Catriel Beeri, Andreas Heuer, Gunter Saake,
Susan Urban (editors):

Formal Aspects of Object Base Dynamics

Dagstuhl-Seminar-Report; 62
26.04.-30.04.93 (9317)

ISSN 0940-1121

Tel.: +49-6871 - 2458
Fax: +49-6871 - 5942

Das Internationale Begegnungs- und Forschungszentrum für Informatik (IBFI) ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Bezugsadresse: Geschäftsstelle Schloss Dagstuhl
Universität des Saarlandes
Postfach 1150
66041 Saarbrücken, Germany
Tel.: +49 -681 - 302 - 4396
Fax: +49 -681 - 302 - 4397
e-mail: office@dag.uni-sb.de

DAGSTUHL SEMINAR: 9317

FORMAL ASPECTS OF OBJECT BASE DYNAMICS

Organized by:

Catriel Beeri (University of Jerusalem, Israel)
Andreas Heuer (TU Clausthal-Zellerfeld, Germany)
Gunter Saake (TU Braunschweig, Germany)
Susan Urban (Arizona State University, USA)

April 26-30, 1993

SUMMARY

Object-oriented concepts have become important in helping to formulate next-generation database systems. The STRUCTURAL part of object-oriented database models has been formalized and is well understood. On the other hand, for the BEHAVIOURAL part of such models -- the dynamics of object bases -- there seems to be only a few approaches to a serious foundation. Nevertheless, this aspect of object-oriented systems is critical for defining the semantics of object-oriented database systems and their applications.

The behavioural part of an object base includes the concepts of database updates, transactions, and temporal integrity, as well as schema evolution, object migration, query processing, and active rule processing. This workshop brought together researchers working on foundations of programming languages, object-oriented systems, and databases to address the research directions for the formal aspects of object base dynamics. The international group, with representatives from Austria, Germany, Portugal, Estonia, Sweden, The Netherlands, Switzerland, Belgium, Italy, Israel, Canada, and the USA, met for five days in the peaceful setting of the Dagstuhl Castle to present current research projects on the dynamic aspects of object-oriented databases and to discuss research issues that need to be addressed.

The abstracts that follow provide summaries of the topics presented at the workshop. The topics are broadly categorized into five different areas. The first area is that of behavior specification. A large number of the abstracts fall into this category, addressing subtopics such as inheritance, the flow of object behaviour in information systems design, interaction between objects, the evolution of object-oriented databases, and the specification and use of constraints. This area included a group discussion on the problems associated with the inheritance of object behaviour.

The second area addresses the important issue of object updates. Update issues have often been ignored in object-oriented databases. The abstracts presented at the workshop represent some of the formal work that has been initiated in this area, addressing the semantics of updates, the declarative specification of updates, and problems associated with set-oriented updates.

The third topic presented at the workshop was that of transactions. The issues discussed in this area ranged from concurrent execution of transactions in object-oriented environments to more formal issues such as the declarative specification of state changes in transaction execution. The specification of distributed transactions was also discussed.

A fourth category for discussion was the area of active database systems. The research issues in this area focused on the integration of active rule processing with object-oriented databases, the automatic generation of rules for maintaining constraints, the integration of active, deductive, and object-oriented concepts, as well as the need for tools to support the analysis of rule behavior. The active database category was followed by a lively panel discussion on the future the active database paradigm.

The fifth and final topic addressed at the workshop was that of object-oriented queries. The subtopics in this category included issues related to the evaluation of path expressions as well as the evaluation of queries in logic programs involving negation. Issues related to the use of queries to support derived information in object bases were also addressed.

Although the papers have been organized into groups, many of the papers could actually fit into

several of the above categories. For example, some of the abstracts on updates and active databases are related to transaction issues as well as the specification of behavior. The overlap between these area characterizes the difficulty of research issues that must be addressed in object base dynamics. We hope that this workshop has helped to identify the important aspects of object base dynamics, providing motivation for continued research in this area.

Catriel Beeri
Andreas Heuer
Gunter Saake
Susan Urban

1 Specification of Behaviour

Some Thoughts on Inheritance of Object Behavior

Gerti Kappel (University of Vienna)
Michael Schrefl (University of Linz)

In this talk we describe desirable semantic properties of inheritance of object behavior specifications and discuss problems of mapping these specifications to typed object-oriented (database) languages. As underlying specification language we use OBD (Object/Behavior Diagrams [1]).

The specification of object behavior takes place at two interrelated levels of detail. At the object type level, the interdependencies of the activities of one object type are defined stating preconditions (in terms of prestates) and postconditions (in terms of poststates) of the activities. At the activity level, the signature of an activity is defined stating types and preconditions (= prestates) of input parameters, and the type and postcondition (= poststate) of the return value. In this realm, inheritance of object behavior is an important mechanism for structuring domain knowledge and for incremental behavior specification. Applying inheritance at the type level resolves to inheriting life cycle specifications and extending as well as specializing them, i.e., activities are added and pre- and postconditions are strengthened. Applying inheritance at the activity level resolves to inheriting signatures and specializing them, i.e., adding parameters, redefining types of parameters to subtypes, and strengthening pre- and postconditions by adding prestates and poststates. Unfortunately, these desirable semantic properties known as specialization (isa) inheritance following the covariance rule contradict the specification (subtype) inheritance following the contravariance rule. In the latter approach preconditions may only be weakened and types of input parameters may only be generalized to supertypes. Subtyping is necessary for approaching type safety and type substitutability, which are desirable properties of typed (database) languages. Providing covariance at the specification level and contravariance at the implementation level makes run-time (type) checks necessary. We discuss mechanisms to (partially) bridge this gap, such as distinguishing between abstract and concrete activities (the former have no implementation and thus can not be invoked on objects), and distinguishing abstract object types from concrete ones (the latter may only be leaf nodes in the inheritance graph).

[1] G. Kappel, M. Schrefl: Object/Behavior Diagrams. In: Proceedings of the 7th International Conference on Data Engineering, IEEE Computer Society Press, Kobe, Japan, April 1991, p.530-539.

Superposition of Behaviour

Georg Reichwein (INESC, Lisbon)
(joint work with Jose Fiadeiro)

Category theory is the mathematics for structure and modularity. Superposition is one of the main structuring principles for the modular design of parallel programs and distributed systems. We provide a categorial theory of superposition and place this theory in the context of the formalisation of object-oriented disciplines for decomposing and organising the development of reactive systems.

The presentation concentrates on the propositional fragment of this theory, i.e. object descriptions have only boolean control variables. We study the impact of various notions of superposition on the shape of the behaviour diagrams which graphically describe the finite automata representing the control part of object descriptions. The results are relevant for CASE tools which support graphical object specification languages like OBLOG.

Logic-Based Specification of Object Dynamics

Ralf Jungclaus (TU Braunschweig)

Dynamic Object Systems consist of objects that have a local state, that show local behavior and that interact. In the conceptual modeling phase, A Universe of Discourse is specified according to such a perspective. For the representation of evolving object systems, a mixture of operational (i.e. action/event-based) and declarative features should be available to allow for "natural" specifications.

In this talk we present a Temporal Object Logic that underlies a language for conceptual modeling of object systems. The syntax of the logic provides predicates that represent the enabledness of actions, the occurrence of actions, and the observability of attribute values in a state. The logic supports a strong notion of locality in that signatures are local to objects, predicates are localized by binding them to object identifier variables and only permitting local action to change the local state. The logic supports the specifications of temporal properties involving both actions and attributes.

