Semantics in Databases

07.01.2001-12.01.2001

organized by

L. Bertossi (Santiago, Chile), G.O.H. Katona (Budapest),
K.-D. Schewe (Massey Univ., NZ), B. Thalheim (BTU Cottbus)

In the early days of database research, issues related to database semantics played a prominent role, and papers discussing database models, conceptual design, integrity constraints, normalization often dominated major database conferences. This began to change more than a decade ago, and nowadays those issues do not appear to be part of the mainstream database research. This situation is caused by several reasons: the problem began to be too difficult, the community was hoping on solutions based on better database models, the variety of buzzwords for new models and approaches required foundation and clarification, the problems raised by application required a lot of research, the pending hope on a universal, simple and genius solutions. Nevertheless the semantical foundations are left open for most of the modern database models or are not existing for models such as UML or XML. At the same time, the community was forgetting the achievements of the early database research.
The seminar "Semantics in databases" will be a forum for researchers still interested in database semantics and which can contribute on the basis of research on database semantics and modern approaches of logics, algebra and combinatorics to the solution of very difficult problems raised in application. The first workshop "Semantics in Databases" has been organized in Rez in January 1995. The results of the discussions of the workshop have been summarized in the post workshop proceedings published in LNCS 1358.
Semantics of databases and information systems can be based on approaches which have been developed and successfully used by different communities: the logics community especially those working on non-monotonic reasoning, theorem proving, deduction, abduction, induction, finite model theory, constraint problems, non-classical semantics and categorical foundations; the type theory community especially those working on algebraic foundations of information systems; the complexity theory community especially those working on combinatorial foundations of information systems complexity; the database theory community which has been continuing research on constraints; the AI community especially those working on agents, application of non-classical logics and reasoning, deduction, abduction, induction in data and knowledge bases and non-monotonic reasoning;

the database community especially those contributing to foundations of information systems design including Web-based information systems, temporal aspects, integrity and security and database dynamics.
Research on semantics of databases is forced by a variety of problems challenging the modern information society. First, modern approaches are based on complex database and information system models. It requires the reformulation of classical results and their re-summarization including the restoration of older results. Dynamic semantics remains to be still open. Almost nothing is known on interaction semantics. Second, current technology faces difficult challenges. Information systems are becoming part of the everyday's infrastructure. They are used mainly by unexperienced users mainly on the basis of natural language understanding. The success of the Internet caused the utilization of huge varieties of semantics. XML and other ML are becoming the baseplate of Internet applications. However, XML supports semantics and consistency to a very limited extend. Third, the importance of research on semantics is hyper-raising by applications. Applications are more and more decentralized and require careful integration. UML become to be the 'inter-galactic' application specification language. It has almost no semantics. In order to develop consistent application the semantics foundation needs to be worked out.
This Dagstuhl seminar aims in bringing together different communities such as the database theory community, the logics community, the AI community, the complexity theory community, the type theory community and the database applications community. The aim of the seminar is to provide a working environment where: different streams of semantics research are becoming aware of research of other communities; maintaining the database semantics community; working on plans for a survey on semantics in databases; discussing challenges of modern applications to semantics; figuring out research challenges of the next decade. To encourage discussion, we will have talks with a responder. The length of a talk is restricted to 60 minutes. The talk is followed by a discussion introduced by the responder of about 30 minutes overall.

Contents

1 Semantics of Type Hierarchies under Open Specification

Stephen J. Hegner, Umea University, Sweden

Type hierarchies are central in many applications, including in particular systems which perform constraint-based inference. In those systems, the meet operation in the hierarchy corresponds to unification of types. Typically, a closed representation is employed, in which a single instance of a hierarchy embodies all of the information. Systems which perform such inference must also process many alternatives in search of a solution. Such search may be aided greatly by making judicious use of the join operation in the hierarchy. Therefore, it is advantageous to employ hierarchies in which both meet and join are meaningful in the context of computational inference. For such systems, closed representation has at least two drawbacks. First, since all applicable data constraints must be represented, the hierarchy can become extremely large; often too large for any practical explicit representation. Second, not all information about the relationships between classes may be available, so that the construction of a single, representative hierarchy may not be possible. The alternative to closed representation is open representation, in which the presentation is via a collection of constraints which describe the set of possible hierarchies. Open representation yields increased representational power plus much more concise representation, at the expense of increased complexity of inference. In the described work, this tradeoff is investigated; in particular, the semantic and computational aspects of open representation are studied systematically. First, a formal constraint language is developed, and a semantics for these constraints is provided in terms of bounded distributive lattices. Next, an algebraic characterization of the properties of these models is developed, in terms of a generalized form of bounded weak partial lattices. Finally, a syntactic language of inference is developed, and a characterization of complexity of inference for queries (co-NP-completeness) is obtained.

