be essential for a useful notion of components. In a sense, this established a "common language of discourse across the cultures" as stated in the first concrete goal in the motivation text. This is much more than one would expect in retrospect.

Discussion: C++ as a Functional Language

Moderated by Erik Meijer and Lutz Kettner

If you look closely at many C-APIs such as the Win32 API, you can recognize a lot of concepts such as lazy evaluation, call-backs, and closures from functional languages. Perhaps surprisingly this means that many "low-level" programs can be coded more cleanly in a functional language than in `C` or `C++`.

Besides the obvious fact that `C++` is not a functional programming language, it is surprising to see to what extend `C++` has borrowed concepts from functional programming languages. One of the first examples in

the introduction to the Standard Template Library STL, part of the C++ standard, makes extensive use of function objects. Function objects are first class citizens in STL. Even *currying* and higher-order functions can be expressed and easily used. However, the implementation of them is considerably longer than in functional programming language.

Another surprising fact about C++ is a kind of lazy-evaluation at compile time. A member function of a class will only be compiled if it is actually used. In consequence, there will be no error messages for even syntactically wrong code (besides basic rules such as matching curly braces) in the body of unused member functions.

At the previous day a generic function `flatten` was used as an example for polytypic programming in functional languages. It raised the question whether a similar program could be written using templates in C++. Besides that the meta-information for the self-inspection of user-defined types must be given explicitly, it can be written, see `http://www.inf.ethz.ch/personal/kettner/pieces/flatten.html`

Discussion Session

Moderated by Philip Wadler

The word 'component' is used to denote a wide range of different things, and the tendency to stretch its meaning is perhaps exacerbated in a workshop that contains 'component' in its title. Just as Eskimos need fifty words for ice, perhaps we need many words for components. The following were suggested (though not everyone in the group agreed to all of what follows).

Component (typical example: COM)

- Can be used in object form, without access to source
- Can be used from a variety of programming languages
- Communicate by methods, each method with a signature
- Dynamically linked

Process (typical example: Erlang)

- Runs concurrently with other processes
- Processes communicate by means of a protocol

- Process may be sent messages from processes in other languages or on other machines

Module (typical example: Modula)

- Unit of independent compilation
- Used for namespace control

Functor (typical example: ML)

- A module parameterised by other modules
- Based on sophisticated type theory

Composent (typical examples: Demeter, Aspect-oriented programming)

- A unit of functionality weaved together with other units
- Lines of code adjacent in a composent may be far removed in the program woven from the composent
- Requires access to the source code

The word 'composent' was a new one, coined by Wadler and adopted by Lieberherr and Mezini.

Joe Armstrong argued that processes can be superior to components. A key research issues for processes is to devise ways of specifying protocols, analogous to the use of method signatures in a component. Erlang processes are successful because of a number of features not shared by other concurrent paradigms, such as threads: there may be many processes (typically, about ten per phone call, up to 20,000 running concurrently on one machine); there is no shared memory; messages are structured trees (Erlang data structures, roughly similar to Lisp S-expressions or XML trees); processes can monitor each other for errors.

4 Deduction

Seminar No. **99091** Report No. **232** Date **28.02.–05.03.1999**
Organizers: Ulrich Furbach, Harald Ganzinger, Ryuzo Hasegawa, Deepak Kapur

Logic has become a prominent formal language and formalism for all of computer science. It serves in many applications such as in problem specification, program development, program transformation, program verification, hardware design and verification, consistency checking of databases, theorem proving, expert systems, logic programming, and so on and so forth. Its strength derives from the universality of the language as well as from the fundamental logical operations and relations. Logical manipulations as needed in all these applications are realized by mechanisms developed in the field of deduction which has produced a variety of techniques of great importance in all these applications.

All these research issues have been subject of a "Schwerpunktprogramm Deduktion" funded by the Deutsche Forschungsgemeinschaft.

During the last years successful research in this program has led to the development of high performance deduction systems, and to laying a broad basis for various applications.

This success of deduction can be observed within the international AI and computer science scene as well. Deduction systems recently have achieved considerable successes and even public press: it was a first-order theorem prover which first proved the Robbins algebra conjecture and even reached the New York Times Science section. But not only in proofing mathematical theorems, also in various other disciplines of AI, automated deduction made substantial progress. In planning, for example, it turned out that propositional theorem provers are able to outperform special purpose planning systems. This is remarkable, since it was considered folklore that planning requires specialized algorithms, which was only recently disproved by the development of propositional satisfiability testing methods which can now handle much larger planning problem sizes. A very similar development can be observed in the field of model based diagnosis.

5 Computational Geometry

Seminar No. **99102** Report No. **233** Date **07.03.–12.03.1999**
Organizers: Michael Goodrich, Rolf Klein, Raimund Seidel

Geometric computing is creeping into virtually every corner of science and engineering, from design and manufacturing to astrophysics

and cartography. This report describes presentations made at a workshop focused on recent advances in this computational geometry field.

Previous Dagstuhl workshops on computational geometry dealt mostly with theoretical issues: the development of provably efficient algorithms for various geometric problems on point sets, arrangements of curves and surfaces, triangulations and other sets of objects; proving various combinatorial results on sets of geometric objects, which usually have implications on the performance of performance guarantees of geometric algorithms; describing the intrinsic computational complexity of various geometric problems.

This workshop continued some of this tradition, but as one more point there was a strong emphasis on the exchange of ideas regarding carrying the many theoretical findings of the last years into computational practice. There were presentations about the recent development of software libraries such as CGAL, LEDA, JDSL, VEGA, and TPIE, and also some experimentation with them. These libraries should help to simplify the realization of abstractly conceived geometric algorithms as actually executable software.

6 Systems Integration

Seminar No. **99111** Report No. **234** Date **14.03.–19.03.1999**
Organizers: Paolo Ciancarini, Stefan Conrad, Wilhelm Hasselbring

The integration of systems which have been developed and evolved independently is one of today's major challenges in computer science. In a large spectrum of application areas the necessity of integrating (pre-)existing software systems is present and demands for applicable solutions.

Problems of coupling and integrating heterogeneous database and information systems are for instance investigated in the database area. Building multidatabase systems or federated database systems incorporating legacy systems is a big challenge. Current work covers topics like schema integration, transaction models for federated database systems, consistency enforcement in heterogeneous systems, security models, and query processing.

On the other hand, systems integration is an important challenge for the area of software engineering as well. Current work deals with

questions of adequate software architectures and design patterns, coordination languages and models, composition of software components, development of workflow systems, the proper use of middleware tools such as CORBA, and methodological approaches for the integration process.

The rapid development of Web-related methodologies and tools also stimulates new problems, with respect to the access to Web data, the design and maintenance of Web sites, and their integration with traditional applications.

This Seminar was initiated due to the fact that there was only a rather loose coupling of the work done in these scientific communities, although the work of these areas is obviously highly interrelated. Therefore, we saw the chance that both areas can profit a lot from a mutual exchange of problems and ideas.

The Seminar brought together scientists from these two areas. A main focus of the Seminar was set on integration on system level. However, the influence of other integration levels (e.g., integration on schema or model level) was considered as well. Cross-disciplinary working groups were established during the Seminar aiming at a more detailed investigation of common problems.

Working Group I: Applications & Processes

Myra Spiliopoulou

Working Group 1 addressed issues under the general title "Applications & Processes" in the context of systems' integration. A deep terminological discussion on the notions of process, application etc was decided to be of limited usefulness, while defining a precise notion of the term "integration" is indispensable. Two orthogonal categorizations were agreed upon. The first covers the phases of analysis, design and implementation/runtime. The second distinguishes between high-level interworking, interoperation and low-level interconnection. The level of integration agreed upon was that of *interoperation*.

The notion of integration is goal-oriented. Depending on the goal, communication is a means for integration. At any case, integration may not focus solely at structural issues; semantic problems must also be resolved.

Integration of specifications is the first step to systems' integration. Beyond data, behaviour must be integrated. There is a variety of models for describing them. Mapping those models to a universal canonical model is not always appropriate. In this context, XML is not a solution to the integration problem.

The final issue concerned the cost of integrating a large system, whereby the term "large" is ambiguous. It is recognized that the cost of integration covers expenses for equipment, installation and maintenance, training of the users and time. Ad hoc solutions are less costly than the establishment of a federated DBMS. A reliable budget estimation for a FDBMS is very difficult, for the integration of software it is almost impossible.

Working Group II: XML and Canonical Data Models

Mark Roantree, Karl Aberer

The focus of the workgroup was to discuss whether or not XML could function as a canonical data model for systems integration, or if it could in some way, support an existing canonical model. The workgroup identified a number of different themes under which to discuss the model:

XML's relationship with Java/C++ Java can make use of the Domain Object Model (DOM) to interact with XML. The main point to emerge was that it makes no sense to combine XML with Java, as this defeats the purpose of a language like XML which should be visible as an information transportation medium.

XML's relationship with CORBA This topic sparked quite a lot of debate and the Boeing Project which employs Orbix and XML to integrate a large number of disparate systems provided a good reference point.

For large scale projects such as the Boeing Project it was generally agreed that CORBA could act as the main communicating technology, but it was felt that some high level technologies such as XML also has a role to play (where it made no sense to use CORBA). One of the problems pointed out by the group was the requirement to create Document

Type Definitions (DTDs) for every time of object to be transported. Although it was felt that DTDs could be specified for entire domains, it was agreed that there would be many cases where DTDs would need to be constructed for isolated objects.

The concept of XML replacing IDL was also raised but quickly dropped as being non-viable. CORBA has a specific role for defining behaviour and managing the integration of behaviour, an area where XML was lacking.

The failure of SGML was also raised, and questions were asked as to whether or not XML could avoid the mistakes (complexity) of SGML. Finally, XML was highlighted as a possible replacement for CGI.

Query Languages for XML Details of upto four communities defining different query languages were provided, with XQL and XMLQL being the most prevalent current query language proposals. Microsoft is supporting XQL, XWL update facilities can be expected from the providers.

What emerged from this topic is that there are query languages currently available for XML (a good thing) but no query language has emerged as a standard (bad thing). In fact, different query languages currently offer different types of output as a result of XML queries (documents, elements sets, restructured documents).

XML's role in integrating data and services The focus of this theme was on DTDs. The question of strategy was raised: do we merge two DTDs or can we simply map the same document to different DTDs? The idea of the query language being used to solve the integration issue (possibly through the use of constraints) was also raised.

Can a generic DTD like XMI be used to exchange schemata? In general, it was felt that using XML would make it easier to exchange schemata but that the same issue of semantics still exists: it has just moved to another platform. In fact, it was felt that DTDs were not as rich as database schemas.

Attaching semantics to XML This topic focused on the idea of extending XML to include semantics or some form of rules. The group felt that this was a bad idea, as it wasn't possible to achieve properly. We

were also reminded that SGML's reason for failure was its complexity. If XML is to succeed, it should use its simplicity as a key strength.

Can XML be a CDM, or is it just a model for data transfer? The (almost) unanimous feeling is that XML cannot be a CDM, but should be used where it is strongest: the transfer of data.

Working Group III: Software Architectures & Coordination

Klaus-Peter Löhr

As this was the last of the three workshops, there was a strong feeling among the participants that we should walk away with some general insight into the overall theme of the seminar. So after surveying several topics that had been suggested for discussion, we decided to focus on *patterns of integration.*

The unevitable quest for a precise definition of terms (pattern, integration) was quickly resolved for "pattern" — to mean a recurring structure in space and time that is revealed by abstraction of some kind. Understanding "integration" turned out to be harder. There were widely varying opinions on what constitutes an "integrated system" or a "system resulting from integration", ranging from loosely coupled systems to supersystems where the individual components are no longer recognizable.

An agreement was reached that there are three — largely orthogonal — impediments to integration: *heterogeneity*, *distribution* and *autonomy.* Different degrees of integration will exhibit different degrees of masking heterogeneity, hiding distribution and accommodating autonomy. So a more or less integrated system would be placed somewhere in a three-dimensional space, and the distance from the origin would be a measure of integration.

Typical approaches to coping with heterogeneity, distribution and autonomy were identified to follow four important patterns:

(Heterogeneity:)

1. *Homogenization*: Introduce a new, well-chosen "canonical" entity in addition to the given heterogeneous entities. Define mappings between these and the new entity. Support

those mappings by providing wrappers, mediators, adaptors and similar devices.

2. *Direct accommodation*: Define pairwise mappings between the participating entities. Support those mappings by providing bridges for all pairs where accommmodation is indeed required.

(Distribution:)

3. *Remote invocation via proxies*: This well-known approach is the basis for several middleware systems and can achieve a high degree of invocation transparency.

(We would have loved to come up with a pattern for choosing appropriate middleware in a systematic way. Unfortunately, we didn't; so the issue is left as a topic for further research :-)

(Autonomy:)

(This pattern was not developed in the working group; it was suggested in the following plenary discussion.)