We show how features of the conceptual modeling language TROLL^R describing object life cycles and interaction are translated into the logic to define their semantics.

TROLL *light* A Core Language for Specifying Objects

Martin Gogolla (TU Braunschweig)

TROLL *light* is a language for conceptual modeling of information systems. It is designed to describe the Universe of Discourse (UoD) as a system of concurrently existing and interacting objects. TROLL *light* objects have observable properties modeled by attributes, and the behavior of objects is described by events. Possible object observations may be restricted by constraints, whereas event occurrences may be restricted to specified life cycles. TROLL *light* objects are organized in an object hierarchy established by sub-objects relationships. Communication among objects is supported by event calling. Apart from introducing the various possibilities for the syntactical description of objects, we aim to describe how the state of an object community may be changed by event occurrences.

As a basis for our language we took the specification language TROLL. However, TROLL *light* is not just a subset of TROLL. Some details have been added or modified in order to round off TROLL *light*. This was necessary because we needed a clear and balanced semantic basis for our specification language. In particular we want to stress the fact that in TROLL *light* classes are understood as composite objects having the class extension as sub-objects. Therefore in contrast to TROLL an extra notion of class is not needed in TROLL *light*. This leads to a more orthogonal use of object descriptions. Over and above that concepts like class attributes, meta-classes, or heterogeneous classes are inherent in TROLL *light* while they had to be introduced in TROLL by additional language features. Second TROLL *light* incorporates a query calculus providing a general declarative query facility for object-oriented databases. For instance, terms of this calculus may be used in object specifications to describe derivation rules for attributes, or to query object communities in an ad hoc manner.

Our specification language is described in more detail in [1], the accompanying specification environment is outlined in [3], and the operational semantics of the language is presented in [2].

[1] S. Conrad, M. Gogolla, and R. Herzig. TROLL *light*: A Core Language for Specifying Objects. Informatik-Bericht 92--02, Technische Universitat Braunschweig, 1992.

[2] M. Gogolla, S. Conrad, and R. Herzig. Sketching Concepts and Computational Model of TROLL *light*. In A. Miola, editor, *Proc. 3rd Int. Conf. Design and Implementation of Symbolic Computation Systems (DISCO'93)*. Springer, LNCS Series, 1993.

[3] N. Vlachantonis, R. Herzig, M. Gogolla, G. Denker, S. Conrad, and H.-D. Ehrich. Towards Reliable Information Systems: The KORSO Approach. In C. Rolland, editor, *Proc. 5th Int. Conf. Advanced Information Systems Engineering (CAiSE'93)*. Springer, LNCS Series, 1993.

Specification of Coordinated Behaviour in the Software Development Process

Gregor Engels (University of Leiden)
Luuk Groenewegen (University of Leiden)

Software development is a complex, long-term process where a lot of different persons are involved to develop and maintain a lot of different documents. Nowadays software development environments which aim at the support of the software development process lack an explicit support of the process while offering a lot of isolated tools to handle documents. These so-called product-centered environments hava to evolve to process-centered environments in order to offer the desired and needed support for the software development process.

We present a specification language by which all static and dynamic perspectives of a software process model can be described in an integrated way. The language consists of several interrelated sublanguages. All of them are pure visual languages and are based on well-known diagrammatic languages like Entity-Relationship diagrams, data/object flow diagrams and state transition diagrams [1, 2].

The talk concentrates on the specification of coordinated behaviour of different human or computerized agents during the software development process. This sublanguage is based on the use of two-level state transition diagrams, which originally were developed in the PARADIGM approach [3].

The results presented in the talk are partly being developed within the ESPRIT BRWG PROMOTER, a basic research working group on software process modelling.

[1] G. Engels, L.P.J. Groenewegen: Specification of Coordinated Behaviour in the Software Development Process (Position Paper). In J.C. Derniame (ed.): Proc. 2nd European Workshop on Software Process Technology (EWSPT 92), Trondheim (Norway), Springer-Verlag, Berlin, LNCS 635, 58-60, 1992

[2] G. Engels, L.P.J. Groenewegen: Modular, Visual Specifications of Software Processes (Position Paper). To appear in Proc. 8th Intern. Software Process Workshop, SchloDagstuhl (Germany), March 1993

[3] Groenewegen, L.P.J., Morssink, P.J.: Modeling structural constraints as behaviour. In Informatiesystemen in beweging, Eds. P.W.G. Bots, H.G. Sol, I.G. Sprinkhuizen-Kuyper. Kluwer, Deventer, The Netherlands, 1991, 195-212

Modelling Evolution in Object-Oriented Databases

Hele-Mai Haav (Estonian Academy of Sciences)
(Joint work with M. Matskin)

In this talk we propose a methodology for specification of semantics of evolution in object-oriented databases. We define metalevel constraints on evolution of database schema, classes and

objects in terms of Horn logic. A part of metalevel constraints is generated automatically from the specification of classes and inheritance lattice or is obtained on the basis of object states. Another set of metalevel constraints can be defined as rules for modelling the behaviour of the data model concepts and these are considered as general constraints integrated to the metalevel. For example, implicit constraints on inheritance lattice or class reorganization rules can form this set of constraints. In addition, application oriented constraints can be defined by the user. We use Partial Deduction techniques besides the traditional deduction to guarantee the validity of managing evolution in OODB. Although using Partial Deduction is unusual in object- oriented databases, it allows to make deduction on incomplete specification of the semantics of evolution . This is the case when the object-oriented specifications are transformed to Horn clauses and not all possible predicates (facts) are generated at the compile time of classes. But the partial deduction method allows to do a part of deduction before knowing all facts. The result of partial deduction of a set of metalevel constraints can be considered as a collection of specialized conditions with respect to the particular problem of managing evolution in OODB. Validity of these conditions can be checked before performing real changes in OODB. Implementation principles of the methodology are discussed on the basis of the object-oriented language NUT [1]. Different types of transformations are defined for generation of predicates from the descriptions of classes and objects used in the NUT system. General idea of the methodology proposed is presented in [2].

[1] Tyugu E, Matskin M, Penjam J, Eomois P. NUT- An object-oriented language, Computers and Artificial Intelligence 1986, 6:521-542.

[2] Haav H-M. Specifying Semantics of Evolution in Object-Oriented Databases Using Partial Deduction, In. Modelling Database Dynamics (Eds. U. Lipeck and B. Thalheim), Springer-Verlag 1992, pp 48-63 .

Frame Assumptions and Operational Semantics of Object Behavior in CMSL

Roel Wieringa (Free University of Amsterdam)
Remco Feenstra (Free University of Amsterdam)
John-Jules (Free University of Amsterdam and University of Nijmegen)
Paul Spruit (Free University of Amsterdam)

CMSL (Conceptual Model Specification Language) is a language to specify object-oriented databases. Each transaction is viewed conceptually as a finite set of synchronously occurring events in the life of different objects. The effect of events is defined declaratively by means of dynamic logic axioms plus a frame assumption that says what remains unchanged by the event. One problem with this frame assumption is that it cannot serve as a basis for an operational semantics. Second, because the frame assumption says an event e has only a local effect and leaves the rest of the world unchanged, e cannot be combined with any other event that *does* change something in the rest of the world.

To solve these problems, we first define Dynamic Database Logic (DDL), which axiomatizes bulk insert/update/delete actions on logic databases. A sound and complete axiomatization exists

for propositional DDL and for first-order DDL without function symbols and with finitely many constants. In addition, there is a sound and complete operational semantics for these variants of DDL. We next show how CMSL update events can be implemented as DDL update programs that have an operational semantics as well as a compositional declarative semantics of transactions.