2 A Theory of Consistency Enforcement on Top of Arithmetic Logic

Sebastian Link, Massey University, New Zealand

Consistency enforcement gives an alternative in comparison with the customary program verification strategies within formal program specification languages. The existing approach uses a partial order on semantic equivalence classes of program specifications, called specialization, and aims to replace a given specification S by the greatest consistent specialization $S_{\mathcal{I}}$ which is provably consistent with respect to the given static invariant $\mathcal{I}$.
An investigation of computability or even decidability issues in connection with the constructive generation of $S_{\mathcal{I}}$ has not been possible so far due to the cardinality of the underlying logic.
In this talk we justify the axiomatic approach to Dijkstra's calculus with respect to arithmetic logic and develop a new theory on top of that basis in which the construction problem of the greatest consistent specialization for a complex specification is reduced to involved basic commands and the investigation of a precondition. In general, this construction process turns out to be incomputable and even for computable subclasses the precondition is undecidable. However, there are reasonable classes of specifications for which a computation is effective.

3 Combinatorics of Cardinality Constraints

Sven Hartmann, Rostock University, Germany

Cardinality constraints specify the number of relationships of a given type that instances of their component types may participate in. We give a solution to the finite implication problem for this constraint class, and use this to efficiently detect redundant object types, dummy cardinalities and gaps in the sets of permitted cardinalities. This question happens to be of special interest in the presence of functional dependencies. The tools we used for investigation are purely combinatorial path algebras, Forkas'lemma for integer programming, orthogonal arrays, Hajnal-Szemeredi theorem on cliques in graphs and admissible trees.

4 Cardinality Constraints in Deductive Databases

Dietmar Seipel, Würzburg University, Germany

We consider *cardinality constraints* of the form $M \Theta K$, where $\Theta \in \{=, \leq, \geq\}$, stating that "(exactly, at most, at least) K elements out of a set M have to be chosen". A set $\mathcal{C}$ of constraints can be represented by means of a positive–disjunctive deductive database $\mathcal{D}_{\mathcal{C}}$, such that the models of $\mathcal{D}_{\mathcal{C}}$ correspond to the solutions of $\mathcal{C}$. This allows for embedding cardinality constraints into applications dealing with *incomplete (disjunctive) knowledge*.
Secondly, a *sound calculus* represented by a definite logic program $\mathcal{P}$ is presented. $\mathcal{P}$ allows for directly reasoning with sets of exactly–cardinality constraints, which turned out to be very efficient; it can be used for performance reasons before $\mathcal{D}_{\mathcal{C}}$ is evaluated. For obtaining *completeness*, however, $\mathcal{D}_{\mathcal{C}}$ is necessary, since we show the theoretical result that a sound and complete program $\mathcal{P}'$ does not exist for exactly–cardinality constraints.

5 Data Mining and Statistics. A comparative study.

Oleg Seleznjev, Umea University, Sweden

Data mining (also known as knowledge discovery in databases) is used to discover valid, novel, potentially useful structure in data, with an emphasis on large observational databases. From statistical perspective data mining is a computer automated Exploratory Data Analysis (EDA) of usually large complex databases. Data mining is on the interfaces of database management, AI Machine Learning, and Statistics. We give an overview of the area and consider some statistical issues that are relevant to database mining. Some distinction and close fields for further cooperation will be discussed.

6 Semantic Classifications of Queries to Relational Databases.

Jose Maria Turull Torres, Universidad Nacional de San Luis, Universidad Tecnologica Nacional