4. *Higher authority*: Existing authorities agree to give up part of their autonomy in order to allow for some coordination exercised by a higher authority.

While each of these observations is not radically new, the members of the working group liked the "integrated view" on systems integration that was achieved.

7 Unsupervised Learning

Seminar No. **99121** Report No. **235** Date **21.03.–26.03.1999**
Organizers: Joachim M. Buhmann, Wolfgang Maass, Helge Ritter, Naftali Tishby

What is unsupervised learning and how does it relate to the well founded theory of supervised learning? These questions have been discussed during this seminar which brought together neural modellers,

statisticians, computational learning theorists ("COLT people") and theoretical computer scientists and physicists. The field of machine learning with its broad range of pattern recognition applications in data mining and knowledge discovery, in information retrieval and in classical areas like speech and image processing, computational linguistics or robotics is confronted with various problems beyond classification and regression. The search for structure in large data sets requires automatic inference tools which can also provide quality guarantees to validate the results.

The discussions after the talks and in special discussion sessions circled around two main issues of unsupervised learning:

1. What does it mean to detect structure in a data set and how can we quantify it?

2. How can we provide guarantees that the detected structure generalizes from one sample set to a second one?

It is unrealistic to expect a general answer to the first question. A general theory of structure has not been developed yet and attempts like the inference on the basis of Kolmogorov complexity are debated. One might even argue that such a goal is completely elusive since it encompasses the program of natural science, engineering and the humanities. The different talks, therefore, covered a wide spectrum of special suggestions how structure could be defined and detected ranging from trees in image analysis, informative projections like PCA or ICA representations of high dimensional data, clusters in vectorial data and histograms as well as groups in relational data or principal surfaces.

It became apparent in the discussion of simulation results that fluctuations in the data should have little influence on the learned structures. This requirement might be enforced by bounding techniques as they have been developed for the computational learning theory of supervised learning or by information theoretic compression ideas. The challenges of unsupervised learning for the COLT and the modeling community seem to crystallize around the questions how optimally generalizing structures in data can be discovered and how they are characterized and validated in terms of robustness, compactness (description length) and efficiency for learning.

What does biology teach us about unsupervised learning? Apart from the miracle how supervised learning might be organized in the brain at

the neuronal level, the biological substrate seems to support unsupervised learning and related modeling ideas (or is at least compatible with them) by a potentially large computational power in synapses. Furthermore, spike trains might not only serve as a robust communication protocol but possibly provide probabilistic inference with an appropriate data format.

8 Program Analysis

Seminar No. **99151** Report No. **236** Date **11.04.–16.04.1999**
Organizers: Hanne Riis Nielson, Mooly Sagiv

Motivation for the seminar. Program analysis offers static techniques for predicting safe and computable approximations to the set of values or behaviours arising dynamically during the execution of programs. Traditionally, program analysis has been used in optimising compilers; more recent applications include the validation of safety properties of software for embedded systems, applications in reverse engineering and program understanding, and also program analysis shows promise of becoming an important ingredient in ensuring the acceptable behaviour of software components roaming around on information networks.

Over the years a number of different approaches have been developed for program analysis, all of which have a quite extensive literature. To give an impression of the diversity let us briefly mention some of the main approaches. The *flow based approach* includes the traditional data flow analysis techniques for mainly imperative languages, but also control flow analysis as developed for functional and object oriented languages, and set based analysis as developed for logic and functional languages. The *model based approach* includes the parameterised denotational semantics techniques developed for functional and imperative languages, but also more generally the use of mathematical modelling in the abstract interpretation of imperative, functional, concurrent and logic languages. The *inference based approach* includes general logical techniques touching upon program verification techniques, but also the type and effect systems developed for functional, imperative and concurrent languages.

Typically, the various techniques have been developed and further refined by subcommunities – with the result that often the commonalities between analogous developments are not sufficiently appreciated. To

guide the research effort in our community, it is necessary with an appraisal of the current technology. Therefore one of the primary aims of the Dagstuhl Seminar was to bring researchers from the various subcommunities together to share their experience.

The programme of the seminar. The scientific programme of the seminar consisted on six invited talks (60 minutes each) and a number of contributed talks (30 minutes each); in the evenings extensive discussions were taking place in five working groups with the goal of identifying the most challenging research problems for the next few years. The working groups were within the areas:

- Program modularity and scaling of program analysis;
- New applications of program analysis;
- Security through program analysis;
- Foundations of program analysis;
- Combining static and dynamic analyses.

The findings of the working groups were subsequently presented and discussed in plenum. The abstracts of the talks and the position statements produced by the working groups can be found in the report. Two working group statements were selected by the editor of Dagstuhl News and reproduced below.

A novel application area. At this seminar we begin to see a novel application area for program analysis: *the analysis of security problems.* As described in the position statement from the working group on "Security through program analysis" further interaction with the security community is required in order to understand the possibilities as well as limitations of our techniques; however, classical program analysis techniques have already been successfully applied to a few security problems:

- Ensuring type and memory safety and thereby detecting leaks of secure information caused by accessing memory not allocated by the application itself – this is important for protection against certain viruses.

- Detection of information flow using program dependency analysis allows one to validate certain security and integrity constraints by imposing a distinction between trusted and untrusted data, between secure and public data etc.

- Validation of security protocols by extraction of the actual protocol used by an application and subsequent analyses of its properties.

Novel results presented at this Dagstuhl seminar indicate that program analysis techniques may be superior to many of the other techniques suggested for analysing security properties: Based on a program analysis it was shown how to construct a "hardest attacker" that then can be analysed in conjunction with the program of interest so as to ensure that certain security properties are fulfilled.

Also the seminar showed quite some interests in applications of program analysis to legacy code. Within the more traditional application areas of program analysis in particular the combination of static and dynamic analysis received attention.

Unifying the community. A previous Dagstuhl Seminar on Abstract Interpretation (Dagstuhl Report No. 123, 1995) showed that there were serious gaps between the subcommunities and that there was a need for an exchange of ideas. This view was confirmed in a number of position statements written for the celebration of the 50th anniversary of the ACM (ACM Computing Surveys vol. 28 no. 4, 1996). We are very pleased to observe that today the program analysis community is moving towards a greater appraisal of competing techniques. In addition to the informal discussions between the subcommunities, a number of talks emphasised the relationship between different approaches and so did the discussions of the working group on "Foundations of Program Analysis". Also it was noteworthy that techniques developed in one area are being taken up in other areas – an example is the further development of techniques used for analysing pointer structures to analysing mobile computation.

Security through program analysis (working group)

A classification of security. Security is an important area where there is reason to believe that techniques from program analysis may

prove very helpful. This includes applications to the following ingredients of security:

- *Confidentiality* (or *secrecy*). This amounts to making sure that certain sensitive data are not becoming generally known; to model systems, several levels of confidentiality may be appropriate. The properties are mainly safety properties in the sense that certain invariants must continue to hold; we ignore here the possibility that keys used in encryption can be compromised using brute-force computational attacks.

- *Integrity.* This is based around the notion of trust, meaning the use of data or code that is deemed to not lead to any breaches of security; again this is largely a safety issue. *Authentication* is a branch of integrity aiming at preventing a document to be tampered with; a useful technique is digital signatures (which in RSA is much the same as encryption). *Watermarking* is a branch of integrity aiming at preventing the origins of a document to be obfuscated even though attempts are made at masquerading it.

- *Availability.* This deals with ensuring that certain services are always possible (e.g. for creating fresh keys that have never been used before). This is mainly a liveness issue.

- *Auditing.* This does not enforce security per se, but is a technique for ensuring accountability: that breaches of security can be traced back to the offender; here it is important to check against the possibility of performing actions that are not logged appropriately.

Clarifying breaches of security. It is difficult to perform a convincing application of the techniques of program analysis unless it is made clear what constitutes an attack. To some extent this is a "moving target" that should be addressed by the security community rather than by the program analysis community. Two facets of this are:

- How does one detect breaches of security? It makes a big difference whether one is only allowed to inspect the values communicated, or also to measure computation time, or even to monitor the areas of a chip that produce the most heat (in case an algorithm has been laid out in silicon).

- The assumptions that can be made on the attacker. As an example, is it sensible to assume that an attacker is always written in the same programming language being used for the application being analysed?

Possible answers might include a formalisation in terms of testing equivalence or one of a number of bisimulations (as has been done for security studies expressed in terms of the spi-calculus); somewhat similar techniques have already been used in program analysis for validating classical analyses (like live variables).

Identifying the important issues. It is a useful technique to reduce the size of the trusted base; for example proof carrying code only requires the checker to be trusted and not the theorem prover proving the result in the first place. As a result, scalability of the analyses will be less of an issue for program analysis; hence it may be feasible to perform relatively expensive analyses in order to gain the necessary precision.

Formal notations are likely to be indispensable in expressing the bad or unacceptable behaviour of systems. This is also related to the issue of proof carrying code in the sense that it is useful for the code to be able to contain annotations about its behaviour. A more general treatment makes use of probabilistic notions in order to perform the risk analyses: the rate at which semi-confidential data might leak. Unfortunately, it is unclear how to integrate probabilistic considerations with program analysis.

The precise details of the encryption and decryption techniques are probably less relevant for program analysis. (This presupposes that we ignore the risk of brute force cracking a code as was already stated above.)

It is also important to understand the limitations of our techniques in attempting to guarantee against attackers. Various forms of code obfuscation might be used to severely limit the abilities of program analysis to detect useful information about the information flow happening in a system (like multiple uses of the same variable for different purposes or obscuring the control structure). Also it may be necessary to integrate dynamic checks (e.g. taintedness checks) with the static analysis in order to ensure confidentiality.

Techniques from program analysis. The classical program analyses are often necessary preconditions for ensuring security; as an example

type and memory safety can be used to detect leaks of secure information caused by accessing arrays outside their declared bounds. Only when studying special purpose designed calculi (such as the pi-calculus, the spi-calculus or the ambients calculus) is it meaningful to study security without first performing the standard analyses for ensuring type and memory safety.

Several of the approaches to *program dependency analysis* (like control flow analysis, use definition chaining etc.) may form the core of analyses for detecting information flow (including implicit flows). Another useful technique for maintaining security and integrity constraints is type systems with *effect annotations* (in the form of trusted/untrusted, secure/public etc.). Both of these techniques are likely to be very useful in determining the information flow that lies at the heart of many security notions. Indeed, on top of demonstrating that certain illegal flows do not take place, they may also be useful in determining the potential breaches of security that may result from accidental or deliberate instances of compromising data (e.g. by allowing only partly trusted applications to operate on selected parts of the sensitive data).

Techniques from type and effect systems may be useful for extracting the inherent protocols used for communication in programs (including legacy code) so as to allow validating the overall protocol of a software system. This will also help detecting errors that arise when the system is implemented; similar considerations apply to deliberate code modification so as to facilitate a later attack. Also auditing can be ensured through these techniques by ensuring that data is only accessed or modified after the appropriate logging actions have been performed.

Recent research suggests that program analyses that abstract the semantics of the program might be useful for devising tests that can be used to validate key software components. To be specific, based on the analysis it may be possible to characterise an infinite set of attackers by a finite set of hard attackers; then one can simply analyse the software component under each of the attackers and validate the component in case none of the analyses exhibit illegal behaviour.

Challenges for program analysis. The following challenges are aimed at demonstrating the usefulness of program analysis for security and at studying which of our techniques are likely to be useful for security. It should be possible to obtain progress on both within a few years from

now.

- Perform a successful study of one system or protocol; either find an undetected flaw or prove that no attacks of a certain kind can succeed. (This would be useful even for toy protocols from text books such as the alternating bit protocol.)
- Identify techniques, beyond "extended reachability analyses" and "effect annotations" that are applicable; to which extent does this uncover techniques not already known in the security community?

Group members: Chris Hankin, René Rydhof Hansen, Thomas Jensen, Flemming Nielson (editor of the position statement), Jon Riecke, Hanne Riis Nielson, Andrei Sabelfeld, Mooly Sagiv and Helmut Seidl.

Foundations of program analysis (working group)

Program analysis has a number of foundational approaches. We identified ten such approaches, some of which overlap with the others. Those ten approaches, with their defining characteristics, are

1. Type based: Uses an inductive, compositional definition of a typing relation. Type-based analyses may require types or other annotations in programs.
2. Constraint based: Uses systems of equations, inequations, or, more generally, conditional constraints generated from the program by an algorithm. These systems are solved by another algorithm to yield an analysis result.
3. Abstract interpretation: Uses concrete and abstract domains of values, with concretization and abstraction functions between them that form a Galois connection. Widening and narrowing are important techniques to improve the efficiency and precision of the analyses. Algorithms in this area are based primarily on least fixpoint computations.
4. Grammar flow: A method that traces the state spaces of tree automata, and uses alternations between bottom-up and top-down phases.