[1] P.A. Spruit, R.J. Wieringa, and J.-J.Ch. Meyer. Dynamic database logic: The first-order case. In U.W. Lipeck and B. Thalheim, editors, *Modelling Database Dynamics*, pages 103--120. Springer, 1993.

[2] P.A. Spruit, R.J. Wieringa, and J.-J.Ch. Meyer. Axiomatization, declarative semantics and operational semantics of passive and active updates in logic databases. *Journal of Logic and Computation*, To be published.

[3] R.J. Wieringa. A formalization of objects using equational dynamic logic. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *2nd International Conference on Deductive and Object-Oriented Databases*, pages 431--452. Springer, 1991. Lecture Notes in Computer Science 566.

Petri Net Based Modelling of Procedures in Complex Object Database Applications

Andreas Oberweis (Universitaet Karlsruhe)
(Joint work with Peter Sander, Universitaet Karlsruhe)

A new type of high level Petri nets is introduced for modelling procedures in complex object database applications. Each net defines a class of possible system procedures, i.e. sequences of (possibly concurrent) operations, in a complex object database application. Places (predicates) in these so-called nested relation/transition nets (NR/T-nets) represent schemes of unnormalized relations ("nested relations"). The marking of each place is a (possibly empty) nested relation of the respective type. Each transition represents a specific type of operation on the relations in the adjacent places. These operations may not only operate on whole tuples of a given relation but also on "subtuples" of existing tuples. Arcs in a net are inscribed with so- called filter tables which allow (together with an optional logical expression as transition inscription) to formulate conditions on the specified (sub-) tuples.

CLOOD: A Class-less Models for Object--Oriented Design Databases

Margret Gro-Hardt
Gottfried Vossen
(Justus-Liebig-Universitat Giessen)

Engineering design applications have specific requirements to database support which are not yet met completely by existing systems. We argue that one reason is the inability of current

object models to capture the needs of design applications appropriately. First, object-oriented data models typically center around the notion of class; however, applications like CAD or CASE frequently center around objects with types varying during the design process (which has a creative or experimental flavor) and therefore have little use for defining classes in the first place. In a design environment a class lattice is seldomly defined first and then populated with objects, but objects exist prior to their classes. The result of a design process is, for instance, a prototype representation for a certain car, and as the word "prototype" already indicates, it is a singleton with respect to the representation of other objects.

Second, design applications need a versioning mechanism, which is typically not provided within the data model, but as an additional construct whose usage then creates unnecessary overhead.

We show how to combine concepts from prototype programming languages with a versioning mechanism for objects into a new model which does without the notion of a class. The idea is that objects can freely exist in a database, have multiple versions, and can share structure and behavior with others. This provides much more flexibility in specifying structure and behavior of a design object (e. g., there are no problems with migration of objects from one class to another). Nevertheless, a grouping mechanism for objects is provided, so--called collections, which serve as addressees for declarative queries. Collections contain arbitrary objects (and/or versions), though (in general) they do not partition the whole objectspace. In addition, working with design objects is done both associatively (via user-defined identifiers) and declaratively (via collections as a query tool).

Principles of Object Oriented Database Design

Klaus--Dieter Schewe
Bernhard Thalheim
(Cottbus Technical University)

The design of complex information systems requires a transparent model-based methodology. It has been claimed that object orientation will have a significant impact on the development of such a methodology, especially as reusability and naturality of conceptual modelling are concerned.

The methodology presented in this paper concentrates on four significant principles of object oriented database (OODB) design. The basic constituent is *stepwise refinement*, i.e. to begin the design process with a partial model that is completed and concretized furtheron depending on the growth of application knowledge. *Class abstraction*, i.e. to support libraries of incomplete parameterized designs that are instantiated and specialized later, is a natural consequence hereof. *Declarativity* is achieved by constraint centered design with (up to some degree) automatic transformation into consistent transactions. *Variations* enable the design of information systems with heavy reuse of existing design components.

The methodology is based on a theoretically founded object oriented datamodel (OODM). Hence the support of inferences such as deciding the identifiability of objects, detecting the

relation of an intended design to components in existing design libraries, and checking operations for reducedness as a prerequisite for the automatic transformation of constraints into consistent transactions.

Inheriting and Using Integrity Constraints or Not?

Bernhard Thalheim
(Cottbus University)

The talk gives an overview on the theory of integrity constraints in different database models. First, the history of the theory of integrity constraints is surveyed briefly. The main results obtained mainly in the context of the relational model are discussed. This development led to a very rich theory of integrity and to an overwhelming set of different classes. The book [1] surveys only 95 different classes of static integrity constraint. During the last decade this theory has been extended to different (semantical) database models. There are some papers with extensions to object-oriented models. However, it is often claimed that this theory has been useless. During the talk several arguments (representation in constructors, combinatorial, human abilities, systems use, bad properties) against the generalization of this theory to other database models are discussed. The main part of the talk is used for the development of a general theory of integrity constraints. It is shown how different classes of integrity constraints can be generalized in accordance to the constructors used in the new model. The basis for this part of the talk is the approach used in [2]. The next part shows how integrity constraints can be used in other models and in database management systems. There are several areas where these constraints are necessary and useful. Finally, it is shown that integrity constraints can be inherited by any other database model.

[1] B. Thalheim, Dependencies in relational databases. Teubner, Leipzig, 1991.

[2] B. Thalheim, Fundamentals of entity-relationship modeling. Springer, Heidelberg, 1993

Aspects of Interaction for Abstract Objects

Gunter Saake
Thorsten Hartmann
(TU Braunschweig)

The object-oriented approach provides a natural view of evolving information systems as collections of objects having both structure as well as behaviour. This view is supported by specification languages like TROLL [1], Oblog or Troll light. The basic communication primitive in these languages is *event calling*. Event calling can be characterized as synchronous asymmetric communication. A formal object model based on life cycles consisting of event snapshots (sets of synchronized events) allows to model event calling inside (composite) objects [2]. Properties

of interest for snapshots resulting from event calling are consistency (no attribute modification conflicts) and finiteness. Both properties are in general undecidable but sufficient conditions for them can be analyzed based on syntactic properties of specification documents. Ordering relations based on control and data flow between events of a snapshot can be used to describe execution models extending the notion of parallel execution of events [3], respectively to derive execution plans for snapshots.

[1] Jungclaus, R. and Saake, G. and Hartmann, T. and Sernadas, C.: Object-Oriented Specification of Information Systems: The TROLL Language. TU Braunschweig, Technical Report 91-04, 1991.

[2] Hartmann, T. and Jungclaus, R. and Saake, G.: Aggregation in a Behavior Oriented Object Model. In: Proc. European Conference on Object-Oriented Programming (ECOOP'92), Lehrmann Madsen, O. (ed.), Springer, Berlin, LNCS 615, 1992, pages 57--77.

[3] Hartmann, T. and Jungclaus, R. and Saake, G.: Animation Support for a Conceptual Modelling Language. In: Proc. Int. Conf. on Database and Expert Systems Applications (DEXA 93), Springer, Berlin, LNCS series. To appear.

Dynamic Modelling in Database Design

Moira C. Norrie
(ETH, Zurich)

Dynamic modelling in terms of entity life cycles can be used as a basis for the determination of classification structures in object model schemas. The talk presented some preliminary work in this area based on the use of two forms of entity state transition diagrams (ESTD). The w work follows on from previous work on interactive systems for the support of object-oriented database design which was done in the context of the Esprit project Comandos. The main features of the Comandos data model, BROOM, and accompanying proposals for mechanisms to support and control object evolution were also presented.