We consider classifications of queries in terms of properties of the queries as functions, following the formal definition of computable query by A. Chandra and D. Harel, as opposite to classifications in terms of properties of expressions in a given language, which express the queries, like Conjunctive Queries, Horn Queries, different Turing Machine Complexity Classes, etc. (which can be regarded as syntactic). We define a general framework for that matter, by relaxing the property of preservation of isomorphisms. For boolean queries, we define different classes of queries by requiring the preservation of equivalence in different logics less expressive than FO (First Order). That is, for a given logic L, whenever a boolean query is evaluated on two db's which satisfy the same L sentences, then the result must be the same for both of them. As to r-ary queries, we use the notion of type from Model Theory, and we ask, for a given logic L, that whenever two db's have tuples with the same L types, then the relations resulting from the evaluation of the query in both db's should also have tuples with the same L types. We consider as target logics bounded variable fragments of FO (FOk), FOk plus Counting (Ck), bounded quantifier rank fragments of both logics, and existential fragments for them. By varying the parameter k, these classifications yield hierarchies which turn out to be orthogonal with the hierarchy defined by the Time and Space Complexity in Turing Machine Complexity. So that one important use of these hierarchies is to define finer classes of queries by intersecting layers in both hierarchies. As equivalence in most of these logics can be characterized in terms of the existence of systems of partial isomorphisms with certain properties, then we can also define the different classifications in a purely algebraic way, without considering any logic at all. These semantic classifications help in a better understanding of the nature of db queries.

7 Semantic Modeling with XML

François Bry, University Ludwig-Maximilians Munich, Germany

The presentation investigates semantic modeling with XML. It describes a proposal to view XML documents as (ground) terms of a Hermbrand universe. Attributes are coped with functions, reference (send as ID IDRFF attributes) with a notion of normal interpretation . It is further proposed to distinguish between a logical "hyperlink model" and a "browsing model" in the later assigning a behavior with the semantic notions of the (logical) former model. Problems of the current XLink specification are discussed and solutions are sketched. Finally, current running projects in Munich are brightly introduced.

8 Combinatorial Questions of the Relational Database Model

Dezsö Miklos, Alfred Renyi Institute of Mathematics, Hungary

In the theory of relational databases functional dependencies play an important role. In the first part of the talk an overview of the results concerning (minimal) realizations of given set of functional dependencies (or corresponding closure operations, sets of closed sets, family of minimal keys) were given where by minimal representation we mean the minimal needed number of records of databases of a given set of attributes resulting the object. Second part of talk attempted to give an overview of the different generalization of functional dependencies, like partial dependencies and error correcting keys. Detailed discussion of branching dependencies were given with the list of the known results similar to the "classical case" and the list of the most important open questions.

9 Specification and Semantics of Consistent Query Answers

Leopoldo Bertossi, Pontificia Universidad Catolica de Chile

We present a notion of consistent information in an inconsistent database based on the notion of minimal repair of database that violates a given set of integrity constraints. We present a computational counterpart of this semantic notion and several ways to specify and represent it. In particular, we show how to use disjunctive logic programs with exceptions to specify database repairs. This is joint work with: Marcelo Arenas, Jan Chomicki and Francisco Orchard.

10 A unifying framework for semantic constraints and security constraints in databases

Barbara Sprick, Dortmund University, Germany

We view an information system as a reactive system with a finite set of agent that con perform sequences of operations (query, update, grant rights and revoke rights) and that con reason about the system. Each agent only has a partial view on the system, we do not assume any global state. Semantic constraints are in variants over updates, they have to be fulfilled in each database instance. Security constraints are in variants over sequences of operations. They have to be maintained over all operations as well as update operations. We use a temporal and epistemic logic for distributed systems that is based on Mazurkiewicz-traces to formalize semantic constraints and security constraints in the same framework. This allows us to investigate the interaction between the two types of constraints.

11 A Semantical Framework for the Integration of Electronic Services

Burkhard Freitag, Passau University, Germany

Compositions of electronic services, which are essential for many applications using the world wide web, must be easy to specify and understand and must be executed transactionally. Yet, this is only partly addressed by the technology in use today, which is often based on ad-hoc approaches and driven by implementation-centric aspects.
The semantical framework ULTRA is tailored to the rule-based specification of composite actions and their transactional execution. An arbitrary set of basic services can be combined to obtain new complex services. Their logical semantics is represented by sets of possible transitions. In particular, ULTRA instances based on a representation of transitions by partially ordered multisets which reflect concurrent and sequential service-calls can serve as a blueprint for an actual implementation.
Serializability is *the* standard correctness criterion when it comes to the concurrent execution of transactions. Although originally developed with inter-transaction concurrency in mind, it is also used for concurrent actions within one

transaction. Yet, there are cases when applying serializability to intra-transaction concurrency can lead to unwanted effects. The
concurrency model of ULTRA was explicitly designed for intra-transaction concurrency which avoids these effects. Instead of relying on arbitrary serializations, it is based on a common starting state and a single, well-defined ending state for concurrent sub-transactions. Conformity to this model can already be checked at compilation time, such that no additional overhead at runtime is introduced.