5. Data flow: A methodology based on equations between sets of values and transfer functions, typically solved by least or greatest fixpoints.

6. Context-free language reachability: A method using context-free grammars, usually focusing on interprocedural problems and usually taking $O(n^3)$ time.

7. Dependence graphs: An analysis technique using data structures with control dependence and flow dependence edges; the data structures are the basis for many algorithms.

8. Temporal logics and model checking: Specifies properties of programs using temporal logics. Checking of properties (that is, determining the truth/falsity of them) is done by model checking. This technique generalizes many ideas from the data flow approach, but is different in using logic to specify "elements" of data flow sets rather than using sets directly.

9. Denotational based: Uses a compositional translation from syntax into mathematical spaces of values. These mathematical spaces, typically called "domains," can be based on syntax as well as partial orders. Strictness analysis is an example.

10. Abstract reduction: Uses a nonstandard rewrite semantics, often requiring an interesting form of loop detection via a syntactic notion of "widening." This technique is used in the CLEAN system, a compiler for a lazy functional language.

Some simple analyses, e.g., Barendregt's neededness analysis for the untyped lambda calculus, do not seem to fit naturally into one of these classifications. We may also have missed some frameworks, since not all analysis communities were represented.

Criteria for choosing an analysis framework

When choosing a suitable framework for approaching an analysis problem, a number of criteria can be used:

1. Utility: Does the framework naturally express the analysis problem?

2. Reliance on syntax: Does the approach rely on the syntax of the language, or does it use some other generic data structure?

3. Specification cleanliness: How well does the approach separate specification of "what" the analysis does from "how" the analysis is performed?

4. Algorithmic issues: Does the analysis suggest an obvious algorithm? An efficient algorithm?

5. Expressivity: Can one formalism express the analysis as well as another? How succinctly can the analysis be expressed? Can an analysis be done as efficiently in one formalism as it can in another? For instance, temporal logics seem strictly more powerful than data flow approaches.

6. Scalability: How well does an analysis in one formalism scale to large programs?

7. Modularity: How modular is the specification of an analysis? Can it be extended easily to larger programs, or to other forms of analysis?

8. Semantics directedness: Does the framework force the analysis to look like the semantics of the language?

9. Tools: Does the framework admit the construction of tools? How wide is the coverage of these tools?

Challenges

1. Low-complexity analyses: Are there methods in the various frameworks for constructing lower complexity analyses? Techniques from different frameworks may need to be combined.

2. Reductions: Can one construct reductions between different analyses written in different frameworks, or between different frameworks themselves? If so, do these reductions preserve complexity bounds of the analyses?

3. Expand scope of foundations: Can one reformulate certain analyses as type systems or in one of the other frameworks? Foundational work should promulgate the use of frameworks in analyses that

don't seem to use them, and should consider these analyses in potential expansions of the frameworks.

4. Tools: Can one say general things about how much of a particular framework—that is, how many analyses developed in a certain framework—a tool covers? Can one expand the coverage of tools using foundational methods?

5. Open systems: Can one devise or extend analysis frameworks to dealing with open systems (i.e., those with processes, mobile code, or simply modules)?

6. Probabilities: Can one devise foundational understandings of analyses that use probabilities (e.g., an analysis that predicts, with probabilities rather than yes/no, when a definition reaches a use)?

7. Redundant code: Are there ways of building analyses that eliminate redundant code?

Group members: Chris Hankin, Jörgen Gustavsson, René Rydhof Hansen, Thomas Jensen, Flemming Nielson, Hanne Riis Nielson, Jon G. Riecke (editor of the position statement) and Mooly Sagiv.

9 Instruction-Level Parallelism and Parallelizing Compilation

Seminar No. **99161** Report No. **237** Date **18.04.–23.04.1999**
Organizers: Damal K. Arvind, Kemal Ebcioglu, Christian Lengauer, Keshav Pingali, Robert S. Schreiber

Introduction

Parallel programming has been around for three decades and has remained a difficult field. The biggest challenge arises when the main purpose of parallelism is to increase performance, i.e., computation speed. Parallel programs are notoriously hard to get correct and efficient. Although progress has been made on the semantics and verification of parallel programs in certain domains, no practical technique for the devel-

opment of reliable, portable application parallelism for high performance has been achieved.

One approach towards this goal is to unburden the programmer from the difficult task of handling parallelism and delegate this to the compiler or the machine architecture. The research area which gives the compiler the control over the parallelism is *parallelizing compilation*, the research area which lets the machine infuse the parallelism is *instruction-level parallelism* (*ILP*).

The aim of the seminar was to bring together these two research areas, which have developed side by side with little exchange of results. Both areas are dealing with similar issues like dependence analysis, synchronous vs. asynchronous parallelism, static vs. dynamic parallelization, and speculative execution. However, the different levels of abstraction at which the parallelization takes place call for different techniques and impose different optimization criteria.

In instruction-level parallelism, by nature, the parallelism is invisible to the programmer, since it is infused in program parts which are atomic at the level of the programming language. The emphasis is on driving the parallelization process by the availability of architectural resources. Static parallelization has been targeted at very large instruction word (VLIW) architectures and dynamic parallelization at superscalar architectures. Heuristics are being applied to achieve good but, in general, suboptimal performance.

In parallelizing compilation, parallelism visible at the level of the programming language must be exposed. The programmer usually aids the parallelization process with program annotations or by putting the program to be parallelized in a certain syntactic form. The emphasis has been on static parallelization methods. One can apply either heuristics or an optimizing algorithm to search for best performance. Resource limitations can be taken into account during the search, or they can be imposed in a later step, e.g., through tiling or partitioning the computation domain.

Summary of the Presentations

Embedded software applications have traditionally tended to use low-level languages and hand-crafted techniques for optimizing execution time and memory usage. Given the scope for exploiting parallelism in

embedded software, especially in multimedia applications, and the emergence of ILP processors, such as VLIW ones, there is a growing body of work investigating automatic parallelization of high-level programs aimed at ILP targets. A number of these compiler infrastructure projects were presented at the workshop.

The Esprit OCEANS project (Eisenbeis) is aimed at embedded VLIW architectures, with emphasis on understanding and exploiting interactions between high-level optimizations, such as loop unrolling, and low-level ones, such as software pipelining

The ACROPOLIS project (Omnès) at IMEC considers the impact of data organization in embedded applications on performance metrics such as instruction throughput and power consumption. The latter is fast becoming an important consideration for multimedia applications running on mobile appliances. The approach in this project is to inform the choices in the parallel compilation process of the impact of the dominant costs of data transfers and complex data manipulations on the overall performance.

The TriMedia project (Augusteijn) at Philips has developed a compilation environment for embedded programs written almost exclusively in C/C++, and targeted at the 5-issue TriMedia VLIW processor. It supports predicated execution and special operations for DSP algorithms, such as vector instructions on subwords.

Vectorizing techniques for exploiting sub-word parallelism in ILP architectures was the subject of two further talks from the University of Vienna (Krall) and the Indian Institute of Sciences, Bangalore (Govindarajan); and the method of predicated execution for exploiting ILP in the EPIC (Explicitly Parallel Instruction Computing) architecture was the subject of a paper from University of California, San Diego (Ferrante).

The traditional instruction sets of processors have been extended to exploit efficiently sub-word parallelism, in which a number of short data elements are packed in a single register and data-parallel operations are executed on them in parallel. Examples of these so-called multimedia extensions include, the Visual Instruction Set for the UltraSPARC processor, the AltiVec for the PowerPC, the MMX extension for the Pentium processor, and the MAX-2 instruction set of the PA-RISC processor. At present, there is little or no compiler support to exploit sub-word parallelism – the user is expected to handcode large parts of their application

in assembly language.

Krall uses the technique of vectorization by unrolling to automate this process. Data dependence analysis and dynamic run-time checking are used to handle unaligned memory accesses. Govindarajan uses standard vectorization techniques on loops which are tailored for short vector lengths.

Predicated execution is one of many approaches used to finding instructions that can be executed simultaneously in ILP architectures. One of the drawbacks, however, is that predicated code presents challenges to traditional compiler optimizations. Ferrante presented an extension to the well-known Static Single Assignment form, called the Predicated Static Single Assignment form, which, when used in conjunction with speculation and control height reduction, enables instructions to be issued at their true data dependence height.

The microarchitecture of future processors has been the subject of intense debate and was the topic of two talks – from the University of Wisconsin (Sohi) and the University of Delft (Corporaal). Both speakers recognized common problems – future architectures have to contend with increasing workloads and longer communication and memory latencies – but they advocated quite different solutions. Microarchitectures have traditionally been based on certain observable program behaviours, such as spatial and temporal locality. Sohi advocated the need for information about program structure – the data and control relationship between instructions, i.e., the relationship which causes the observable behaviour – to be the basis for the design of future microarchitectures. Corporaal recognized communication as being of primary importance in the design of future microarchitectures. In the transport-triggered architectures, for instance, the communication between functional units, and with the register files, are programmed explicitly; the computation is now a side-effect, triggered by the communication. All communication inside the microarchitecture is visible to the compiler, which leads to a number of communication-level optimizations that the compiler can perform to increase the performance. Embedded programs can be analyzed and implemented on a transport-triggered architecture with an optimal number of functional units and communication pattern.

Embedded systems have requirements such as low cost and low power consumption, in addition to high performance. In many cases, off-the-shelf processors cannot meet the performance specification. The design

of embedded systems tuned to a particular application in a given domain demands an approach which takes an integrated view of the software and hardware design of the system. Cadence Design Systems (Martin, Hoover) presented an approach to performance estimation of software running on the processor by using a virtual instruction set model, and a scheduling model for the real-time operating system.

Looking beyond ILP, talks from the University of Minnesota (Yew), Chalmers University (Stenström), and the University of Jena (Unger) presented ideas on compilers that exploit thread-level and instruction-level parallelism. Yew discussed the Agassiz compiler which is targeted at a concurrent multithreaded architecture and supports speculative execution at both the thread and instruction level, in addition to run-time data dependence checking and very fast communication between thread processing units. High-level program information, such as aliases, and cross-iteration data dependences, are passed from the thread-level compiler to the ILP one. Unlike Yew, Stenström does not assume that the target architecture supports speculative execution. His ideas for thread-level data speculation are implemented entirely in software. The aim is to demonstrate that the overhead is acceptably low with reasonable performance gains through the exposed parallelism. Unger presented the Simultaneous Speculation Scheduling, which is a combined compiler and architecture technique for multithreaded processors. The speculation is controlled entirely by the compiler and is aimed at simultaneous multithreaded processors.

A number of talks discussed the dependence analysis of programs and their optimization for ILP processors. The paper from the University of Jena (Zehendner) describes a method for memory reference disambiguation on assembly language code for increasing ILP. The method derives value-based dependences between memory operations and is integrated in the SALTO system. The presentation from the University of Versailles (Collard) looked at statically deriving the probability, in the case of arrays, that two references access the same memory location. This is useful in moving a load speculatively above a possibly aliasing store. Mills Strout from the University of California, San Diego, presented a method of register tiling which exploits data dependence analysis to reduce storage requirements in superscalar ILP architectures. Jens Knoop from the University of Dortmund presented an automata-theoretic approach to interprocedural data flow analysis. The structural behavior of the program is modelled by an appropriate pushdown system; the reaching definitions

problem boils down to a reachability problem on pushdown systems.

One dominant target of parallelizing compilers is the domain of nested loop programs. A number of presentations in the seminar came from this domain.

The computation domain of a loop nest is often modelled by embedding the loop steps on a high-dimensional integer grid. One problem which arises is how to use the structure of this domain to advantage for an efficient execution. Griebl from the University of Passau presented an algorithm to shorten the parallel execution which uses breaks in the dependence structure of a polyhedral computation domain. Quilleré from IRISA in Rennes presented a method for the execution of domains which are unions of polyhedra. Rather than testing at run time, whether a computation point falls into the domain or lies outside – which can lead to a lot of overhead – he splits the domain into pure polyhedra and scans these without any run-time tests. Darte from the ENS Lyon uses a combination of loop shifting and loop compaction to shorten the parallel execution of a program composed of separate loop nests. Robert from the same school extends static techniques of partitioning the domain to the context of limited computational resources with different-speed processors.

At a lower level of abstraction, Coelho restructures mathematical expressions to evaluate them more efficiently. Gregg addresses the software pipelining of loops with branches in the loop body. Moore presented work at the University of Frankfurt on associative architectures for the support of run-time parallelization.

Conclusions

The seminar exposed the exciting developments taking place in parallel computer architecture. It also exposed the heavy burdens which are being placed in compilers by current parallel machines. The efficient use of performance-increasing hardware such as cache hierarchies, pipelined functional units and predication calls for highly sophisticated analysis and code generation techniques. It remains to be seen how the portability of parallel software can be maintained in this scenario. Portability is essential. After all, a parallel computer whose main purpose is high performance becomes obsolete after about five years.