Panel Discussion on the Specification of Behaviour: Subtyping versus Specialization

Gerti Kappel
(University of Vienna)

Subtyping or specification inheritance refers to inheriting attributes and methods of a supertype T to a subtype T' such that objects of T' may be used whenever objects of T are expected and no run-time type error occurs (type-safe type substitutability). This kind of inheritance is based on the contravariance rule. Specialization or isa-inheritance refers to inheriting attributes and methods of a supertype T to a subtype T' such that the objects of T' constitute a subset of the objects of T and

T' is a more specialized specification than T (type subsumption). This kind of inheritance is based on the covariance rule. Whereas the latter is a desirable feature of object-oriented modeling and design, the former is a desirable feature of type-safe system development.

The lifely discussion, hence, the dilemma of these conflicting goals is best summarized by the following example (brought up by Andreas Heuer). Assume object type BOOK with attributes and corresponding domains Title:STRING, Price:REAL and method price_Calc(pub:PUBLISHER) (the price of the book depends also on the publisher's tax). Assume further subtype AUSTRIAN_BOOK with inherited (and overridden) attributes and method Title:STRING, Price:REAL, and price_Calc(pub:AUSTRIAN_PUBLISHER). The variables b and p have the static types BOOK and PUBLISHER, respectively. The statement b->price_Calc(p) should be dynamically bound to the appropriate implementation depending on the dynamic types of b and p. Find solutions in your favorite language which are type-safe!

2 Updates

Determinism of Bulk Application of Update Methods

Marc Andries (University of Leiden)
Jan Paredaens (University of Antwerp, UIA)
Jan Van den Bussche (University of Antwerp, UIA)

Assume M is an update method, which when called on a receiver object and possibly a number of parameters, transforms the database. We discuss the issue of calling such a method not to one receiver-parameter tuple, but to a set of them. We consider two possible intended semantics for such "Bulk Applications": a parallel one and a sequential one. These two intended semantics are compared. A fundamental difference between them is that the sequential one may be non-deterministic.

We study ways to test for determinism. Two major possible directions toward this problem are mentioned. The first remains in the general framework but provides additional semantics of the method in terms of "scheme colorings", which, for example, describe which types of objects and properties in the database are deleted, created, or used. This direction brings up the issue of axiomatizing the behavior of an arbitrary database transformation, in analogy with type theory.

The second direction assumes that the methods are programmed in a declarative language like the relational algebra. This leads to the novel application of known relational database theory techniques, like testing for containment of positive relational algebra expressions, as well as the extension of these techniques to, say, object creation. Open problems of this kind have already been posed in the literature [1,2,3].

[1] A. Chandra. Programming primitives for database languages. In Proceedings POPL'81.

[2] R. Hull and Y. Su. Accessing object-oriented databases. In Proceedings SIGMOD'89.

[3] R. Hull and M. Yoshikawa. Equivalence of database restructurings involving object identifiers. In Proceedings PODS'91.

Specification and Evaluation of Database Updates

Weidong Chen
Jinghong Zeng
(Southern Methodist University)

We present a novel approach to specification of database updates. Declarative update languages are derived by extending existing query languages with atomic formulas for a new database after an update. Evaluation of database updates reduces to abductive reasoning in which atomic formulas about a new database are abducible.

An extension of logic programming with updates is described as a case study. Recursive

and set-based updates are naturally specified. Current query processing techniques can be readily applied for efficient update evaluation.

Deterministic Semantics of Set-Oriented Update Sequences

Marc H. Scholl
(University of Ulm)

Updates and set-orientation can not be combined as easily as retrievals can. The reason for this is that the state transitions performed by each update step might interact with expressions that are part of the update specifications, leading to non-deterministic behavior due to order dependencies. The problem shows up even in simple SQL updates with so-called self-referential update statements. It becomes even worse, if one tries to combine set-orientation and sequences of elementary updates (or other control flow mechanisms), as necessary in method languages for ODBMSs.

A generic iterator ("apply_to_all") is proposed that allows to apply sequences of update operations in a set-oriented way with deterministic semantics. Because the mechanism is independent of a particular model, it can be used in the relational and in object-oriented ones. Thus, the deterministic semantics of embedded SQL cursors, and of triggers that are applied after (set-oriented) SQL updates can be checked. Furthermore, the iterator can be used to apply object-oriented methods, which are typically sequences (or other more complex control structures) of elementary update steps defined on a single object, also to sets of objects in a deterministic way.

The criteria that are used to detect whether or not a sequence of updates can be applied to a set are essentially the same that can be used for semantic concurrency control. What we need is a matrix of the conflicts between the elementary update and retrieval operations provided by the data model. This is the only model dependency of our approach, all the rest is completely generic.

We can exploit the relationship with concurrency control even further, by using a nested transaction model that immediately shows the potential for fully parallel (intra-transaction parallelism) implementation of set-oriented updates.

[1] Christian Laasch and Marc H. Scholl: Deterministic Semantics of Set-Oriented Update Sequences, Proc. 9th Int'l Conf. on Data Engineering (ICDE), Vienna, 1993, pp. 4-13.

Updates and Rules

Georg Lausen
(Universitat Mannheim)

The problem of performing updates in relational database systems is studied. A program to perform updates consists of a set of rules, where the intended updates are indicated by +, meaning insert, respectively, −, meaning delete in the head of the rules. The specific feature of the language presented is the automatic creation of states of the relations being updated (*frame rule*). In this

way an intuitiv control of the update process is achieved. The impact on the power and the declarativeness is discussed. In addition, a rewriting of update programs into standard Datalog is outlined.

Transactions and Updates in Deductive Databases

Danilo Montesi (Universita di Pisa)
Elisa Bertino (Universita di Genova)
Maurizio Martelli (Universita di Genova)

We present a new approach introduced in [1] to study the update problem and its integration in logic-based languages, with the aim of understanding how a logic language may express updates and how they can have a transactional behavior.

In particular, we first analyze the major drawbacks of the approaches in the literature and then we propose a new approach to base relation updates, called non-immediate update semantics, which has some nice properties. The basic idea is to consider a two steps computation. In the first step the tentative updates are collected and in the second step they are executed altogether. The updates resulting from the first step of the computation are not redundant and can be executed in parallel. Moreover, the first step can be computed top-down or bottom-up. In this way one reconciles forward and backward reasoning in logic-based language with updates. In addition expensive runtime roll back is not required. Then we analyze the Constraint Logic Programming (CLP) framework, and we show that it provides a formal setting for the non-immediate update semantics. However, to completely characterize our approach, CLP will be extended to allow modular program construction, resulting in the notion of open CLP program and its semantics. Indeed, considering a database as composed by the time varying extensional database and the time invariant intensional one, one needs a way to provide the semantics of the intensional database modulo the possible extensional ones. Updates to base relations are expressed as constraints in CLP scheme and a query triggering updates has a transactional behavior. Moreover, in order to express complex transactions, some control is introduced explicitly, outside the logic language, through sequence and iteration constructs.

The resulting language, called Extended Constrained Datalog has a four steps semantics. This semantics captures a rich observable property which induces two interesting equivalence notions for transactions and for databases. Those equivalence notions extend those for relational databases and deductive databases capturing the all-or-nothing behavior of complex transactions.

[1] D. Montesi, *A Model for Updates and Transactions in Deductive Databases*, Ph.D. Thesis, Dipartimento di Informatica, Universita di Pisa, March 1993.