12 A calculus of fixpoints for modeling interactive IS behavior

Srinath Srinivasa, Brandenburg Technical University at Cottbus, Germany

Information system (IS) dynamics are made up of semantic processes that map between problem and solution states. We call such semantic processes as "Problem Solving Processes" or PSPs.
An IS is a collection of PSPs. If the PSPs are algorithmic in nature, the IS is said to be a closed system; and if the PSPs are interactive in nature, the IS is said to be an open system. Interactive PSPs have one or more intermediate interactions with the user before they reach their solution states.
In this talk we show that open system dynamics are richer than closed system dynamics. Interaction that forms the core of open system dynamics is shown to be made up of three properties: computation, persistence of state and channel sensitivity. By introducing a concept called the solution space of a PSP, we show that interactive solution spaces are richer than algorithmic solution spaces. Also, interactive solution spaces are shown to require at least a three-valued system of logic for their characterization.
The formal model for the solution space is based on domain theory and deontic logics is used to obtain the three-valued system of logic.

13 Semantic Errors in Database Queries

Stefan Brass, Giessen University, Germany

Errors in SQL queries are of three types: (1) syntax errors, (2) queries which are legal but obviously not intended, no matter what the task was, (3) queries which are meaningful, but not the right one for the task at hand. The goal of this talk was to analyze and classify errors of the second type, and to develop algorithms for a tool that can detect and explain them. The following classes of semantic errors were identified: (a) queries which always give an empty result, (b) missing join conditions; (c) many duplicate answers, (d) constant result columns, (e) implied conditions, (f) unnecessary joins, (g) unnecessary DISTINCT, (h) uncorrelated EXISTS subqueries, (i) cases where all groups consist of exactly one row, i.e. grouping by a key of the intermediate query result, (j) grouping without aggregation function, (k) HAVING without GROUP BY, (l) conditions without aggregations under HAVING, (m) cases where there will be always only a single group. In case (a) the query result is certainly too small, in cases (b) to (d) the query result is too large, the other cases can understood as unnecessary complications. All of these types of errors actually appeared in exams, some statistics were given. Current systems like Oracle simply execute queries containing such errors without any warning. Most problems are undecidable, but some algorithms were given for simple cases. The envisaged tool was compared to the semantic checker "lint" for the programming language C. The importance of constraints was emphasized. Soft constraints and heuristic assumptions can also be used. Progress on this project will be reported on the web page http://www.informatik.uni-giessen.de/staff/brass/sqllint/. The author would appreciate suggestions for further errors.

14 Semantics for Valid XML Documents

Harold Boley, Deutsches Forschungszentrum fr künstliche Intelligenz, Germany

First, we introduce (DTD-)valid XML documents to establish a 'grammar-typed' syntax for Web data. Then we survey some practical (Web) aspects of three semantics: (1) Transformational semantics (incl. proof-theoretic techniques) (2) Model-theoretic semantics (Herbrand models only) (3) Metadata semantics (as XML-page annotations) We study the (Web) applicability and combination of the three semantics. Systems using or implementing such semantics are XSLT for transformations, Horn rules for model generation, and RDF for metadata. Their combination is demonstrated via a problem of (inferential, XML) data mining with report generation for data findings. We also present the Rule Markup Language (http://www.dfki.de/ruleml), which tackles the issue of XML semantics by introducing a partial order into a modular system of RuleML subDTDs.

15 Multi-Level Concurrency Control Testbed - Foundations, Evaluation and Outlook -

Markus Kirchberg, Massey University, New Zealand

Multi-level transaction schedulers adapt conflict-serializability on different levels. They exploit the fact that many low-level conflicts become irrelevant, if higher-level application semantics is taken into account.
Locking protocols for multi-level transactions have been studied since the very beginning. More recently, a hybrid concurrency control protocol for multi-level transactions called FoPL (Forward oriented Concurrency Control with Preordered Locking) has been developed. It employs access lists on the database objects and forward oriented commit validation. The basic test on all levels is based on the reordering of the access lists. When combined with queuing and deadlock detection, the protocol is not only sound, but also complete for multi-level serializable schedules.
This seems to be an advantage of FoPL compared with locking protocols, but so far a detailed analysis of its benefits is missing.
This talk introduces a multi-level concurrency control testbed used for evaluating FoPL (in comparison to the strict two-phase locking protocol). Starting with the consideration of the most necessary formal foundations we will explain the system architecture of the realized testbed informally. Moreover, the talk presents the main test proceeding, the expected outcome of the executed tests, their results, together with a general assessment of the considered concurrency control protocols or multi-level transaction schedulers, respectively. At the end a short outlook to further optimizations improving the current testbed and the processed evaluation will be given.

