Dagstuhl-Seminar 9529

Role of Logic in Information Systems

17–21 July 1995

organized by

Jan Chomicki (Kansas State University)

Gunter Saake (University of Magdeburg)

Christina Sernadas (Instituto Superior Tecnico Lisboa)

Contents

1 Preface

There is a long tradition of using formal logics in the design and realization of information systems. Logics are for example used as frameworks for describing the relevant information managed by information systems and to give formal background to data models and query languages.

The workshop is intended to bring together researchers working in different aspects of the use of logics in information systems such as specification of information systems with logic frameworks (namely dynamic or temporal logic), logics for temporal databases, logic-based queries and updates, consistency checking and verification, etc. The list of participants covers people working on different aspects of logical frameworks as well as people using logics in practical information system technology. The aim of this workshop is to bring together researchers in logics and people who actually build IS applications or IS tools. This combination may foster interaction between people that do not usually meet at conferences.

Since the topic "Logics in Information Systems" is very broad, we will concentrate on three aspects and their relation.

- **Design of Dynamic Information Systems**

  Logics for information system behavior, provable correct information systems, conceptual modelling, specification and verification

- **Temporal Databases**

  Data models for temporal data bases, temporal query languages, temporal integrity, time models, real time applications

- **Logics for Querying and Integrity Constraints**

  Deductive databases, query optimization, integrity checking, logics for updates, query language semantics

Presentations combining aspects of these three topics are of special interest, e.g. using one logical framework (eg, temporal or dynamic logic) for designing and querying a temporal database. Bringing together researchers of the three mentioned disciplines seems to be promising because they start to use similar formalisms (for example, non-standard logics) and techniques (for example, automated reasoning) in different areas having different research traditions.

The three topics are seen under the main theme "Practical Impact of Logics for Information System Applications". We are specially interested in contributions showing the applicability of logic techniques to information system applications.

2 Final Program

Monday, July 17, 1995

Morning Session (09.00-12.15)
Chair: Gunter Saake

1. *E. Dubois, P. Du Bois*: The Role of Logic in Requirement Engineering
2. *J.-J. Meyer*: Deontic Logic in Information Systems

Afternoon Session (13.30–18.00)
Chair: Jan Chomicki

3. *H.-D. Ehrich*: Granularity for Specifying Information Systems
4. *P.-Y. Schobbens*: Multi-Preferential Logics for Information Systems
5. *B. Thalheim*: Fundamentals of Object-Oriented Database Models

Tuesday, July 18, 1995

Morning Session (09.00–12.15)
Chair: Bernhard Thalheim

6. *M. Gogolla*: Identifying Objects by Declarative Queries
7. *S. Brass*: Query Evaluation for Modular Specifications
8. *R. Marti*: Constraint Propagation and Intensional Answers in Deductive Databases

Afternoon Session (13.30–18.00)
Chair: Hans-Dieter Ehrich

9. *B. Freitag*: Using a Modal Logic for Component Library Management
10. *B. Paech*: Formal Description Techniques in Requirements Engineering
11. *K. Böhm*: Real-Time Object Specification Logic
12. *R. Wieringa*: A Comparison of the Notations Used in the Shlaer-Mellor Method and in TCM

Wednesday, July 19, 1995

Morning Session (09.00–12.15)
Chair: Udo Lipeck

13. *H. Decker*: The Role of Logic Programming for Knowledged Information Systems
14. *J. Dix*: The Role of Partial Evaluation for Disjunctive Deductive Databases
15. *M. Ryan*: Defaults in Logic and the Lexicographic Order

Thursday, July 20, 1995

Morning Session (09.00–12.15)
Chair: Gunter Saake

17. *S. Conrad*: Logical Support for Active Databases
18. *B. Ludäscher*: Logical Foundations of Active Rules
19. *P. Ladkin*: Airbus A320 Braking as Predicate-Action Diagrams
20. *J. Chomicki*: Logical Semantics and Database Interoperability

Afternoon Session (13.30–18.00)
Chair: Jan Chomicki

21. *P. A. Sistla*: Temporal Triggers in Active Database Systems
22. *R. Feenstra*: Constraint-based Reachability Analysis of Information Systems
23. *Y. Venema*: A Modal Predicate Calculus
24. *M. Schrefl*: Inheritance of Business Rules

Friday, July 21, 1995

Morning Session (09.00–12.15)
Chair: Jan Chomicki

25. *G. Saake*: From Object Specification towards Logic-based Agent Modelling
26. *B. Thalheim*: Constraints in Database Models (Overview)
27. *Discussion & Report of the Working Group*

3 Abstracts of Presentations

The following abstracts of presentations appear in alphabetical order of the speakers.

Real-Time Object Specification Logic

Klemens Böhm (GMD Darmstadt)

Abstract

A general formalism to describe processes in an organization should allow for the expression of real-time constraints. A real-time object specification logic (RTOSL) is motivated and presented. The logic is a linear-time point-based formalism with real-numbered time and a (syntactic) variant of the explicit-clock-variable approach. The main difference to previous real-time extensions of temporal logic is that this one is embedded in a logic for the specification of object behavior. The object paradigm has turned out to be advantageous to model arbitrary facets of organizations. On the one hand, RTOSL is an extension of the Object Specification Logic developed by A. Sernadas and others, on the other hand, it is an application of approaches from the field of real-time temporal logic.