3 Transactions

Distributed Transaction Specification

Rolf A. de By
Hennie J. Steenhagen
(University of Twente, The Netherlands)

The formal specification of a network of cooperating database systems is a tedious and error-prone task in which the designer(s) should be supported by tools based on formal theories.

We discuss the motivation and goals of a recently started research project in which we will try to combine the formal specification languages LOTOS and TM. The first language is an ISO standard for Open Systems Interconnection description based on process algebra. The second is a (sub)typed functional database specification language. Around both languages fairly complete tool sets have been developed in the past, or are currently being developed, and it is our aim to combine those tool sets in the coming years.

The goal of this effort is to provide a complete environment for distributed database design, verification, prototyping, simulation and testing. A theory of correctness-preserving transformations should lead to an evolutionary, and formally verifiable design trajectory towards correct implementations.

Semantics-Based Transaction Management in Object-oriented Database Systems

Gerhard Weikum (ETH Zurich)

The talk presented a model for reasoning about the correctness of concurrent transaction executions in object-oriented database systems, based on the notion of open nested transactions. Transactions are modeled as method invocation trees, and schedules are partially ordered forests. For each object type, a compatibility specification for its methods is assumed to be given, where two methods are considered compatible if their ordering in a schedule is insignificant from an application point of view (e.g., based on commutativity properties). A schedule is semantically serializable if it can be transformed into a serial schedule of the transaction roots by applying the following two rules: 1) the order of two compatible leaves that are ordered and ajacent in the schedule can be exchanged, and 2) "isolated" subtrees (i.e., subtrees that are ordered with respect to all nodes other than their own descendants) can be pruned. This model allows correctness reasoning about both inter- and intra-transaction parallelism, and it can deal with the coexistence of encapsulated-object hierarchies and directly accessible objects.

Towards a Unified Theory of Concurrency Control and Recovery

Hans-Jorg Schek
Gerhard Weikum
Haiyan Ye
(ETH Zurich)

The classical theory of transaction management is based on two differe nt and independent criteria for the correct execution of transactions. The first criterion, serializability, ensures correct execution of parallel transactions under the assumption that no failures occur. The second criterion, strictness, ensures correct recovery from failures.

In this paper we develop a unified model that allows reasoning about the correctness of concurrency control and recovery within the same framework. We introduce the correctness criteria of (prefix-)reducibility and (prefix-)expanded serializability and investi gate their relationships to the classical criteria. An important advantage of our model is that it captures schedules with semantically rich ADT actions in addition to classical read/write schedules.

Transaction Logic Programming (or, A Logic of Procedural and Declarative Knowledge)

Anthony Bonner (University of Toronto)
Micheal Kifer (SUNY at Stony Brook)

An extension of predicate logic, called Transaction Logic, is proposed, which accounts in a clean and declarative fashion for the phenomenon of state changes in logic programs and databases. Transaction Logic has a natural model theory and a sound and complete proof theory, but unlike many other logics, it allows users to program transactions. This is possible because, like classical logic, Transaction Logic has a "Horn" version which has a procedural as well as a declarative semantics. In addition, the semantics leads naturally to features whose amalgamation in a single logic has proved elusive in the past. These features include both hypothetical and committed updates, dynamic constraints on transaction execution, nondeterminism, and bulk updates. Finally, Transaction Logic holds promise as a logical model of hitherto non-logical phenomena, including so-called procedural knowledge in AI, and the behavior of object-oriented databases, especially methods with side effects. Apart from the applications to Databases and Logic Programming, we also discuss applications to a number of AI problems, such as planning, temporal specifications, and the frame problem.

A technical report is available by anonymous ftp to "csri.toronto.edu", in the file "csri-technical-reports/270/report.ps" It includes a complete development of the model theory, proof theory, and numerous applications.

4 Active Rule Processing

Reactive Object Oriented Databases: -or What Causes Events in ACOOD?

Mikael Berndtsson (University of Skovde, Sweden)

Reactive databases [1] have been proposed as an approach to support rules in database systems. Most proposals for reactive object oriented databases support event-condition-action rules, where method invocations are the triggering events. In recent, both rules and events have been proposed to be represented as first class obejcts.

ACOOD is a prototype reACtive Object Oriented Database, built on top of a commercial object oriented database. Both rules and events in ACOOD are treated as first class objects. This approach allows us to introduce a subscription mechanism, where rules can subscribe to events in order to reduce rule checking. Future research will focus on: i) dynamic event specification, ii) composite events, iii) event hierarchies, where events can subscribe to other events, and iv) the semantics of EC-CA rules.

[1] IEEE Bulletin of the TC on Data Engineering, Vol. 15, No. 1-4, Special Issue on Active Databases, December 1992.

Automatic Generation of Production Rules for Integrity Maintenance

Stefano Ceri
Piero Fraternali
Stefano Paraboschi
Letizia Tanca
(Politecnico di Milano)

We present an approach to integrity maintenance, consisting in automatically generating production rules for integrity enforcement. Constraints are expressed as particular formulas of Domain Relational Calculus; they are automatically translated into a set of compensating actions, encoded as production rules of an active database system. Production rules may be redundant (they enforce the same constraint in different ways) and conflicting (because repairing one constraint may cause the violation of another constraint). Thus, it is necessary to develop techniques for analyzing the properties of the set of active rules and for ensuring that any computation of production rules after any faulty transaction terminates and produces a consistent database state.

Along these guidelines, we describe a specific architecture for constraint definition and their enforcement. The components of the architecture include a *Rule Generator*, for producing all possible compensating actions, and a *Rule Analyzer and Selector*, for producing a collection of production rules such that their execution after a faulty transaction always terminates in a consistent state (possibly by rolling back the transaction); moreover, the needs of applications are modeled, so that integrity-enforcing rules reach the final state that better represents the original intentions

of the transaction's supplier. Experimental results of a prototype implementation of the proposed architecture are also described.

This work has been supported by ESPRIT II of the EC (project n.6333 IDEA) and by Centro Nazionale delle Ricerche (Progetto LOGIDATA+.

[1] S. Ceri, P. Fraternali, S. Paraboschi, L. Tanca "Automatic Generation of Production Rules for Integrity Maintenance", Tech. Rep. n. 92-054, Laboratorio di Calcolatori, Dipartimento di Elettronica, Politecnico di Milano, 1992 (submitted for publication).

[2] S. Ceri, P. Fraternali, S. Paraboschi, L. Tanca "Constraint Enforcement through Production Rules: Putting Active Databases to Work", Data Engineering, Vol. 15 No. 1-4, Dec. 1992, pp. 10-14

[3] P. Fraternali, S. Paraboschi, L. Tanca "Automatic rule generation for correction of constraint violations in active databases", Proc. 4th Int. Workshop on Foundations of Models and Languages for Data and Objects, Volkse, Germany, October 1992, pp. 93-112

[4] P. Fraternali, S. Paraboschi, "Selecting Production Rules for Constraint Maintenance: Complexity and Heuristic Solution", Tech. Rep. n. 92-057, Laboratorio di Calcolatori, Dipartimento di Elettronica, Politecnico di Milano, 1992 (submitted for publication).

[5] P. Fraternali, S. Paraboschi, "A Review of Compensating Techniques for Integrity Maintenance", paper submitted for publication.

Integrating Active and Deductive Rules in an Object Based Model

Riccardo Torlone
(IASI--CNR)

It is widely recognized that both *deductive* rules, which allow to specify deductions in a logic programming style, and *active* rules, which allow to specify actions to be undertaken whenever certain events occur, enhance the capabilities of database systems as they provide very natural and powerful mechanisms for the management of various kinds of activities. Therefore, an integration of these two linguistic paradigms into a unique homogeneous semantic framework appears to be an important and challenging issue. However, in spite of their syntactical similarity, the task is not easy because active and deductive rules differ quite a lot in the semantics traditionally provided for them.