16 Database Semantics: State-of-the-Art versus State-of-the-Practice

Parke Godfrey, York University, Canada

In this talk, we present current work in which we explore the available functionality in leading commercial database systems (such as DB2 and Oracle) to express and handle semantic constraints in databases. Facilities for expressing semantic integrity in academic systems and concept (as with deductive databases) have long been richer than what is available in practice in commercial database systems. In recent years, however, data integrity has become more important than ever and systems developers have responded with a wider array of tools for capturing a database's semantics. New facilities include triggers, materialized views, user defined types and functions, and richer classes of integrity constraints.
Adding new features has lead to so much greater complexity, however, that the situation is now in some ways worse. New facilities offer a fragmented approach to the expression and handling of semantic properties. In expression, there is redundancy, lack of orthogonality, and non-declarativeness. In handling, there are problems of incompleteness where the system cannot always recognize when a semantic property is already declared and problems of unsoundness where it is sometimes possible to circumvent the database's integrity. We illustrate these problems with examples. We present a database designer's waterfall model for integrity and provide design considerations that make the process of defining semantic properties simpler and more efficient.
Our goal in this work is to make recommendations to database systems implementors so that they may resolve the problems with expressing and handling semantic integrity and provide better semantic facilities. We also make recommendations to database researchers on where work is most critical to have an impact on commercial design. Particular recommendations include denial constraints (expressed as materialized views) as a uniform representation for semantic constraints that may be efficiently implemented, and the need for a "semantic query facility" in RDBMSs to allow one to query about a database's semantic properties.

17 Extreme Conceptual Modeling

Oscar Pastor, Universidad Politecnica de Valencia, Spain

Conceptual modeling-based methods and their corresponding CASE tools have traditionally had a main weak point: the use of different notations for the problem space system view -centred on the what the system is- and the solution space view -centred on the how it is to be represented- These different notations, and the complex and often undefined path to go from one to the other, makes unclear the real value of conceptual modeling from a pragmatic point of view. To overcome this problem, a conceptual schema should be a precise representation of the user requirements, and it should also be fully executable, meaning by it that the resulting database and the programming tasks are really done at a higher level of abstraction, provided by the conceptual modeling constructs. A software production method based on these ideas is introduced in this work, within an object-oriented model and under the generic name of "Extreme Conceptual Modeling" (XCP). Three main aspects must be considered:

- how to assure that we capture in a precise way the user requirements,
- how to represent them properly at the problem space level in a correct conceptual schema, and
- how these conceptual constructs are properly converted into their corresponding software representations at the solution space level, obtaining the desired final software product.

To put into practice an OO software production method compliant to these XCP ideas, an extended version of the OO-Method approach is introduced in this talk. It has three main steps:

1. A requirements engineering step, where user requirements are captured in a structured way using use cases and sequence diagrams.
2. A conceptual modeling step, where those user requirements obtained in step 1 are represented in an OO conceptual schema (CS). This CS constitutes the problem space representation of the system being modeled. It consists of 4 complementary system views (static, dynamic, functional and presentation), based on UML diagrams that have a precise formal basis, and based on the specification of the corresponding analysis patterns.
3. A software representation step, where the canonical conceptual constructs used to built OO conceptual schemas are translated into their corresponding software representations. This automated software production strategy provides a true conceptual model-based compilation process.

This complete software production process, mainly based on conceptual modeling techniques, is denoted as Extreme Conceptual Modeling, due to its analogy to the ideas introduced in Extreme Programming, but in this case at a higher level of abstraction, using conceptual modeling concepts as advanced programming tools.