An issue of the International Journal of Parallel Programming dedicated to this seminar will appear in due course.

10 High Level Parallel Programming: Applicability, Analysis and Performance

Seminar No. **99171** Report No. **238** Date **25.04.–30.04.1999**
Organizers: Murray Cole, Sergei Gorlatch, Jan Prins, David Skillicorn

It is generally acknowledged that programming parallel computers effectively and correctly is a conceptually challenging task for all but the simplest of applications. Consequently, there is widespread research interest in models and methodologies which can assist the process. In order to provide some degree of durability, such approaches must abstract from the detailed characteristics of specific systems, while remaining efficiently implementable by those systems.

The previous Dagstuhl Seminar 9708 brought together a spectrum of researchers with interests related to the "higher order" aspects of this area, ranging from those with theoretical interests in program development to practical systems builders. The present seminar has aimed to focus more closely on the developments in two of the areas which emerged at the original workshop as being on the critical path to progress to include approaches which are more generally "high level" rather than specifically "higher order". By "high level" we mean going beyond the simple extension of sequential languages with communications or shared data primitives, to models and languages in which the expression of conceptual structure is encouraged and supported.

Most interest in parallel programming is motivated by the quest for dramatically improved performance in processing large applications. To gain credibility with that community, we must be able to show that our methods are competitive. The quantification of performance is simplified by the computational structure inherent in the high level approach. This applies equally to attempts to predict performance on the basis of static program analysis and a small number of architecture specific parameters (more commonly known as "cost-modeling") and to the benchmarking and post-hoc analysis of the behaviour of implemented systems.

Similarly, we must be able to demonstrate convincingly that high

level parallelism enhances the *applicability* of the underlying technology by simplifying the expression of real programs for real problems (rather than the sanitized and simplified examples appropriate to the early stages of research). Paradoxically, our target audience must also be convinced that there is no loss of expressiveness when dealing with those parallel programming sub-tasks in which there is no well behaved structure to capture.

In summary the following questions and the many subordinate issues they raise were addressed during the seminar:

- How well can high-level parallel programming methods match the performance of more machine specific approaches?
- What do (and should) we mean by performance in this context?
- Can such systems be effectively cost-modelled, and if so, would this be an attractive feature to practitioners?
- Can we support such models with other conceptual tools in ways which enhance their attractiveness?
- Can the tension between the use of abstraction and the requirement for detailed ad-hoc control in certain problems be satisfactorily resolved?

In order to ensure that contributions remain focused on cost and applicability, and to provide some common ground on which debate can be conducted, we circulated the participants well in advance with two realistic problems to act as case studies: (1) the Frequent Sets Problem in data mining, and (2) the Barnes-Hut algorithm for the N-Body problem. Both problems are important in practice but have not been treated well to date because of methodological limitations. Lively and deep discussions on both problems, in particular about such evaluation criteria as succinctness, correctness and clarity, as well as performance, contributed a lot to the success of the seminar:

- The Frequent Sets Problem is one of the basic building blocks of many data mining algorithms. Suppose that an organization has recorded the set of objects purchased by each customer on each visit. The goal of the frequent set problem is to find those (smaller) subsets of objects that appear in more than a given fraction of the

sets. This information can be used to, for example, place objects that are often purchased together near each other on the shelf. The algorithm is also applicable in scientific domains, for example to find the "interesting" parts of complex simulations.

Given a set M (all of the possible objects that can be sold) and a bag N of subsets of M (each element of the bag records the subset of objects purchased by one customer in one visit), find all subsets of M that appear in more than a fraction x of the elements of N.

The problem is trivial in the sense that there is an obvious algorithm. However, the size of the data concerned is so large that it becomes critical to do as little work as possible. Clever algorithms are necessary.

There have been two main approaches. The first depends on the observation that a set can only be frequent if all of its subsets are frequent. This reduces the number of sets whose frequencies need to be checked. A summary of this approach can be found at
`http://www.cs.helsinki.fi/~htoivone/pubs/toivonen.ps.gz`
in Toivonen's thesis. The other tries to use the vast mathematical literature on lattices to improve the search. An example is the work of Zaki at Rochester:
`http://www.cs.rochester.edu/trs/system-trs.html`

The frequent set problem fits well with calculational approaches in the sense that it is straightforward to write down a solution, but harder to transform it to an efficient solution.

This problem was presented to Dagstuhl participants in advance of the workshop. During the workshop, two attacks on the problem were made. Zhenjiang Hu was able to derive a much-improved version of a functional implementation. Its performance was compared to a direct implementation by Christoph Herrmann. On some small synthetic datasets, performance improvements of an order of magnitude were demonstrated. Second, Charles Leiserson pointed out that an approximate algorithm due to P. Gibbons and J. Matias for computing distributions in datasets might be applicable to the problem. Discussion along this line took place during the meeting, although no substantial progress was achieved (the approach has since been extended and shown to work well, however).

- The N-Body Problem: given a self-gravitating system consisting

of n distinct particles characterized by their mass, initial position, and velocity, the problem is to compute the force on each particle that is induced by the other particles.

A direct force calculation would require the computation of $O(n^2)$ interactions, a large amount of work for particle systems encountered in practice. There exist a variety of methods that compute approximations to the exact solution with reasonable accuracy and with an improved asymptotic complexity.

The Barnes-Hut hierarchical force-calculation algorithm exploits the fact that, at a distance, the combined potential of a group of particles can be approximated by the potential of the center of mass of that group. The algorithm makes use of a hierarchical quad-tree (or oct-tree in 3D) decomposition of the space containing the particles, and associates with each region its center of mass. After the tree is built, the force calculation traverses the tree top down for each particle to accumulate the total force on the particle. Subregions are explored only if the region's center of mass is not sufficiently far away from the particle to be used as an approximation.

The treeForce computation for each particle is independent and can be computed in parallel. Moreover, the recursive force-computations in the tree traversal are independent and can be computed in parallel. The parallelism specified in treeForce is dynamic since the available parallelism increases with the depth of the recursion. It is irregular because the degree of parallelism specified and the locality of the interactions depend on the distribution of the particles.

This problem has been suggested for consideration in Dagstuhl because the algorithm can be succinctly expressed in a high-level notation, yet an efficient implementation is challenging. Furthermore, some comparative performance data are available for low-level implementations.

In order to encourage critical comment on our ideas, we invited a small number of open-minded participants who are prominent in the use of currently dominant parallel programming technologies (such as MPI, HPF and multi-threading). Their pragmatic perspective on our proposals was illuminating.

11 Mobile Multimedia Communication – Systems and Networks

Seminar No. **99061** Report No. **239** Date **04.05.–06.05.1999**
Organizers: Andrew Campbell, Ernst Rolf, Stephen Pink, Martina Zitterbart

Wireless and mobile communication are becoming increasingly important for various application areas going far beyond pure telephony. Just to name a few trends: body area networks, sensor-based networks and multihop ad hoc networks are of great interest leading to new infrastructures and propelling new applications in the context of ubiquitous computing. Furthermore, it is obvious that multimedia data need also to be transported through wireless networks or through a wireless access. Moreover, multimedia data will be exchanged during mobility and, thus, require proper networking support, for example, regarding continuous delivery and the like.

Mobile communication asks for both, proper network solutions including protocols and mechanisms as well as good systems including issues related to power management and hardware/software co-design. This Dagstuhl seminar on mobile multimedia communication has brought together researchers from both groups (systems and networks) to stimulate interdisciplinary discussions.

The report gives an overview of the presentations given during the seminar and reflects the current research in both systems and networks for mobile multimedia communication. Although the majority of topics was related to networking issues (e.g., active networking, Quality of Service, protocol header compression), issues considering systems were also discussed, for example timing analysis of mobile communication systems.

12 Geometric Modelling

Seminar No. **99201** Report No. **240** Date **16.05.–21.05.1999**
Organizers: Hanspeter Bieri, Guido Brunnett, Gerald Fagin

The field of Geometric Modeling emerged from the need for efficient tools for geometry handling in CAD/CAM applications. Initially, it was mainly concerned with the development of modeling techniques for continuous free-form geometry as Bezier and B-spline surfaces. Since efficient geometry processing is important for a variety of applications apart from CAD/CAM modeling tools have been demanded that correspond to the special requirements of theses applications. In this way applications served as the driving force for the development of the field. Today, Geometric Modeling addresses modeling problems that involve continuous as well as discrete geometry and arise in application areas as Scientific Visualization, Robotics, Optimal Control, Biomedicine and CAD/CAM.

Out of the diversity of topics discussed on the workshop we would like to highlight the following issues.

1. **Reverse Engineering**

   is concerned with the automatic generation of a CAD model from a point set that has been digitized from an existing 3D object. The reverse engineering process can be divided into four main phases: data preprocessing, segmentation, surface fitting and CAD model creation. During the preprocessing step the data is organized such that it is possible to estimate local properties of the surface to be reconstructed. This information may then be used to group the points into segments appropriate for the surface fitting step.

   On the workshop a new algorithm has been presented for computing a triangulation of the data set as an intermediate step of the reconstruction. The method approximates the Delaunay tesselation of the object's surface. Thus, it intends to generalize the common Delaunay triangulation to arbitrary 2D manifolds in Euclidean 3 space. In contrast to previous algorithms it combines generality with computational efficiency. In a different talk a new approach towards automatic segmentation has been presented. Here, the computation of a triangulation is completely avoided. Instead a hashing strategy is used to compute a neighborhood graph that serves as the basic structure of the point set. This methods is extremely fast. It allows the automatic segmentation of large data sets (several 100 000 points) in the range of seconds.

2. **Geometric Modeling for Scientific Visualization and Simulation**

An important issue of Scientific Visualization is the realistic modeling of fluid flow. In its most general setting fluid flow is governed by a system of non-linear partial differential equations known as the Navier-Stokes equations. However, in several important settings, these equations degenerate into simpler systems of linear partial differential equations. On the workshop it has been shown that flows corresponding to these equations can be modeled using subdivision schemes for vector fields. With this approach realistic flows can be modeled and manipulated in real time.

In a different talk new results have been reported in the development of parametric descriptions of biological objects of complex shape, e.g. the human heart. The method described is based on partial differential equations as modeling elements and allows a realistic analysis of the functioning of the models.

3. **Error Propagation in Geometric Constructions**

   Surprisingly little is known on the propagation of input errors on the result of geometric constructions. On this workshop a theoretical analysis for the most simple type of geometric construction, i.e. affine combination, has been given and the consequences for spline curves which are created by repeated affine combination of control points have been discussed.

4. **Solid Modeling**

   is concerned with the processing of volumetric objects for design purposes. The representation schemes used for solid modeling deliver complete information on the object that has been designed. This allows to perform a consistency check on the model. On the workshop several talks were devoted to this subject. A new type of solid has been presented that allows dimension independent geometric modeling. For a specific type of orthogonal polyhedra a new extremely concise data structure has been described. It has been shown that volumetric properties of solids can be efficiently computed using low-discrepancy sequences. Finally, methods have been investigated for constrained based modeling in a mixed environment of free form curves, surfaces and solids. Especially, it has been demonstrated how the curve-surface incident relation can be maintained while the curve is edited.

13 Graph Decompositions and Algorithmic Applications

Seminar No. **99231** Report No. **241** Date **06.06.–11.06.1999**
Organizers: Andreas Brandstädt, Stephan Olariu, Jerry P. Spinrad

There are many notions of graph decomposition which arise in the literature. Some decompositions involve decomposing a graph using separators of special types (balanced or polynomially bounded, star cutsets, clique cutsets), others involve identification of special sets (substitution or splits), while others involve tree decomposition (treewidth, cliquewidth, branchwidth) or tree composition (Cartesian product, lexicographic product).

These decompositions are of fundamental importance for solving optimization and recognition problems on classes of graphs. For example, substitution decomposition is closely related to such problems as solving problems expressible in monadic second order logic quantifying over vertices and/or edges and comparability graph recognition and optimization. Treewidth and its generalizations are of special importance due to the Robertson-Seymour results on tree decomposition and existential proof of existence of algorithms. Clique cutsets and star cutsets are fundamental tools used in the study of chordal and perfect graphs. Particular tools for working with these decompositions, such as partition refinement and lexicographic breadth first search, have recently been improved and generalized in this context.

This seminar was designed to bring together researchers working on a variety of aspects of graph decomposition. Talks were given studying special classes of graphs, new decomposition techniques and optimization algorithms, and data structures which allow faster decomposition algorithms.

We had 37 participants from Austria, Brazil, Canada, France, Germany, Hungary, Italy, The Netherlands, Norway, Republic of China, Switzerland and USA. During the seminar 25 lectures were given. Moreover, two evening sessions presented open problems.