To this aim, we first try to analyze the technical problems that make this integration difficult. Then, we present a rule-based language for an object based data model within which queries, expressed as deductive rules, and event triggered computations, expressed as production rules, can be expressed in the same fashion. The language is based on the notion of *generalized production rule*. It consists of three parts: (1) the event part, which specifies a *generalized* event triggering the rule, (2) the query part, which contains a query that at the same time states the conditions under which the rule has to be executed, and returns a set of bindings to the rest of the rule, and (3)

the update part, which contains a set of updates to be performed on the underlying database. The activation of a rule is triggered based, and its execution with respect to a certain instance produces not only a new instance but also the answer to a query with respect to the original state. The answer can be returned to the triggering event, thus simulating a top-down evaluation of a deductive rule. By using several examples, we show that it is possible to program with this model any kind of active rule as well a quite general class of deductive rules, which makes the language a powerful tool for the implementation of several database activities.

A Deductive, Object-Oriented Model as a Formal Framework for Active Database Environments

Susan D. Urban
Anton P. Karadimce
Suzanne W. Dietrich
(Arizona State University)

The successful integration of active, deductive, and object-oriented databases for the basis of future generation databases is a challenging research objective. This presentation outlines specific research directions in this area within a project known as A DOOD RANCH (Active, Deductive, Object-Oriented Databases - Relating Action, Negation, Constraints, and Horn Rules). A DOOD RANCH project has as its goal the development of an object-oriented database system that efficiently and correctly processes queries, constraints, and active rules that involve extensional and intensional data.

After outlining several research issues related to language evaluation and rule execution models, the presentation then focused on a detailed presentation of CDOL (Comprehensive, Declarative Object Language), the deductive, object-oriented data model and language on which this research is based. A unique aspect of CDOL is that it provides a framework for supporting ad-hoc, declarative update requests in an object-oriented databases while maintaining database consistency and atomicity of update requests. The framework is based on the emulation of classic update methods in an object-oriented database by a controlled, active, and user-transparent interaction between a predefined set of elementary updates and a set of integrity methods designed to maintain database consistency upon violations of integrity constraints.

Given an object-oriented framework and a declarative query language, we extend this framework by (1) isolating declaratively stated integrity constraints as as separate concept, (2) developing a high-level update language on top of the query language, and (3) developing active integrity methods from the integrity constraints. The advantage of the approach is that users can freely pose declarative ad-hoc updates without jeopardizing database consistency.

Termination and Confluence of Integrity Rules in OODB'S

Anton P. Karadimce
Susan D. Urban
(Arizona State University)

Incorporating active capabilities in OODBs requires means to guarantee correctness and predictability of active rules behavior. The properties of termination and confluence are particularly important for the class of integrity rules as active agents that maintain database consistency upon ad-hoc updates.

In this talk we model integrity rules as conditional term rewrite rules that rewrite the current database state. Rules communicate using messages that represent update requests and occurrences of events. By capturing database dynamics through conditional rewrites it is possible to use termination and confluence results developed for conditional term rewrite systems (CTRSs). A sufficient condition for termination is to have a reducing CTRS. A sufficient condition for confluence is the convergence of all contextual critical pairs. Under a set of simplifying assumptions, we present a decision procedure for ground confluence of a reducing CTRS. These results can serve as a basis for methodology and criteria for developing well-behaved integrity rules.

Panel Discussion on Active Database Systems (ADBS)

Gerti Kappel (University of Vienna)
Michael Schrefl (University of Linz)

This discussion session was motivated by the fact that several talks at the seminar have introduced quite different approaches to ADBS. Out of this situation the following questions were addressed in the discussion: 1. Are ADBS the right solution to the right problem? What are the problems ADBS should solve? 2. What are the basic concepts behind ADBMS? Should ADBMS include object-oriented, deductive, real-time, <include your favorite> concepts? 4. Provocative Statement: AI people have found that large systems of rules are hard to manage. They are looking for alternatives, e.g., model-based approaches. DB people move into rules having realized that they need to go beyond data modeling. 5. Traditionally, relational database systems have been made active by extending them with rules. They had no notion of behavior. Object-oriented database systems already have a notion of behavior, realized by methods associated with classes. Adding rule management to object-oriented database systems offers two ways of specifying behavior: method bodies and rule bodies (actions of rules). Are then message events still meaningful? Or should the reaction to a message be implemented solely by the method body? 6. ADBS help to distinguish between local behavior modeled via methods and global behavior modeled via rules introducing a second level of modularity. But appropriate languages, methods and tools for designing and working with ADBS are still missing.

5 Queries

An Extension of Path Expressions to Simplify Navigation in Object-Oriented Queries

Jan Van den Bussche (University of Antwerp (UIA))
Gottfried Vossen (University of Giessen)

Path expressions, a central ingredient of query languages for object-oriented databases, are currently used as a purely navigational vehicle. We argue that this does not fully exploit the potential expressive power as a tool to specify connections between database objects. In particular, a user should not be required to specify a path to be followed in full, but rather should provide enough information so that the underlying system can infer missing details automatically. We present and study an extended mechanism for path expressions which resembles the omission of joins in universal relation interfaces. The semantics of our mechanism is given in the general framework of a calculus-like query language. Techniques from semantic query optimization are employed to obtain efficient specifications. We also consider the possibility that links can be traversed backwards, which subsumes previous proposals to specify inverse relationships at the schema level and also fully exploits the meaning of inheritance links.

This work was partially sponsored by the NFWO.

Default-Based Semantics of Logic Programs with Negation

Catriel Beeri
(Hebrew University of Jerusalem)

In the last five years, a variety of formalisms have been presented to account for the semantics of logic programs with negation. One of the most well known is the well-founded semantics, that assigns a meaning to every program, albeit often a three-valued semantics. A similar approach, also defined for all programs is the valid semantics. A common problem with these and other approaches is that they are complex to understand and reason about.

The talk describes an approach that simplifies and unifies the approaches above. Simplification is achieved by using computations rather than fixed points. The derivation of false facts is represented abstractly by the notion of a default rule.

Among the results: Identification of useful properties of default rules; an extremely simple proof that a monotonicity property of a default rule implies existence and uniqueness of the semantics for programs defined by the rule; the well-founded and valid semantics are special cases, obtained from appropriate default rules.

The results are general, hence can presumably be applied to logic programs defined over many kinds of logic, not just over first order predicate logic.

The Role of Generic Operations in OODBs

Andreas Heuer (TU Clausthal)

Fundamental to the specification of behaviour or non-trivial object life-cycles and non-trivial update operations is an object-oriented database model allowing the derivation of new information (derived classes). This derivation should be safe, efficient, and optimizable, among others, meeting exactly the criteria of query languages in database systems.

To provide a persistent, lomg-term, centralized management of data, an object-oriented database model should support object evolution, schema evolution, and different user views on the same set of objects. Besides explicit update operations, object evolution can be performed by definition of derived classes, which are defined by query expressions.

Generic operations can be relational (producing complex values), object-generating (creating new classes with new objects) and object-preserving. The last kind of query operation either computes a subset of existing objects in the database, or a class, which may be a sub- or superclass of existing classes. One can attach attributes or methods to derived classes, so that they can be used like any base class in the database scheme.

Derived classes can be used for user views, the resolution of conflicts in dynamic binding of methods, to integrate different database models, languages, and schemes, and for a more appropriate database design avoiding redundant information and class explosion.