18 Semantics of Logic Programming with Types and Built-ins

Christoph Beierle, Hagen University, Germany

Historically, logic programming does not support a type system, and there have been many different proposals to introduce types into logic programming. E.g., from a software engineering point of view one wants to be able to detect type errors before running a program, or types are used to analyze programs and queries in order to allow optimized run-time processing for (well-typed) programs and queries. Another approach to types involves the introduction of type hierarchies, to be used for instance to speed up the deduction process, effectively by using types as constraints and extending the unification/constraint solving part of the underlying resolution procedure.
We distinguish three almost orthogonal dimensions of using types in logic programming: (1) types for proving partial correctness, (2) types as constraints, and (3) types as approximations. In the third dimension, type declarations are interpreted as approximations of the success set of a predicate. Comparing this approximation with the program model yields the basis for an approach to detect type inconsistencies in logic programs, leading to the detection of useless expressions and clauses. This approach can be extended to full Standard Prolog. It is realized in the TYPICAL system which checks Prolog programs enriched with type and predicate declarations for type inconsistencies. (joint work with Gregor Meyer)

19 A combinatorial upper bound in Data Mining

Floris Geerts, Limburgs Universitair Centrum, Belgium

In the context of mining for frequent patterns using the standard levelwise algorithm, the following question arises: given the current level and the current number of frequent patterns, what is the maximal number of candidate patterns that can be generated on the next level? We answer this question by providing a tight upper bound, derived from a combinatorial result from the sixties by Kruskal and Katona. Our result is useful to reduce the number of database scans.

20 Consistency Preserving Updates in Deductive Databases

Enric Mayol, Ernest Teniente, Universitat Politecnica de Catalunya, Spain

Deductive databases generalize relational databases by including not only base facts and integrity constraints, but also deductive rules. Several problems may arise when a deductive database is updated. The problems that are addressed in this thesis are those of integrity maintenance and view updating.
Integrity maintenance is aimed to ensure that, after a database update, integrity constraints remain satisfied. When these integrity constraints are violated by some update, such violations must be repaired by performing additional updates.
The second problem we deal with is view updating. In a deductive database, derived facts are not explicitly stored into the database and they are deduced from base facts using deductive rules. Therefore, requests to update view (or derived) facts must be appropriately translated into correct updates of the underlying base facts.
There is a close relationship between updating a deductive database and maintaining integrity constraints because, in general, integrity constraints can only be violated when performing an update. For instance, updates of base facts obtained as a result of view updating could violate some integrity constraint. On the other hand, to repair an integrity constraint could require to solve the view update problem when integrity constraint may be defined by some derived predicate.
In this presentation, we propose a method that deals satisfactorily and efficiently with both problems in an integrated way. In this sense, given an update request, our method automatically translates it, in an efficient way, into all possible ways of changing the extensional database such that the update request is satisfied and no integrity constraint is violated. The method is sound and complete in the sense that it provides all possible ways to satisfy an update request and

that each provided solution satisfies the update request and does not violate any integrity constraint. Efficiency issues are also addressed in our approach, either for integrity maintenance as well as for view updating.
To perform integrity maintenance efficiently, we propose a technique for determining the order in which integrity constraints should be handled. This technique is based on the generation at compile time of a graph, the Precedence Graph, which states the relationships between potential violations and potential repairs of integrity constraints. This graph is used at run-time to determine the proper order to check and repair integrity constraints. This order reduces significantly the number of times that each integrity constraint needs to be reconsidered after any integrity constraint repair. To improve efficiency during view updating, we propose to perform an initial analysis of the update request, the database contents and the rules of the database. The purpose of this analysis is to minimize the number of accesses to the base facts needed to translate a view update request and to explore only relevant alternatives that may lead to valid solutions of the update request.

21 Query Processing in Embedded Control Programs

David Toman and Grant Weddell, University of Waterloo, Canada

An embedded control program can be viewed as a small main-memory database system tailored to suit the needs of a particular application. For performance reasons, the program will usually define low-level data structures to store the database which in turn have to be understood by application developers and maintainers. This is in contrast with the data independence that can be achieved by using a relational database system. However, because of space and performance requirements, the use of a traditional DBMS is not likely to be feasible in this setting.
To partly address this, we have developed a query optimizer capable of compiling SQL/OQL queries on a conceptual schema to native Java or C code that navigates low-level data structures. Of particular significance is that an arbitrary collection of such structures, *perhaps already devised for an earlier version of the control program*, can be given as a part of the input. The underlying algorithms used in our optimizer to accomplish this are the main focus of this paper. The algorithms are based on a novel resource bounded plan generation mechanism in which integrity constraints abstracting the definition of stored views are applied

to source queries to extend the search space of possible query plans. We also report on some preliminary experiment results that suggest generated code navigates the data structures with an efficiency comparable to code written directly by expert programmers.