Jens Gustedt, editor of the electronic journal DMTCS (`http://dmtcs.loria.fr/`) proposed to the organizers to edit a special volume of this journal devoted to our Dagstuhl seminar.

14 Requirements Capture, Documentation, and Validation

Seminar No. **99241** Report No. **242** Date **13.06.–18.06.1999**
Organizers: Egon Börger, Bärbel Hörger, David Parnas, Dieter Rombach

The goal of the workshop, namely to bring together software engineering researchers from academia and software engineers from industry to compare the state of industrial practice and academic research for capturing, documenting and validating software requirements, has been reached.

After two days of short introductory presentations, with ample time for critical discussion, we had two days of intensive discussion in working groups.

The three themes

- Integrating Process, Tools and Formal Methods (moderator Connie Heitmeyer),
- Requirement Engineering Process, Evolution of Requirements and Traceability (moderator Barbara Paech),
- The Light Control Case Study (moderator E. Börger)

were selected by the participants on Tuesday evening, the results obtained were presented to all participants during the closing session on Friday morning. Reports by the moderators of the working groups can be found in the report.

The focus of the presentations and discussion was on the industrial strength of the used methods and on their relevance for the production of large software.

To make sure that the discussion was suitably concrete, the workshop made extensive use of a case study that could be discussed in detail. The example, taken from the area of building automation, was a light control system.

15 Competitive Algorithms

Seminar No. **99251** Report No. **243** Date **20.06.–25.06.1999**
Organizers: Amos Fiat, Anna Karlin, Gerhard Woeginger

Decision making can be considered in two different contexts: making decisions with complete information, and making decisions based on partial information. A major reason for the study of algorithms is to try to answer the question: 'Which is the better algorithm?' The study of the computational complexity of algorithms is useful for distinguishing the quality of algorithms based on the computational resources used and the quality of the solution they compute. However, the computational complexity of algorithms may be irrelevant or a secondary issue when dealing with algorithms that operate in a state of uncertainty. 'Competitive' analysis of algorithms has been developed in the study of such algorithms.

Competitive analysis is useful in the analysis of systems that have some notion of a time progression, that have an environment, that respond in some way to changes in the environment, and that have a memory state. Competitive analysis is used for so-called 'on-line algorithms' that have to respond to events over time. Competitive analysis is used whenever the nature of the problem is such that decisions have to be made with incomplete information.

16 Foundations for Information Integration

Seminar No. **99261** Report No. **244** Date **27.06–02.07.99**
Organizers: Serge Abiteboul, Dana Florescu, Alon Levy, Guido Moerkotte

We are currently witnessing an explosion in the amount of information that is available on-line (e.g., sources on the Internet, company-wide intranets, etc). Providing easy and efficient access to this information — known as the problem of *data integration* — raises an important challenge to several fields of Computer Science including Database Systems, Artificial Intelligence, Operating Systems, Networking and Human Computer Interaction.

The challenge is to develop techniques for providing uniform access to the wealth of available information. Usually, data integration is achieved by providing the user a mediated schema that hides the details of each of the data sources, and lets the user focus on specifying what he wants, rather than specifying how or where to find the information. The data integration problem is complicated by the fact that the data sources are autonomous, employ different data models and are heterogeneous both semantically and syntactically. Furthermore, the data sources are often only semistructured (e.g., they do not have explicit schemas, the schemas are unknown, or the sources contain extraneous information such as advertisements or other information meant for human consumption). The techniques that need to be developed include modeling of the contents of information sources, high-level query facilities, flexible approaches to selecting relevant sources, novel methods for query optimization and flexible query execution models.

17 Agent Oriented Approaches in Distributed Modeling and Simulation: Challenges and Methodologies

Seminar No. **99271** Report No. **245** Date **04.07.–09.07.1999**
Organizers: Paul Fishwick, Adelinde Uhrmacher, Bernard Zeigler

Metaphor plays a key role in Computer Science and Engineering. New ideas and methods are infused into existing areas, creating fresh material and methodologies. An "agent" carries out our purpose in performing an act on our behalf. The achievement of this purpose may involve differing degrees of autonomy. If we take the everyday meaning of "agent" and marry this concept with software development, we derive the "software agent".

Software agents address the demand for programs that inter-operate to solve problems in an open and dynamic environment. As the number and complexity of agent-oriented programs increases, so does the need for software engineering tools and simulation systems that support their design and evaluation. New research questions whether agent-oriented techniques hold part of the answer to some urgent problems in engineering simulation systems, such as how to facilitate reuse and exchange of

models and services between simulation systems.

Simulation of Multi-Agent Systems

Agent-based systems are often safety critical, and like other software systems, must be tested and evaluated before being deployed. Agents are embedded systems, and their dynamic behavior determines their efficiency and effectiveness. Therefore, simulation is an intrinsic part both during development and for testing purposes, to learn more about their behavior or investigate the implications of alternative architectures and coordination strategies.

However, work to date has largely ignored recent work in simulation methodology and systems and has instead tended to employ various adhoc approaches to simulation. The model of the environment the agent shall be tested in and the simulation mechanisms are typically developed and implemented from scratch. In setting up the test environment, modeling and the execution of the model are not distinguished. This hampers a reuse of test scenarios, reproducing the results of experiments, and comparison of results achieved through experimentation.

The requirements of a simulation system which has a chance to be applied by working groups that design agents have to meet a variety of requirements. Its model design should ideally support:

- a compositional, hierarchical model design
- an integration of diverse agents and agent architectures
- a comfortable interface to plug in agents' modules
- a dynamic adaptation of the model's composition and coupling structure
- a combination of continuous and discrete models

Model execution should be clearly separated from model design. Simulation techniques are required which combine a flexible model design (see above) with an efficient execution. The computational requirements of simulations of agent-based systems exceed the capabilities of single platforms. Each agent requires typically considerable computational resources, and many agents may be required to investigate the behavior

of the system as a whole. Distributed, concurrent simulation techniques have to take into account that:

- to determine a lookahead in the domain of deliberative multi-agent systems is very difficult since during test runs the time needed for deliberation often varies drastically.
- rollbacks are even more storage expensive due to the flexible structures which require not only the storage of states but of entire models
- a distributed execution necessitates dynamic load balancing

Simulation systems for multi-agent systems should also be able to interact with other simulation systems. This leads us to the second focus of the discussions, i.e. employing agents to execute models and to implement flexible, state of the art simulation systems.

Software Agents for Distributed Modeling and Simulation

Distributed modeling and simulation imply a geographically distributed set of models and their components, as well as concurrent execution of model-derived code. In distributing model components, we must design model and component repositories over the web. It should be possible to search these repositories for reusable objects. Although existing mechanisms do not yet exist for distributed model repositories, certain technologies such as object oriented databases and XML (Extensible Markup Language) will help in creating appropriate vehicles for model and component representations. Distributing the simulation (i.e., model execution) is another matter and has been more widely studied in the distributed simulation research community. For both model design and execution, agent-oriented approaches create a natural fit with the problems of distributed modeling and execution.

Simulation Systems Executed by a Community of Agents

The execution of a model can be realized by a community of distributed, concurrently interacting, and moving entities. Their interaction, composition and location structure adapts itself to improve the efficiency of the

simulation run. Different parts of the model can be executed by different agents specialized in different formalisms, e.g. continuous and discrete models. To balance the work load processes migrate from one site to another during simulation. The flexibility of these approaches promise a scalability, which is necessary to deal with heterogeneous, large-scale applications.

Simulation Systems as a Community of Heterogeneous Agents

Simulation systems can be developed as a community of heterogeneous agents. These approaches can be rooted back to research of the late 80ties where knowledge based systems and simulation systems were combined, e.g. to select test data, evaluate and display simulation results. A multi-agent design emphasizes modularity, flexibility, and concurrency in constructing intelligent simulation systems where modules, e.g. so called intelligent "front-ends" or "back-ends" work as autonomous agents. Depending on the functionality, e.g. if searching for specific data on the web, a module's performance might even benefit from the mobility of an agent.

Simulation Systems as Agents

Due to the diversity and multitude of simulation systems and existing simulation models, the need for standardization and an improved interoperability have been recognized as pressing problems in this area. To facilitate the exchange and reuse of models, and services between simulation systems, recent research suggests exploration of agent-oriented techniques, including knowledge interchange languages and protocols for interaction and negotiating.

The entire simulation system becomes an agent. If different simulation systems shall interoperate, as the DoD initiative HLA requires, ensuring that simulation systems "understand" each other becomes crucial. Having the same semantics refers to the objects, i.e. variables, which are exchanged between simulations, but also to the temporal horizon of the simulation systems.

Conclusion

The agent metaphor very nicely supports the development of state of the art simulation systems, since it complements the main stream of current simulation research. However, agents do not solve problems by themselves. Interfaces, semantic frames, and collaboration strategies have to be defined and negotiated. Distributed model design and model execution techniques are found to be not only supportive, but also necessary for the systematic design and testing of multi-agent systems.

18 Parallel and Distributed Algorithms

Seminar No. **99291** Report No. **246** Date **18.07.–23.07.1999**
Organizers: Bruce Maggs, Ernst W. Mayr, Friedhelm Meyer auf der Heide

The presented talks covered a wide range of topics, like routing, load balancing, accessing global variables, graph partitioning, and many more. Furthermore, during an additional evening session, several open problems have been discussed.

19 Computer Science in Astronomy

Seminar No. **99321** Report No. **247** Date **09.08.–12.08.1999**
Organizers: Walter Oberschelp, Wilhelm Seggewiss, Reinhard Wilhelm

After reliable astronomical software had proved that a total solar eclipse would take place on August 11, 1999, the participants of the Dagstuhl Seminar on Computer Science in Astronomy who happened to gather in Dagstuhl during the corresponding week decided to visit an exposed place near Saarlouis to watch the eclipse. The wheather was rather unsupportive. However, from 20 seconds before until 20 seconds after the totality the sky was open and displayed the eclipse in all nuances. (The photographs were taken by Arnold Oberschelp.)

20 Linear Logic and Applications

Seminar No. **99341** Report No. **248** Date **22.08.–27.08.1999**
Organizers: Richard Crouch, Josef van Genabith, Valeria de Paiva, Eike Ritter

Introduction

Linear Logic was introduced by J.-Y. Girard in 1987, and has attracted much attention from computer scientists as a logical way of coping with resources and resource control. The basic idea of Linear Logic is to consider formulae in a derivation as resources, which can be consumed or produced. To realize this basic idea we consider a logical system where the duplication and/or erasing of formulae must be explicitly marked using a unary operator, called an exponential or a modality. Rather than an alternative to Classical Logic, Linear Logic is a refinement of it: using Girard's well-known translation we can investigate the usage of resources in proofs of the traditional theorems. Linear Logic shares with Intuitionistic Logic a well-developed proof-theory and shares with Classical Logic the involutive behavior of negation, which makes its model theory particularly pleasant.

More importantly, because resource control is a central issue for Computer Science, Linear Logic has been applied to several areas within it, including functional programming, logic programming, general theories of concurrency, syntactic and semantic theories of natural language, artificial intelligence and planning. Several sessions in prestigious conferences

like LiCS (Logic in Computer Science) as well as whole conferences (Cornell 1993, Cambridge 1995, Tokyo 1996, Luminy-Marseille 1998) have been devoted to Linear Logic, a measure of the penetration of these ideas in the community.

Report

The Dagstuhl seminar Linear Logic and Applications contained talks concerning a number of different application areas of linear logic, as well as talks on more foundational issues.

(a) There was a large representation of linguistic applications. A repeated theme was a need for proof search methods that could efficiently uncover all distinct proofs of a given formula from a set of premises; distinct proofs either corresponding to distinct parses of a sentence, or distinct semantic interpretations of a parse. Many of the linguistic applications employed some version of categorial grammar, which through the Lambek calculus bear close connections to linear logic. However, the applications were not limited to this one style of linguistic theory: there were presentations on how linear logic could account for the resource sensitive nature of minimalism, how it could be used for rewrite rules in machine translation, and how categorial semantics could be combined with other grammatical theories. There was additionally discussion of how proof nets could account for observed processing costs in different types of linguistic construction. There was also discussion of how Linear Logic could be used to encode pragmatic distinctions (in natural language meanings), e.g. between assertions of fact, obligations.

(b) Applications to verification and specification. There were two presentations on using linear logic for formal specification and verification. Both stated that linear logic appeared to provide a compact and intuitive representation for a variety of problems, as compared to many other logical specification languages. However, the need for a combined linear and temporal logic was clearly felt.

(c) There were also presentations on the application of linear logic to functional programming. There has been a longstanding hope that the resource/usage counting aspects of linear logic could be

used to allow efficient memory management / garbage collection in functional languages. Presentations showed how this goal could be met in a number of useful special cases, but higher-order functions continue to raise problems for the more general goal.

(d) Semantics of linear logic. The seminar included talks on game semantics for Linear Logic as well as semantic spaces useful for linguistic representations.