More information about a concrete query language in this flavour can be obtained from [1].

[1] A. Heuer and P. Sander, The LIVING IN A LATTICE rule language Data and Knowledge Engineering Vol. 9, No. 3, January 1993, 249 - 286.

Dagstuhl-Seminar 9317: List of Participants 29.06.93

Catriel **Beeri**
The Hebrew University of Jerusalem
Department of Computer Science
91904 Jerusalem
Israel
beeri@cs.huji.ac.il
tel.: +972-2-585 266

Mikael **Berndtsson**
University of Skovde
Department of Computer Science
Box 408
541 28 Skovde
Sweden
spiff@his.se
tel.: +46-500-477 600

Antony **Bonner**
University of Toronto
Dept. of Computer Science
10 King's College of Road
Toronto Ontario M5S 1A4
Canada
bonner@db.toronto.edu
tel.: +1-416-978-7441

Rolf **de By**
Universieit Twente
Faculteit der Informatica
Postbus 217
NL-7500 AE Enschede
The Netherlands
deby@cs.utwente.nl
tel.: +31-53-89 37 53

Weidong **Chen**
Southern Methodist University
Computer Science & Engineering
Dallas TX 75275-0122
USA
wchen@seas.smu.edu
tel.: +1-214-768-3097

Gregor **Engels**
University of Leiden
Department of Computer Science
P.O. Box 9512
NL-2300 RA Leiden
The Netherlands
engels@wi.leidenuniv.nl
tel.: +31-71-277065

Martin **Gogolla**
TU Braunschweig
Informatik Abt. Datenbanken
Postfach 3329
W-3300 Braunschweig
Germany
gogolla@idb.cs.tu-bs.de
tel.: +49-531-391-3102

Margret **Groß-Hardt**
Universität Gießen
Arbeitsgruppe Informatik
Arndtstraße 2
W-6300 Gießen
Germany
margret@neumann.informatik.uni-giessen.dp.de
tel.: +49-641-7022584

Hele-Mai **Haav**
Estonian Academy of Sciences
Institute of Cybernetics
Akadeemia tee 21
EE-0026 Tallinn
Estonia
helemai@ioc.ee
tel.: +7-0142-527 314

Andreas **Heuer**
TU Clausthal
Institut für Informatik
Erzstrasse 1
W-3392 Clausthal-Zellerfeld
Germany
inah@ibm.rz.tu-clausthal.de
tel.: +49-5323-72-2070

Ralf **Jungclaus**
Tu Braunschweig
Informatik
Abteilung Datenbanken
Gaußstr. 12
W-3300 Braunschweig
Germany
jungclau@idb.cs.tu-bs.de
tel.: +49-531-391 7443/7442

Gerti **Kappel**
Universität Wien
Institut für Angewandte Informatik
und Informationssysteme
Liebiggasse 4/3-4
A-1010 Wien
Austria
gerti@ifs.univie.ac.at
tel.: +43-1-432367 ext. 15

Anton P. **Karadimce**
Arizona State University
Department of Computer Science
and Engineering
Tempe AZ 85287-5406
USA
karadimc@enuxha.eas.asu.edu
tel.: +1-602-965-4196

Georg **Lausen**
Universität Mannheim
Fakultät Mathematik und Informatik
W-6800 Mannheim
Germany
lausen@pi3.informatik.uni-mannheim.de
tel.: +49-621-292-54 03

Danilo **Montesi**
Politecnico di Milano
Dipartimento di Electronica e Informazione
Piazza Vinci 32
I-20133 Milano
Italy
montesi@ipmel2.elet.polimi.it
tel.: +39-2-2399-3667

Moira **Norrie**
ETH Zürich
Department of Computer Science
ETH-Zentrum
CH-8092 Zürich
Switzerland
norrie@inf.ethz.ch
tel.: +41-1-254-72 45

Andreas **Oberweis**
Inst. für Angewandte Informatik u.
Formale Beschreibungsverfahren
Postfach 6980
W-7500 Karlsruhe
Germany
oberweis@aifb.uni-karlsruhe.de
tel.: +49 -721-608-4283

Georg **Reichwein**
INESC
Apartado 10 105
Rua Alves Redol 9
P-1017 Lisboa Codex
Portugal
ger@inesc.pt
tel.: +351-1-3100 000

Gunter **Saake**
TU Braunschweig
Informatik
Abteilung Datenbanken
Gaußstr. 12
W-3300 Braunschweig
Germany
saake@idb.cs.tu-bs.de
tel.: +49-531-391-32 67

Klaus-Dieter **Schewe**
Krampengrund 20 b
W-2000 Hamburg 67
Germany
tel.: +49-40-6039998

Marc H. **Scholl**
Universität Ulm
Fakultät für Informatik
Abt. Datenbank u. Informationssysteme
Oberer Eselsberg
D-89069 Ulm
Germany
scholl@informatik.uni-ulm.de
tel.: +49-731-502-4135 /4131 (Sekr.)

Michael **Schrefl**
Johannes Kepler Universität Linz
Inst. für Wirtschaftsinformatik
Data & Knowledge Engineering
Altenbergerstr. 69
A-4040 Linz
Austria
schrefl@dke.uni-linz.ac.at
tel.: +43-732-2468-9480

Letizia **Tanca**
Politecnico di Milano
Dipartimento di Elettronica e
Informazione
Piazza Leonardo da Vinci 32
I-20133 Milano
Italy
tanca@ipmel2.polimi.it
tel.: +39-2-23 99 36 24

Bernhard **Thalheim**
Universität Rostock
Fachbereich Informatik
Albert-Einstein-Str. 21
O-2500 Rostock 6
Germany
thalheim@informatik.uni-rostock.dbp.de
tel.: +49-381-44424

Riccardo **Torlone**
IASI-CNR
Viale Manzoni 30
I-00185 Roma
Italy
torlone@iasi.rm.cnr.it
tel.: +39-6-77161

Susan D. **Urban**
Arizona State University
Department of Computer Science and
Engineering
Tempe AZ 85287-5406
USA
urban@asuvax.eas.asu.edu
tel.: +1-602-965-2784

Jan **Van den Bussche**
University of Antwerp/UIA
Dept. of Mathematics & Computer Science
Universiteitsplein 1
B-2610 Wilrijk/Antwerp
Belgium
vdbuss@uia.ac.be
tel.: +32-3-8202403

Gottfried **Vossen**
Universität Gießen
Arbeitsgruppe Informatik
Arndtstraße 2
W-6300 Gießen
vossen@informatik.rwth-aachen.de
tel.: +49-641-702-2551

Gerhard **Weikum**
ETH Zürich
Department of Computer Science
ETH-Zentrum
CH-8092 Zürich
Switzerland
weikum@inf.ethz.ch
tel.: +41-1-254-7242

Roel **Wieringa**
Faculteit Wiskunde en Informatica
Dept. of Mathematics and
Computer Science
De Boelelaan 1081 a
NL-1081 HV Amsterdam
The Netherlands
roelw@cs.vu.nl
tel.: +31-20-548 5568

Haiyan **Ye**
ETH Zürich
Department of Computer Science
ETH-Zentrum
CH-8092 Zürich
Switzerland
ye@inf.ethz.ch
tel.: +41-1-254-7259

Zuletzt erschienene und geplante Titel:

K. Compton, J.E. Pin , W. Thomas (editors):
Automata Theory: Infinite Computations, Dagstuhl-Seminar-Report; 28, 6.-10.1.92 (9202)