22 On the Semantics of Views: An Amateur‘s Retrospective

Rainer Manthey, Bonn University, Germany

Views are a well-established concept in databases today which has been thoroughly investigated in the field of deductive databases. During the last 25 years, a large body of literature on this issue has been produced. But now it is pretty hard to identify key issues and insights which still appear relevant from today's perspective. Most trouble in defining view semantics has been caused by recursions, in particular if interleaved with negation: perfect models, stable and well-founded models have been proposed; iterated, alternating and conditional fixpoint semantics are associated with them. Recently, relevance of all these results increased considerably as recursive views have been admitted in the new SQL'99 standard: All problems arising from recursions are now likely to turn up in "real" (commercial) situations. Therefore a survey of these results might be of interest. I called it "an amateur's retrospective", because I did not contribute to the topic of view semantics myself (whereas several participants of this seminar did) - my own experience is restricted to intensive teaching only. But sometimes the view of a "knowledge outsider" may be of interest even to the experts.

23 Context in Information Services

Thomas Feyer, Brandenburg Technical University at Cottbus, Germany

In the early 80s, the first formal approaches towards a notion of context were introduced by Kamp's discourse representation theory, Barwise and Perry's situ-

ation semantics, and Sowas conceptual graphs. Later, in the beginning 90s, John McCarthy's introduction of the ist(c,p) predicate (meaning ,,proposition p is true in the context c") together with the lifting rules for exchanging information between different contexts stimulated a lot of further research, for example, the Cyc project aiming at providing a knowledge base for common sense.
Our motivation to investigate in formal models of context is originated by the current problems in the design/maintenance of adaptive information services available for a large variety of users and platforms, in particular, in the Web. To specify the notion of adaption for these applications, an explicit representation of context is a necessary requirement. In the talk, we illustrate how context can be incorporated into our design system — which basically models application semantics by higher-order place/transition nets in which data elements are trees. The main intuition is to attach context directly to data elements and maintain the context through a separate context management system. The adaption is then defined by rules that may alter the data elements according to their context. In further work, we are aiming to define a notion of consistent adaption, and to apply it in the design.

24 Proposing ASMs for Database Modeling

Egon Boerger, Universia degli Studi di Pisa, Italy

This talk is targeted at showing the power of Abstract State Machines (ASM) for high-level design, verification and validation of complex systems. I hope to attract some researchers in the database community to look into possible applications of the method in their field. I survey the basic definition of the concept and its main applications over the last decade, and illustrate it by an ASM definition of Java, its compilation and its bytecode verification and execution on the Java Virtual Machine (JVM).
Concise abstract (ASM) code is provided for interpreting Java programs and for a fast Java Virtual Machine (JVM) which serves as platform for a general compilation scheme of Java programs to JVM code. These definitions allow us to prove that any compiler that satisfies some natural conditions compiles legal Java code correctly.
The modularity of the ASM definitions for Java, the JVM and the compilation scheme exhibit orthogonal language, machine and compiler components, which fit together and provide the basis for a stepwise and provably correct design–for–reuse. The above definition for the JVM is extended to a defensive JVM

which can be used as Java independent platform for safely executing byte code. Separating the checking mechanism from the execution we obtain a model for the bytecode verifier which can be proved to be correct and complete. By extending the compiler to a certifying compiler we can prove our compiler to generate code which successfully passes the bytecode verifier. As last step this bytecode verifier is integrated into a loading machine which together with the executing ASM provides a full JVM model.
These abstract machines for Java, the JVM and the compiler have also been refined to executable machines, which have been validated through experimentation with ASMGofer (available at http://www.uni-ulm.de/ s_jschmi/AsmGofer). A full documentation is in the book by R. Staerk, J. Schmid, E. Boerger: Java and the=20 Java Virtual Machine: Definition, Verification, Validation. Springer-Verlag (2001).

25 Subsupiton and XML Types

Gabriel Kuper, Bell Labs, USA

This talk described the concept of subsuption for XML type systems: Subsuption is weaker than inclusion, but permits the reuse of type information form the database, for query process data evaluation etc. We discuss issues of expressive power, and the impact notion of greatest subsuption of 2 schemes.