(e) Proof theory, search, complexity and syntax. Talks on proof search and checking included a discussion of potentially incremental connection methods, and the presentation of a linear time algorithm for multiplicative linear logic. There was also a talk on the complexity of proofs, using their graph-theoretical properties and also presentations on the treatment of quantifiers as infinite tensors and pars, as well as the significance for various translations / embeddings between logics.

(f) The semantics of resources and bunched implications. The seminar also contained three presentations that, collectively, argued that linear logic should not be viewed as "the" logic of resources. Thus while Linear Logic may serve well as a process logic, and as providing a better understanding of proof and the nature of some familiar connectives, it was argued that it fell short in its account of resources. It was argued instead that a logic of bunched implication grew more naturally out of considering the nature of resources.

It is probably fair to say that the (negative) claims about the resource sensitivity of linear logic provoked the most controversy in the meeting. While there was perhaps a fairly general consensus that linear logic should not be seen as the only logic of resources, there was no similarly general agreement that it was not a resource logic at all, nor that it was thereby devoid of interest, e.g. in shedding light on proof theory / cut-elimination. Also, to many of the participants working at the applied end, much of the dispute had the air of an internal argument, where it was unclear whether or not the results would have far-reaching consequences for many of the more practical applications.

At a more informal level, the meeting succeeded admirably in bringing together people from a variety of different backgrounds, and with different expectations of linear logic, and provoking lively, friendly and productive discussion.

21 Multimedia Database Support for Digital Libraries

Seminar No. **99351** Report No. **249** Date **29.08.–03.09.1999**
Organizers: Elisa Bertino, Andreas Heuer, Tamer Ozsu, Gunter Saake

Digital libraries are a key technology of the coming years allowing the effective use of the Internet for research and personal information. National initiatives for digital libraries have been started in several countries, including the DLI initiative in USA http://www-sal.cs.uiuc.edu/~sharad/cs491/dli.html, Global Info http://www.global-info.org/index.html.en in Germany, and comparable activities in Japan and other European countries.

A digital library allows the access to huge amounts of documents, where documents themselves have a considerably large size. This requires the use of advanced database technology for building a digital library. Besides text documents, a digital library contains multimedia documents of several kinds, for example audio documents, video sequences, digital maps and animations. All these document classes may have specific retrieval methods, storage requirements and Quality of Service parameters for using them in a digital library.

The topic of the seminar is the support of such multimedia digital libraries by database technology. This support includes object database technology for managing document structure, imprecise query technologies for example based on fuzzy logic, integration of information retrieval in database management, object-relational databases with multimedia extensions, meta data management, and distributed storage. The seminar is intended to bring together researchers from different areas like object and multimedia databases, information retrieval, distributed systems, and digital libraries. It is the intention of this seminar to clarify differences in terminology between these areas, to analyze the state of the art, discuss requirements of digital libraries for multimedia databases and to identify future trends in research and development. The seminar should therefore focus on two major questions:

- Which functions of digital libraries need database support?
- What can database techniques offer to support these digital library functions?

These major questions can be detailed to specific topics, which list the technological areas relevant for this seminar (this list is of course not exhaustive):

- How to support digital libraries?
  - Document Servers
  - Supporting Different Types of Documents in Database Systems
  - Document Acquisition and Interchange
  - Extending Object-Relational and Object-Oriented Database Technology for Digital Libraries
  - Storing, Indexing, and Querying Large Sets of Documents
  - Integration of Heterogeneous Meta data and Documents
  - Combining Querying on Structured Meta data and Content-based Retrieval
  - Integrating Vague and Fuzzy Queries
  - Distribution of Queries
  - Different User Views on Large Sets of Documents
  - Visual Interfaces to Digital Libraries
- Which features of digital libraries need database support?
  - Alerting Services
  - Intelligent User Agents, Personal Digital Libraries
  - High Performance Document Servers
  - Efficient Retrieval Functionality
  - Document Delivery and Data Dissemination
  - Security and User Access Models
  - Trusted Document Servers

22 Social Thinking – Software Practice

Seminar No. **99361** Report No. **250** Date **05.09.–10.09.1999**
Organizers: Yvonne Dittrich, Christiane Floyd, Nimal Jayaratna, Finn Kensing, Ralf Klischewski

Approaches Relating Software Development, Work, and Organizational Change

Scope of the Seminar

During the last decade, the embedding of software into work practice has stimulated interdisciplinary cooperation between social scientists and computer scientists in areas such as computer supported cooperative work (CSCW) or human computer interaction (HCI). The discussion is largely driven by experiences gained in development projects. Key questions include: How can the perspectives and conceptualizations of different users and other stakeholders be taken into account? What is the relation between, on one side, reflective and analytic abstractions of the social sciences and, on the other side, generative abstractions developed by the designers in order to be implemented in and with a computer application? A coherent understanding of these and related questions is needed in order to provide suitable methods for software development.

Background: Social Thinking ...

Approaches developed in the social sciences for understanding human learning and communication, individual and cooperative work, and the interrelation of technology with organizations, provide a starting point for dealing with the problems at stake here. Although these approaches have been developed with no specific concern for computing, several of them have been tailored to the needs of software development and use:

- *Activity theory* and developmental work research focus on the activities of individuals, portraying these activities as mediated by tools and taking place in a social context. For understanding and changing activities they rely on representations of complex activity networks.

- *Ethnographic* workplace studies are concerned with how individual or collaborative work is actually being performed, in particular, they show the use of artifacts in work practice. Through participant observation, open interviews and other techniques they aim to understand work practice from the participants point of view.

- *Discursive approaches* start out from the different perspectives of the actors involved, including their interests, and power relations. By facilitating communication, they aim to provide equal opportunities for participation.

- *Systemic approaches* emphasize the interconnections between actors in organizations and between organizations and technology. They focus on different levels of reality construction inherent in human learning and communication, individual and cooperative work.

These approaches emerge from different, to some extent controversial, discourses in the social sciences. So far, their applicability to software development and use has largely been discussed in separate arenas.

... and Software Practice

While the discipline of software engineering is mainly concerned with the formal principles, the technical basis and the methodological support for software development, the reflection of software practice as a human activity needs to go beyond an engineering framework. Several aspects come into focus:

- Software development is a continuous process *based on human learning and communication*, in which all activities contribute to insights into the desired functionality and use of the software to be constructed.

- *Methods* and their underlying concepts are social vehicles for technical work. They embody a perspective on software development, concerning, for example, who is to be involved, which activities need to be supported and how, and what aims should be achieved.

- Key activities of software development, such as requirements analysis, modeling and design, *relate the technical domain* of software

functions and modes of human-computer-interaction *to the social world* of work and organizational change.

- Software development incorporates *organizational concepts*. The use of software products constrains collaborative work and organizational development. Thus, reifications in software systems have an impact on the potentials of organizational change.

In order to take these aspects into account, social science approaches are needed as guidance for software development in social contexts.

Seminar Aim

The seminar was arranged so as to promote a conversation about social science approaches in their relevance to software development. On one hand, approaches from the activity theoretic, ethnomethodological, discursive, and systemic schools were presented in their applicability to software development. On the other hand, mutual understanding was facilitated by identifying complementary views and methods as well as incompatibilities.

Seminar Program and Contribution of Participants

Participants had been invited to the seminar by Christiane Floyd, Nimal Jayaratna, Finn Kensing, and Lucy Suchman. Since it was intended to foster the interaction between different "schools of thought" a group of researchers from different backgrounds were entrusted with the preparation of the seminar. The members of this group were Yvonne Dittrich (Ronneby), Ralf Klischewski (Hamburg), Olav Berthelsen (Århus), Victor Kaptelinin (Umeå), Helena Karasti (Oulu), Jakob Nørbjerg (Copenhagen), Jesper Simonsen (Roskilde), Chris Westrup (Manchester), and Volker Wulf (Bonn).

After the opening on Sunday evening the course of the seminar followed the suggested 'themes of the day': "Making software", "Social thinking", "Software as Social Change", and "Industrializing Software Development". During the conference all participants played an active role, e.g. leading a working group, giving an introductory lecture or taking a part in discussion. Debate was enabled in plenary sessions as well as in small working groups from which the results of the discussions were

presented orally and/or on wall paper. Friday morning was reserved for summing up and discussing further projects of co-operation.

All participants were expected to submit a position paper related to the seminar's theme. Prior to the seminar, these papers had been presented on a web site accessible only by participants. Authors had the chance to submit a revised version of their position paper to be included in the report. Following the seminar a book will be published, and all participants have been invited to submit a chapter based on their position paper.

23 Declarative Data Access on the Web

Seminar No. **99371** Report No. **251** Date **12.09.–17.09.1999**
Organizers: Nicolas Spyratos, K. Vidyasankar, Gottfried Vossen

Today, information is spreading to all sectors of society in ever increasing volumes. This information comes in multimedia digital form and is transmitted over world-wide networks. In particular, the World-Wide Web (WWW) renders it possible to obtain information that is distributed over the entire Internet. Since the Web (and the number of its users) continues to grow at a high speed, adequate tools are needed for finding, storing, and structuring the vast amount of information offered; for *locating, retrieving*, and *presenting* the information to the final user; for aiding the end-user in customizing the information obtained for personal usage. Although technology is advancing fast (e.g., Web browsers built into cellular phones), a lot remains to be done concerning the efficient retrieval of information from large digital collections (often called digital libraries), and its intelligible presentation to the end user.

From a database perspective, the information provided by the Web can be perceived as a huge, heterogeneous database which is distributed world-wide, and which is accessed by multiple users. From this point of view, it appears reasonable to try to adopt concepts and techniques from database technology and in particular from the area of *information retrieval* (IR) to the context of the Web. It makes sense to investigate to what extent they are applicable to a large-scale database such as the Web, or what kind of generalizations, extensions, or completely novel developments become necessary. Motivation to do so is obtained from a look at the present situation. Indeed, when accessing data sources on

the Web with current browser and search-engine technology, a number of issues arise which deserve further study; these include:

1. *Locating the source*: Today's search engines are "primitive" devices for performing searches simply because they rarely do content-based retrieval; this in particular applies to search engines such as `AltaVista`, `Fireball`, `Lycos`, `Yahoo!`, and others. Instead, they mostly rely on searching indexes or directories, where distinct strategies in handling index information are exploited.

   Generally, users of the Web feel the need to develop advanced tools for locating information, for example based on graph navigation techniques, on content-based retrieval as known in IR, or on classification techniques used in present-day data mining. Moreover, a specification of desired results in a declarative way, preferably in an SQL-related language as done in various research prototypes, could aid in locating sources that are appropriate for a given query.

   When retrieving information from a "digital library" such as the Web, the *precision problem* and the *recall problem* arise. Due to the uncertainty associated to information retrieval, the answer to a query usually contains non-relevant data (this is the precision problem), while relevant data may be ranked low or neglected altogether (this is the recall problem).

   In other words, an automated device may retrieve imprecise data, or may not be certain whether or not to include some data found in the answer set. Once again, techniques based on content-oriented retrieval could help.

2. *Retrieving relevant information*: Once a desired data source has been located, the issue of retrieving relevant data arises. Two factors are important in this respect: the *structure* of the data, i.e., data can be anywhere between highly structured and totally unstructured, and the multimedia *nature* of data.

   Both factors are vastly orthogonal, and require different forms of exploration, treatment, and presentation going way beyond what today's browsers like `Netscape`, `Internet Explorer`, or `HotJava` have to offer. Again, information retrieval (and possibly data mining) techniques could help, for example for searching a video database found at some Web server for all scenes in which Humphrey

Bogart kisses Lauren Becall. For specification of such search goals, we still envision an extension of SQL that allows to give possibly incomplete path and class descriptions.

Another important problem is providing *multi-modal retrieval*, i.e., allowing a single request to contain retrieval conditions expressed in different modes (visual, text, etc), and permit the user to select the most appropriate mode.

3. *Organizing retrieved data for personal usage*: Once data has been retrieved, it most likely needs to be reorganized for easy use in the applications the end-user has in mind. Often, data obtained from the Web is kept in a customized database for personal usage.

   The organization and maintenance of personal databases follows naturally as a topic from the other two. However, to the end we can imagine PC or workstation tools that help taking care of this so that these issues do not need fundamental research.

Even in the restricted and simple case that a user accesses a *single* data source through the Web, all these issues described above may arise. When multiple sources are accessed, the additional problem of *combining* and *integrating* information from these different sites comes up. As the information available through the Web becomes more and more complex and voluminous, the importance of providing adequate, application-oriented interfaces becomes a decisive factor. Indeed, users of cellular phones, office computers, or GPS-based navigation systems in cars, to name just a few, have vastly different requirements to their Web interfaces, ranging from simple textual to multimedia output. Considering the various environments from which information can be accessed (including mobile ones), the need to adapt interfaces to these environments arises. For example, an SQL query can hardly be expressed on the interface of a pager that provides one line of text only; on the other hand, a multimedia office computer can easily accommodate more sophisticated query facilities than SQL.