H. Langmaack, E. Neuhold, M. Paul (editors):
Software Construction - Foundation and Application, Dagstuhl-Seminar-Report; 29, 13.-17.1.92 (9203)

K. Ambos-Spies, S. Homer, U. Schöning (editors):
Structure and Complexity Theory, Dagstuhl-Seminar-Report; 30, 3.-7.2.92 (9206)

B. Booß, W. Coy, J.-M. Pflüger (editors):
Limits of Information-technological Models, Dagstuhl-Seminar-Report; 31, 10.-14.2.92 (9207)

N. Habermann, W.F. Tichy (editors):
Future Directions in Software Engineering, Dagstuhl-Seminar-Report; 32; 17.2.-21.2.92 (9208)

R. Cole, E.W. Mayr, F. Meyer auf der Heide (editors):
Parallel and Distributed Algorithms; Dagstuhl-Seminar-Report; 33; 2.3.-6.3.92 (9210)

P. Klint, T. Reps, G. Snelting (editors):
Programming Environments; Dagstuhl-Seminar-Report; 34; 9.3.-13.3.92 (9211)

H.-D. Ehrich, J.A. Goguen, A. Sernadas (editors):
Foundations of Information Systems Specification and Design; Dagstuhl-Seminar-Report; 35; 16.3.-19.3.9 (9212)

W. Damm, Ch. Hankin, J. Hughes (editors):
Functional Languages:
Compiler Technology and Parallelism; Dagstuhl-Seminar-Report; 36; 23.3.-27.3.92 (9213)

Th. Beth, W. Diffie, G.J. Simmons (editors):
System Security; Dagstuhl-Seminar-Report; 37; 30.3.-3.4.92 (9214)

C.A. Ellis, M. Jarke (editors):
Distributed Cooperation in Integrated Information Systems; Dagstuhl-Seminar-Report; 38; 5.4.-9.4.92 (9215)

J. Buchmann, H. Niederreiter, A.M. Odlyzko, H.G. Zimmer (editors):
Algorithms and Number Theory, Dagstuhl-Seminar-Report; 39; 22.06.-26.06.92 (9226)

E. Börger, Y. Gurevich, H. Kleine-Büning, M.M. Richter (editors):
Computer Science Logic, Dagstuhl-Seminar-Report; 40; 13.07.-17.07.92 (9229)

J. von zur Gathen, M. Karpinski, D. Kozen (editors):
Algebraic Complexity and Parallelism, Dagstuhl-Seminar-Report; 41; 20.07.-24.07.92 (9230)

F. Baader, J. Siekmann, W. Snyder (editors):
6th International Workshop on Unification, Dagstuhl-Seminar-Report; 42; 29.07.-31.07.92 (9231)

J.W. Davenport, F. Krückeberg, R.E. Moore, S. Rump (editors):
Symbolic, algebraic and validated numerical Computation, Dagstuhl-Seminar-Report; 43; 03.08.-07.08.92 (9232)

R. Cohen, R. Kass, C. Paris, W. Wahlster (editors):
Third International Workshop on User Modeling (UM'92), Dagstuhl-Seminar-Report; 44; 10.-13.8.92 (9233)

R. Reischuk, D. Uhlig (editors):
Complexity and Realization of Boolean Functions, Dagstuhl-Seminar-Report; 45; 24.08.-28.08.92 (9235)

Th. Lengauer, D. Schomburg, M.S. Waterman (editors):
Molecular Bioinformatics, Dagstuhl-Seminar-Report; 46; 07.09.-11.09.92 (9237)

V.R. Basili, H.D. Rombach, R.W. Selby (editors):
Experimental Software Engineering Issues, Dagstuhl-Seminar-Report; 47; 14.-18.09.92 (9238)

Y. Dittrich, H. Hastedt, P. Schefe (editors):
Computer Science and Philosophy, Dagstuhl-Seminar-Report; 48; 21.09.-25.09.92 (9239)

R.P. Daley, U. Furbach, K.P. Jantke (editors):
Analogical and Inductive Inference 1992 , Dagstuhl-Seminar-Report; 49; 05.10.-09.10.92 (9241)

E. Novak, St. Smale, J.F. Traub (editors):
Algorithms and Complexity for Continuous Problems, Dagstuhl-Seminar-Report; 50; 12.10.-16.10.92 (9242)

J. Encarnação, J. Foley (editors):
Multimedia - System Architectures and Applications, Dagstuhl-Seminar-Report; 51; 02.11.-06.11.92 (9245)

F.J. Rammig, J. Staunstrup, G. Zimmermann (editors):
Self-Timed Design, Dagstuhl-Seminar-Report; 52; 30.11.-04.12.92 (9249)

B. Courcelle, H. Ehrig, G. Rozenberg, H.J. Schneider (editors):
Graph-Transformations in Computer Science, Dagstuhl-Seminar-Report; 53; 04.01.-08.01.93 (9301)

A. Arnold, L. Priese, R. Vollmar (editors):
Automata Theory: Distributed Models, Dagstuhl-Seminar-Report; 54; 11.01.-15.01.93 (9302)

W. Cellary, K. Vidyasankar , G. Vossen (editors):
Versioning in Database Management Systems, Dagstuhl-Seminar-Report; 55; 01.02.-05.02.93 (9305)

B. Becker, R. Bryant, Ch. Meinel (editors):
Computer Aided Design and Test , Dagstuhl-Seminar-Report; 56; 15.02.-19.02.93 (9307)

M. Pinkal, R. Scha, L. Schubert (editors):
Semantic Formalisms in Natural Language Processing, Dagstuhl-Seminar-Report; 57; 23.02.-26.02.93 (9308)

W. Bibel, K. Furukawa, M. Stickel (editors):
Deduction , Dagstuhl-Seminar-Report; 58; 08.03.-12.03.93 (9310)

H. Alt, B. Chazelle, E. Welzl (editors):
Computational Geometry, Dagstuhl-Seminar-Report; 59; 22.03.-26.03.93 (9312)

H. Kamp, J. Pustejovsky (editors):
Universals in the Lexicon: At the Intersection of Lexical Semantic Theories, Dagstuhl-Seminar-Report; 60; 29.03.-02.04.93 (9313)

W. Strasser, F. Wahl (editors):
Graphics & Robotics, Dagstuhl-Seminar-Report; 61; 19.04.-22.04.93 (9316)

C. Beeri, A. Heuer, G. Saake, S.D. Urban (editors):
Formal Aspects of Object Base Dynamics , Dagstuhl-Seminar-Report; 62; 26.04.-30.04.93 (9317)

R. Book, E.P.D. Pednault, D. Wotschke (editors):
Descriptional Complexity: A Multidisciplinary Perspective , Dagstuhl-Seminar-Report; 63; 03.05.-07.05.93 (9318)

H.-D. Ehrig, F. von Henke, J. Meseguer, M. Wirsing (editors):
Specification and Semantics, Dagstuhl-Seminar-Report; 64; 24.05.-28.05.93 (9321)

M. Droste, Y. Gurevich (editors):
Semantics of Programming Languages and Algebra, Dagstuhl-Seminar-Report; 65; 07.06.-11.06.93 (9323)

Ch. Lengauer, P. Quinton, Y. Robert, L. Thiele (editors):
Parallelization Techniques for Uniform Algorithms, Dagstuhl-Seminar-Report; 66; 21.06.-25.06.93 (9325)

G. Farin, H. Hagen, H. Noltemeier (editors):
Geometric Modelling, Dagstuhl-Seminar-Report; 67; 28.06.-02.07.93 (9326)