26 On No Information Nulls in Relational Databases

Hans-Joachim Klein, Kiel University, Germany

The interpretation of a missing data value as "the value is either unknown or not existent" is more difficult to handle correctly than the "unknown but existent" interpretation which has been studied quite extensively. Difficulties arise because the set of possible worlds described by a database with 'no information' nulls includes possible worlds with attribute values definitely known to not exist. Database semantics has to be defined carefully in order not to run into conflicts with the closed world assumption. To achieve this goal negative information has

to be represented in a unique way. In general, additional information concerning relationships between attributes is necessary for guaranteeing unique representation.
In this talk we first demonstrate how semantics of the standard query language SQL allows to interpret null values by choosing appropriate query formulations. These formulations are equivalent for complete databases but not for databases with null values. Then we discuss sure and maybe information in answers and show why many proposals for query evaluation given in the literature do not guarantee sure information. In order to cope with problems in connection with the closed world assumption and not-existent values in possible worlds we introduce so-called concepts. For databases with concepts given for relation types in the corresponding schema, we define semantics based on extensions of relations. Then we show how to evaluate queries efficiently by applying a "switch strategy" such that the answers always represent sure information w.r.t. this semantics. Because of well-known efficiency problems, answers are not complete in general. The method can be applied to the standard query language SQL where missing values are modeled by "no information" nulls.

27 Paraconsistency: A Non-Standard Perspective of Logic, Resolution and Database Semantics

Hendrik Decker, Siemens Business Services, Germany

The purpose of this talk is to raise curiosity about paraconsistency among the participants of the seminar. Its ultimate goal is to convince them that paraconsistency should really be the standard logic foundation for the semantics of databases.
Classical logic takes an absolutely intolerant stand on inconsistency, although the latter is commonplace in reality. In particular, classical logic fails to capture that, in practice, querying inconsistent deductive systems ordinarily produces reasonably useful answers. Paraconsistent logic restricts classical logic so that, from inconsistency, not everything becomes derivable. We briefly survey some formal aspects of the history of mathematical logic, from Aristotle via Frege, Russel, Brower, Heyting, Kolmogorov, Lukasiewicz to Jasowski and da Costa. In this survey, we focus on axioms and inference rules used or renounced in classical logic, intuitionism, paraconsistency, resolution, logic programming, abduction

and deductive databases. We argue that (even without extensions by annotation, additional truth values, probabilistic or other constructs), resolution, logic programming, abductive logic programming and deductive databases already have the potential of capturing a practically viable kind of paraconsistency.

28 Lineage and Reliability

Gerd Wagner, Eindhoven University of Technology, The Netherlands

In many database applications it is assumed that all information sources authorized to enter new information into the database are equally competent, completely reliable and honest, and therefore new information always overrides old information. Under these assumptions, it is not necessary to distinguish between, and keep track of, different information suppliers. Concerning their credibility and competence, they are all treated on par with the owner of the database.
In some application domains, however, these assumptions are not realistic, and different information sources have to be distinguished in order to identify epistemically independent inputs and to take possibly different degrees of reliability into account. We present an approach to the problem of information lineage and reliability that is based on the following four principles: 1) Information items are being stored together with a label identifying their source. 2) An assignment of reliability degrees to information sources is maintained in the database, and can be adapted at runtime, e.g., with the help of machine learning methods. 3) The answers to a query are qualified by a degree of certainty that is derived from the reliability degrees of the involved sources. 4) If a new input affects an already-stored piece of information, the result of assimilating it into the database depends on its source label: if it comes from the same source as the already-stored item, an overriding update is performed; if it comes from a different source, the input item is stored together with its source label in the database, and an evidence combination operation is performed when the resulting database is being queried subsequently.

29 Intentions of Updates in Integrity Enforcement – Characterization and Preservation

Mira Balaban, Ben-Gurion University, Israel
Steffen Jurk, Brandenburg Technical University at Cottbus, Germany

In dynamic situations, where operations can violate necessary properties, the role of integrity enforcement is to ensure semantic correctness of data by applying additional *repairing actions.* The issues that are involved in this process include *termination, quality of repairing actions, effect preservation*, and *efficiency.* The process can be performed at *run-time*, as in the broadly accepted *Rule Triggering Systems* in database products, or at *compile-time.* In general, there is a tradeoff between run-time overhead and the amount of precomputations at compile-time. Effect preservation deals with the characterization of intentions of updates, and is responsible for their preservation. The challenge of compile-time effect preservation is to characterize fine, run-time sensitive effects, so to avoid rejection of good repairs.
In this talk, we show that *Rule Triggering Systems* are not capable to preserve the intention of a users transaction and we discuss possible characterizations of effects. An *effect* theory for the domain of sets is presented.