The goal of this seminar has been to study the problems of locating, retrieving, and presenting information on the Web. To this end, the seminar brought together researchers from the areas of database systems and data warehouses, information retrieval, multimedia presentations and interface design, as well as information integration, in order to discuss

demanding questions and open problems in detail; the issues discussed specifically included:

- query languages for locating and retrieving Web data;
- appropriate user interfaces;
- techniques for query processing;
- information retrieval approaches;
- resource management and information organization;
- webfarming and data mining, for example as applied in areas such as electronic commerce;
- techniques for analyzing the information found in Web servers;
- integration of information found on the Web for the purpose of creating personalized databases;
- emerging Web standards, in particular XML and its proposals for query languages.

24 Computational Cartography – Cartography meets Computational Geometry

Seminar No. **99381** Report No. **252** Date **19.09.–24.09.1999**
Organizers: Martien Molenaar, Marc van Kreveld, Frank Wagner, Robert Weibel

Cartography has a history of several thousand years. With some right it can be claimed to be one of the earliest branches of science. All the way through its history, cartography has also had an intimate relationship with geometry. In fact, the term "geometry" (which includes the Greek root "geos" for "Earth") reveals that the origins of geometry and cartography (which to a large part was and is devoted to the depiction of the Earth on maps) are in fact the same. Map making required the mastery of geometric principles in order to tackle cartographic problems such as map projection, positioning, and measurement.

Computational geometry is a relatively new branch of science. Yet, with the move from manual geometric construction to geometric computation several fields of science and engineering have an urgent need for efficient and robust geometric algorithms and data structures. Cartography is among these disciplines in need of sound algorithmic solutions to its geometric problems. Some problems that were highly cumbersome and complex to solve in manual cartography such as projection transformations or the construction of 3D or panoramic maps are easy to solve today by means of geometric algorithms. On the other hand, some problems which are the "bread and butter" of manual cartography, such as map generalization or the placement of symbols and labels on maps, are still largely withstanding an automated solution. And it is for these problems that interdisciplinary collaboration is necessary in order to advance the research frontier, with cartographers and specialists of geographic information systems (GIS) typically providing cartographic expertise, problem definitions, and a first cut at technical solutions, and with specialists of computational geometry providing the experience of algorithm crafting, thus leading to improved algorithms.

This Dagstuhl seminar was the second one on computational issues of digital cartography, and like the first one, it brought together a range of specialists from cartography, GIS, computational geometry, spatial databases, and spatial analysis, with a common interest in the application of computational geometry to problems of modern cartography and GIS. Two topics, map generalization and map label placement, showed a certain concentration of talks. In these areas, it was interesting to note that a number of speakers presented methodologies that integrate geometric algorithms with optimization techniques and evaluation or cost functions. Other topics included the analysis and visualization of digital terrain models; spatial analysis for exploratory interpretation of spatial phenomena; polygon overlay problems for massive data sets; pattern recognition for geometric and topological structuring of cartographic data; graph algorithms (graph drawing, network schematization, cross-country shortest path); and the integration of geometric data models in spatial DBMS.

As a special feature, eight representatives from R&D divisions of GIS and mapping software vendors were invited to this seminar. The objective was to expose academics and industrial representatives alike to each others' viewpoints, visions, and needs. The industrial representatives were also invited to give a demo of their system of thirty minutes and

share their viewpoints and perspectives with the academic participants in an industry panel session. In the opinion of both academic and industrial seminar participants, the involvement of industry representatives worked very well.

25 Finite Model Theory, Databases, and Computer Aided Verification

Seminar No. **99401** Report No. **253** Date **03.10.–08.10.1999**
Organizers: Georg Gottlob, Erich Grädel, Moshe Vardi, Victor Vianu

The goal of this workshop was to bring together researchers working in finite model theory (FMT), in databases (DB) and in computer-aided verification (CAV). Besides complexity theory, DB and CAV are the two main application areas of FMT in computer science.

A common concern of FMT, DB and CAV is the design and study of logical formalisms with the 'right' balance between expressiveness and complexity. In databases, query languages are developed that should be expressive enough for the relevant queries of a given application area, but nevertheless lend themselves to efficient strategies for query evaluation. In CAV, specification languages are sought that are able to express relevant fairness and liveness conditions, but can be efficiently checked on the important classes of transition systems. In FMT, one studies the relationship between logical definability and computational complexity systematically. One of the central open problem of FMT is the quest for logics that precisely capture the most important complexity classes, in particular the problem whether there is a logic for PTIME. Hence model checking problems, in the broad sense of finding algorithms for and studying the complexity of the evaluation of logical formulae (queries, specifications) in a structure (database, transition system), play a central role in all three fields.

Also the central logical formalisms in the three fields are of a very similar nature. Typically, a basic formalism like first-order logic, relational calculus or modal logic is extended by recursion in one form or another. In particular, fixed-point logics (formalisms that include least and/or greatest fixed points as their essential feature) play a central role in all three fields. In databases, fixed-point and while queries have been

studied quite intensively and fixed-point query languages such Datalog and its extensions are central to the field. In CAV, the mu-calculus is in some sense the quintessential specification language, since it subsumes most of the other common formalisms like PDL, CTL, CTL* etc. The discovery of natural symbolic evaluation of the mu-calculus has lead to the industrial acceptance of computer-aided verification. In FMT, the most important logics are the fixed-point logics LFP, IFP, PFP with a very close relationship to the most important complexity classes. Hence fixed point logics, their expressive power and the algorithmic problems connected with them have been a central topic of this workshop.

In all three communities the main focus has for many years been on finite structures (databases, transition systems). Interestingly all three communities have recently started to extend their methods to suitable classes of infinite structures. New applications like spatial (geographical) databases have lead to the study of infinite database models, notably constraint databases. In CAV, model checking problems on infinite transition systems such as context-free systems or push-down system have been successfully studied and are of increasing interest also for practical applications. Also the general approach and the techniques of FMT have been extended to suitable classes of infinite structures (e.g. metafinite structures or recursive structures), which seems to be one of the most promising perspectives of finite model theory for the future. In fact this new perspective has been partially motivated by the new developments in databases and CAV.

There were 43 participants at the seminar. The program consisted of five invited survey talks, namely

Martin Otto:	Finite Model Theory
Phokion Kolaitis:	Database Query Languages
Jan Van den Bussche:	Constraint Databases
Colin Stirling:	Games in Verification
Pierre Wolper:	Infinite Structures in Databases and Verification

and 25 other presentations, mostly of ongoing research. In addition we had a very lively evening session on *"Logic in Computer Science Education"* (chaired by Wolfgang Thomas) and numerous informal discussions in smaller groups.

We believe that this workshop has been a success. It has certainly helped to increase the awareness of the researchers working in one field

of the problems and methods in the others and thus to increase the interaction and collaboration of the three fields, and the transfer of methodologies from one field to another.

For additional information, see
http://www.dbai.tuwien.ac.at/user/dag99/

26 Temporal Logics for Distributed Systems – Paradigms and Algorithms

Seminar No. **99411** Report No. **254** Date **10.10.–15.10.1999**
Organizers: Edmund Clarke, Ursula Goltz, Peter Niebert, Wojciech Penczek

Distributed systems, i.e. systems characterised by the concurrent operation and interaction of several components, occur throughout information technology; from microprocessors to computer networks. Since the design and development process of distributed systems is very sensitive to errors, it is an accepted fact in both science and industry that formal approaches to specification and automatic verification and debugging are needed. An important formal framework in this line is the family of temporal logics.

Originally, temporal logics were directed to describe a sequentialised (interleaved) and global view of the behaviour of distributed systems. Several problems resulting from this approach have been identified. On the specification level, the global view of the system makes it difficult or impossible to intuitively specify behavioural aspects of selected parts of the whole system. On the algorithmic front, this sequentialised semantics leads to the well known state explosion problem, which is often the reason for automatic verification to fail and makes it neccessary to develop heuristic workarounds. These problems have led in the past to the following investigations:

- On the one hand, various semantic models capturing aspects of distribution and causality of the behaviour of distributed systems have been developed, in particular partial order models and event structures. On some of these models, several extensions of standard temporal logics with differing modes of expressiveness have been

defined. The logics have been investigated under several aspects: Axiomatisations, theoretic and pragmatic expressiveness, complexity of the satisfiability and satisfaction problems.

- On the other hand, research directed towards efficient model checking has focussed on heuristic improvements of model checking algorithms for interleaving logics, which are based on state space exploration. Techniques, which have been evolving in this domain include modular model checking, symbolic model checking (BDDs), partial order reductions, abstraction, and others. Many of these approaches heavily exploit the distributed structure of the system, but do not explicitly rely on a distributed logical framework.

While a lot of research in both directions has happened separately, the natural connection between them has not gone unnoticed: some logics tailored towards distribution allow new verification algorithms, and conversely the heuristics discovered in model checking algorithms influence the design of logics. However, dedicated research is needed in order to achieve practically useful results here.

The goal of this workshop was to bring people from these areas of research together. 35 participants from 12 countries accepted the invitation. They presented their current research in the field of interest in 28 presentations. The following main topics were addressed:

Partial order logics for linear time. Several talks presented the recent developments on the theory of these logics. Two talks addressed the logic LTrL, which is known to have the same expressive power as the first order logic. Albeit non-elementary complexity, automata constructions have been shown to be possible (and thus open a way to do model checking). Moreover, the theory has been cleaned up and new proofs as well as improved syntax have been presented. Two other talks addressed causality based logics, one giving a separation result and proof technique to distinguish some logics wrt. expressive power, the other introducing an elegant μ-calculus on traces, which is expressively complete. Finally, the theoretic foundations for generalizing trace semantics to systems with state dependent independency of actions were presented.

Partial order logics for branching semantics. Three talks introduced logics based on branching partial order semantics. Two talks

addressed an interesting ontology based on intuitions of partial knowledge in a distributed setting, and addressed questions of decidability and model checking. In the third talk, a Petri net oriented logic was proposed.

Proof techniques for distributed systems. In seven talks, most of the spectrum of methods to exploit structural knowledge about systems for proving properties were addressed: Unfolding based model checking for reachability and for linear time, partial order reduction techniques, theorem proving exploiting commutativity of actions, symbolic model checking with and without BDDs, and compositional model checking. The presentations were a fair mix between presentations of established results as well as recent developments.

Probabilistic and real time model checking. Three presentations addressed attempts to apply partial order reductions to timed systems, but with very different approaches, advantages and problems. Two presentations introduced model checking of discrete and continuous stochastic systems. On this line, up to now only symbolic model checking methods with generalisations of BDDs are known.

Verification of Message Passing Systems. Three talks addressed issues of automatic verification in message passing systems. In particular, verification and dedicated logics for message sequence charts were presented.

General aspects of temporal logics. The five presentations in this section are a mix of complexity considerations in search of both simple (yet useful) and very powerful temporal logics, which are of course of importance to the distributed case also.

In addition to the talks, two sessions were used to discuss open problems and controversial issues. Particularly interesting and lively were discussions on

- unfolding methods vs. partial order reductions for model checking,
- usability and pragmatics of logics for practical specification.

Summarizing, the seminar was intense and stimulating. Inspite of the full "official" programme, concentrated work in smaller groups continued into the evenings, using the excellent facilities and working atmosphere of Dagstuhl.

27 Efficient Language Processing with High-level Grammar Formalisms

Seminar No. **99421** Report No. **255** Date **17.10.–22.10.1999**
Organizers: H. Uszkoreit, J.-I. Tsujii

Motivation text:

The topic of the Dagstuhl-Seminar is human language processing with sophisticated models of grammar. During the last decade many researchers have abandoned linguistically sophisticated models of grammar such as HPSG and LFG in favor of shallow processing techniques. The turn was caused by the sobering insight that even after many years of system development the existing methods for deep grammatical processing with powerful grammar formalisms did still not meet the performance criteria posed by applied research. Neither efficiency nor robustness proved sufficient for realistic applications.

On the other hand, progress in shallow processing has demonstrated that many useful applications of language technology could be achieved without accurate deep processing. Some of the shallow methods also exhibit a great potential for the automatic acquisition of language models. Thus large fractions of the discipline arrived at the conclusion that linguistic grammar models are not suited for the efficient and robust processing of human language on the computer.

However, not all researchers followed this move. Most of the groups that continued research on and with declarative grammar formalisms were driven by linguistic motivations. Others maintained their belief in the prospects of the grammar formalisms because they expected that developments in hard and software technology, better performance models and progress in computational semantics would eventually overcome the existing problems.

At a small number of centers considerable efforts were invested in the search for better processing methods. New results from several areas of computer science were exploited. Methods from constraint-logic programming, compilation technology, probabilistic language processing and many other sources were investigated.

Some of the efforts concentrated on the combination of many small improvements, others focussed on the search for radically different processing models.

The diversity of investigated approaches and claims of noticeable progress in efficiency deserve a new assessment of the state of the art. Some questions need to be asked:

- Have there been any real breakthroughs, can they be expected in the near future or does all progress in the area consist of a sequence of numerous small steps?
- Are there any processing systems based on sophisticated grammar models that already meet the demands of realistic applications?
- Have the various attempts to combine statistic and linguistic methods started to bear fruit?
- Do we have methods for grammar acquisition that can replace or complement intellectual grammar engineering?
- Do the employed grammar models pose similar problems for efficient and robust processing or do they differ in interesting ways with respect to their potential for computational processing?
- Is their worst case complexity directly related to the efficiency problems encountered in existing systems?
- Do we have a more sophisticated view today on the sources of real performance problems than we had ten years ago?
- Will both methods for grammatical description and grammatical processing have to change drastically before deep processing can be the basis of useful applications?
- If so, which recent approaches and results from computer science, psycholinguistics or theoretical linguistics can be expected to feed into this development.

The seminar will bring together experienced researchers from many parts of the world. Several grammar models and the major approaches in performance modeling will be represented. The meeting shall serve as a forum for presenting results, exchanging ideas and opinions, discussing new approaches, sharing experiences and assessing the state of the art. In the selection of presentations, priority will be given to reports of new results that are supported by performance measurements. Facilities for

demonstrating systems will be provided. In addition there will be a small number of topical talks summarizing relevant developments in broader research areas.

28 Scheduling in Computer and Manufacturing Systems

Seminar No. **99431** Report No. **256** Date **24.10.–29.10.1999**
Organizers: Jacek Blazewicz, Ed Coffman, Klaus Ecker, Gerd Finke

The objective of the seminar was to provide a forum for the discussion of current and ongoing research in scheduling. The seminar promoted an exchange of ideas covering the entire spectrum from case studies of real applications to recent advances in mathematical foundations. These various aspects of the scheduling area have been covered by 38 lectures which addressed classical application areas such as distributed processing, operating systems, dependable systems, flexible manufacturing, and others. It is worth pointing out that many lectures have been motivated by practical considerations, as for example machine break downs, batch scheduling, synchronous production, robotic cell scheduling, real-time scheduling, resource investment problem and others. But also exciting new areas have emerged such as those in modern communications, examples being wireless networks, multimedia networks, and the internet.
The seminar proceeded along three broad fronts:

- *Applications*, which include empirical studies of existing systems as well as numerical studies of the analysis and simulation of system models. Most of the studied applications came from the area of production scheduling and planning, such as just in time scheduling, due date assignment and project control, including special problems dealing with machine breakdowns, robotic cells, assembly scheduling, load balancing, minimizing the number of workers (human resources). Other presentations considered special problems from chemistry and oceanography, the design of schedulers e.g. for web applications, and planning examination sessions.

- *Algorithms* for various problems such as batch scheduling, resource scheduling, tardiness problems, shop problems, deadline and due

date scheduling, real-time scheduling, on-line scheduling, single machine problems, time lags, scheduling with communication delays, and/or scheduling. The main concern in these presentations was the design and analysis of algorithms ranging from simple and tractable on-line and greedy rules to methods based on semi-enumerative approaches, branch and bound, local neighborhood search, and LP formulations.

- *Theory*, which includes recent results in the analysis of new and classical problems under novel (or multiple) criteria, dealing with particular assumptions on machines, tasks (e.g., release dates, precedence constraints, communication delays, multiprocessor tasks, biprocessor tasks), and other problems such as assembly scheduling problems and on-line scheduling. Typical questions discussed were the structure of problems and their relation to graph theory, complexity of problems including polynomial solvability, the design of algorithms and performance analysis, and the approximability of optimal solutions.

29 Complexity of Boolean Functions

Seminar No. **99441** Report No. **257** Date **31.10.–05.11.1999**
Organizers: David Mix Barrington, Rüdiger Reischuk, Ingo Wegener

The complexity of Boolean functions is one of the central and classical topics in the theory of computation. Despite of some breakthrough results (e. g., exponential lower bounds on the monotone circuit complexity, bounded depth unbounded fan-in circuits, and linear depth branching programs, or the classification of bounded-width polynomial-size branching programs by $\mathbf{NC}^1$) there still seems to be a long way to go before successfully establishing large lower bounds in the case of unrestricted circuits over complete bases. Besides the classical lower bound and classification problems people active in this area are working on related topics like communication complexity, neural nets, quantum computing, and learning.

The organizers are happy that 37 researchers followed their invitation to Dagstuhl, they came from Germany (17 including guests from Poland

and Lithuania), USA (5), Canada (4), Japan (4), Austria (2), Czech Republic (2), England, Netherlands, and Russia.

The 27 talks captured many of the aspects of Boolean function complexity. There were several talks on branching programs (also for variants with applications in CAD and verification), circuits, communication complexity, and learning. Further talks focussed on algebraic methods. Aspects like randomization and nondeterminism were considered as well as quantum computing and cryptography. Besides some classical automata problems also related topics on algorithms and data structures were discussed. The schedule contained an open problem session and an evening discussion on new models motivated from biocomputing.

30 Rigorous Analysis and Design for Software Intensive Systems

Seminar No. **99451** Report No. **258** Date **07.11–12.11.1999**
Organizers: Stephan Jähnichen, Michel Lemoine, Tom Maibaum, Martin Wirsing

The seminar was concerned with a challenging problem in current software technology: the use of non-sequential components in heterogeneous systems. Both topics are related and raise many interesting issues, such as concurrency, distribution, reliability, etc. They challenge existing formalisms and methods and were addressed at the workshop by various speakers. Heterogeneity of systems (e.g., hardware vs. software, continuous vs. discrete, etc.) is asking for the assumption that software can be considered in isolation. The methods used for sequential component development are being extended in an attempt to cope with these new requirements. At present, it is not clear whether these methods are in fact extendable. New methods and formalisms are being invented to address the challenges of building such systems.

To tackle the task of rigorous analysis of large systems, the methods will focus on high level specifications. That is, complex heterogeneous systems and the constituent components are described more abstractly, say on the level of system architecture rather than on the level of mere programs. A system architecture reflects interaction and interfaces between the components without specifying all their complex internal func-

tionality. Analysis of such an architecture is a new challenge for methods being applied to ordinary software systems so far.

When discussing about systems in the large, we are also faced with refinement issues. Detailed information about timing or any physical limitation is not known on the abstract level of specification. For supporting the incremental development new strategies for refinement are introduced, i.e. how to develop a system design straightforwardly from a high level specification.

In practice, semi formal methods like UML are accepted by a broad audience of software engineers in order to describe heterogeneous systems on a high level. Although UML models are primarily used to communicate only a design, the emerging question is how formal notations and languages, which are developed for rigorous analysis already, can support the design phase. A formalization that bridges the gap between semi formal and formal notations is to be developed and investigated.

In order to make technologies available and useful, adequate tool support has to be provided for actual usage in real applications. We aim at environments in which tools and notations are adequately integrated and which support methodological guidance without constraining the user's creativity and individual progress.

In addition to the topics dealt with by the speakers, the workshop participants formed three working groups to discuss further questions of interest:

In order to get a comparison in the results of the different formal approaches, the community should benefit from treating a particular case study with the different formalisms. A new case study reflecting the needs of software intensive systems has to be found. In one working group a list of criteria to be characteristic for a suitable case study was worked out. According to these criteria the group agreed upon a rapid transportation management system (including a train control system, data management, etc.) to be a suitable case study.

In a second working group possible integrations of UML with formal methods were discussed. Although UML is known as an uprising formalism it lacks a formal foundation as well as tool support through formal treatment. Several suggestions how to relate the notations of UML with formal notations were discussed.

A standardization of formal methods based on a formal methods web repository was discussed in the third working group. The repository to be

set up should collect all current formalisms as well as their corresponding software environments (if available) and case studies treated so far. This is to give a survey to industry or other potential users.

As a result of these discussions and the subsequent presentations, it was decided to apply for another Dagstuhl seminar, with a similar orientation but calling additionally for the presentation of techniques and methods in the framework of a common case study to be distributed with the call for participation. A suggestion for this case study was the development of a train-control system (the result of one of the working groups).

31 Computability and Complexity in Analysis

Seminar No. **99461** Report No. **259** Date **14.11.–19.11.1999**
Organizers: Ker-I Ko, Anil Nerode, Klaus Weihrauch

All over the world numerous computers are used for real number computation. They evaluate real functions, find zeroes of functions, determine eigenvalues and integrals and solve differential equations, and so they perform or at least are expected to perform computations on sets like the set of real numbers, the set of open subsets of real numbers or the set of differentiable real functions. The increasing demand for reliable as well as fast software in scientific computation and engineering requires a sound and broad foundation. *Computable analysis* is the mathematical theory of those functions on the real numbers and other sets from analysis, which can be computed by machines. It connects the two classical disciplines analysis/numerical analysis and computability/complexity theory combining in particular the central concepts of limit and approximation on the one hand and of machine models and computation on the other hand. Computable analysis may serve as an additional framework for numerical analysis and all other disciplines which need an exact concept of computation for real functions.

Though computable analysis started in the early years of computability theory, the field is still in its infancy. It has a great potential for further development, since there are numerous challenging open problems, many basic questions have not yet been studied systematically and only

occasionally its concepts have been applied to advanced problems.

32 Symbolic-Algebraic Methods and Verification Methods – Theory and Applications

Seminar No. **99471** Report No. **260** Date **21.11.–26.11.1999**
Organizers: Götz Alefeld, Jiri Rohn, Siegfried M. Rump, Tetsuro Yamamoto

The second Dagstuhl seminar on *Symbolic-Algebraic Methods and Verification Methods - Theory and Applications* brought together 39 participants from 9 countries, with 10 participants coming from overseas. The seminar continues a first one held in 1992 in Dagstuhl.

The 35 talks covered a wide range of topics of the three areas Computer Algebra, Verification Methods and Real Number Theory. The aim of the seminar was to bring together experts of those different areas to discuss common interests.

All three areas aim on computing correct results on the computer. Here correct is to be understand in a mathematical sense including all model, discretization and rounding errors. The methods may also synergize and use good numerical approximations as a basis for subsequent computation of error bounds.

In the talks we saw some algorithms with result verification for finite dimensional as well as infinite dimensional problems, solutions to classical problems in Computer Algebra and a number of efforts to combine different methods and areas. Such methods mutually benefit from each other and are very promising. Moreover, we saw a number of practical applications.

33 Content-Based Image and Video Retrieval

Seminar No. **99491** Report No. **261** Date **05.12.–10.12.1999**
Organizers: Hans Burkhardt, Hans-Peter Kriegel, Remco Veltkamp

Images and video play a crucial role in Visual Information Systems and Multimedia. There is an extraordinary large number of applications

of such systems in entertainment, business, art, engineering, and science. Such applications often involve a client-server architecture, with large file and compute servers. Searching for images and video in large collections is becoming an important operation. Because of the size of such databases, efficiency is crucial.

We strongly believe that image and video retrieval need an integrated approach from fields such as image processing, shape processing, perception, data base indexing, visualization, querying, etc. On the other hand, most ongoing projects only deal with one or two of these aspects. A research emphasis is needed on incorporating multiple models for shape, color, texture, geometry, and syntax, so that the user does not have to specify low-level model parameters and combinations. This should lead to strategies of efficient indexing of visual information, in addition to techniques for combining visual with more traditional database information. Realistic evaluation criteria are needed, including test databases of realistic size in domains of interest, measures of similarity that allow variations in perceptual, semantic, and other criteria, and measures of accuracy and efficiency in assisting the user.

The purpose of this first Dagstuhl Seminar "Content-Based Image and Video Retrieval" was to bring together people from the various fields in order to promote information exchange and interaction among researchers who are interested in various aspects of accessing the content of image and video data, including topics such as:

- Indexing schemes
- Matching algorithms
- Visual data modeling
- Retrieval system architectures
- Image and video databases
- Feature recognition
- Video segmentation
- Picture representation
- Query processing

- Perception issues
- Video and image compression
- Visualizing pictorial information
- Searching the web
- Delivery of visual information
- Benchmarking
- Application areas of image and video retrieval

For this seminar, we have invited internationally known as well as young researchers from various disciplines with a common interest in content-based image and video retrieval. We have been together with a group of 28 researchers for a week, away from the rest of the world, and certainly good interaction and exchange of ideas took place during the sessions as well as in the very "gemtliche" wine cellar, enjoying the cheese platter.

There was a total of 24 presentations, two demonstration sessions, and two discussion sessions. One discussion session was about the challenges and problems to be solved, the other session was about the particular problem of how to assess the quality of retrieval systems and algorithms for subtasks.