Query Evaluation for Modular Specifications

Stefan Brass (University of Hannover)

Abstract

We proposed a modular approach to knowledge representation and especially considered the case where the modules are supernormal default theories. In general, a module simply defines a set of intended models and an equivalence relation: Two interpretations are equivalent iff they differ only locally (e.g., in the truth values of the predicates defined by this module). Assuming for simplicity a global signature, a model of a modular specification is an interpretation which is a locally intended model of every module in the specification.

Since our approach is able to distinguish between defining and using predicate occurrences, it solves a number of problems in nonmonotonic reasoning originating from unwanted contrapositions, e.g. the Yale shooting problem. The approach can be seen as a generalization of logic programming and has some relation to prioritized defaults.

Query evaluation for modular specifications can possibly profit from the module information by looking only at the relevant modules: This is possible if every module is tolerant (i.e., for every interpretation, there is an equivalent locally intended model) and there is a well-founded and acyclic dependency relation between the modules. Furthermore, it is possible to use specialized query-evaluation algorithms, if some module happens to be, e.g., a relational database or a logic program.

In order to evaluate a query modularly, each module partially evaluates the query with the given local information. Formally, an interpretation should satisfy the resulting reduct iff all equivalent locally intended models satisfy the query. Then the query holds in all globally intended models iff repeated partial evaluation (respecting the dependencies between the modules) results in a tautology.

Logical Semantics and Database Interoperability

Jan Chomicki (Kansas State University)

Abstract

In various important application areas, e.g., temporal and spatial databases, there is an abundance of different, usually incompatible, data model proposals. It is essential that those data models can be used together. This is database interoperability.

We present here a formal model of database interoperability, distinguishing between data interoperability (mapping between database instances) and query interoperability (mapping between queries). We propose constraint databases as a layer that mediates between data models. Our approach is schema-independent, which means that not only stored database relations but also views and query results can be interoperated.

Logical Support for Active Databases

Stefan Conrad (University of Magdeburg)
joint work with Can Türker (University of Magdeburg)

Abstract

In this talk we present first ideas on using logic for active databases. One possibility is to employ a logic for giving active rules (ECA rules) a logical semantics. Then certain properties like termination or confluence may be proved. Another way of employing logic for active databases is using an specification language (based on logic) as a design language for active database systems. In the talk we concentrate on this aspect and show by means of an example how to derive ECA rules from a given object specification. As specification language we use the object-oriented language TROLL which is based on temporal logic. For this language we present and discuss a first, rather naive translation to ECA rules. The resulting active rules can be used as a basis for implementing or for prototyping the object specification using an active object-oriented database system. Of course, there are certain limitations for a general translation of TROLL concepts to ECA rules of an active object-oriented database system. For instance, it is possible to specify liveness conditions for objects in TROLL which cannot be represented by ECA rules.

The Role of Logic Programming in Knowledge-Based Information Systems

Hendrik Decker (Siemens München)

Abstract

We first recapitulate the use of horn clause logic, and extensions thereof, as a language for knowledge representation, query evaluation, integrity specification and consistency checking. Then, we sketch a scenario of knowledge engineering in information systems, distinguishing knowledge acquisition, knowledge assimilation and problem solving (query answering). These tasks are supported by the logic programming techniques of induction, abduction and deduction. As an illustration of the role of abduction in knowledge engineering, we discuss some features of abductive logic programming procedures for integrity checking, view updating and hypothetical reasoning with default and non-default abducibles.

The Role of Partial Evaluation for Disjunctive Logic Programs

Jürgen Dix (University of Koblenz-Landau)

Abstract

Last year it has been shown independently by Brass/Dix and Sakama/Seki that the stable semantics satisfies a strong Generalized Partial Evaluation Principle (GPPE) for Disjunctive logic programs.

We show in this talk that

1. STABLE not only satisfies GPPE (as do many other semantics as well) but is already uniquely determined by it, as a semantics based on two-valued models,

2. there exists a nice framework (whose main ingredient is GPPE) to define the disjunctive counterpart of WFS, we call it D-WFS, by purely abstract properties. In addition, this approach gives us for free a bottom-up query evaluation algorithm. The main notion is that of a normal form, the residual program, which is sound for many semantics (so it is a precomputation step) and complete for D-WFS.

The Role of Logic in Requirement Engineering

Eric Dubois & Philippe Du Bois (University of Namur)

Abstract

Requirements Engineering (RE), the early phase of the software life cycle, is recognized as an important activity. It deals with the modelling of composite systems (i.e. systems composed of interacting entities of different nature e.g. people, software components, hardware devices). The Albert II language was designed for that purpose. It is formal since this is an important characteristic for being able to build interesting support tools. Another crucial feature of the language is its naturalness (i.e. the closeness between languages constructs and the real-world concepts).

Albert II is an agent-oriented language: a specification is organized as a hierarchy of interacting agents. An agent is characterized by its time-varying state and the actions it may perform. An Albert II specification defines the set of admissible lives (i.e. behaviours). This is done using templates of constraints. Templates have been isolated to record usual constraints met in practice. The semantics of Albert II has been defined as a mapping between Albert II constructs and a set of axioms in a real-time temporal logic.

We plan to build tools around Albert II in order to make its use possible for real application. These tools should support both verification (prove that the specification is logically consistent) and validation (show that the specification reflects faithfully the customers' needs).

Granularity in Specifying Information Systems

Hans-Dieter Ehrich (Technical University of Braunschweig)

Abstract

Information systems are open, reactive, and often distributed systems that maintain persistent data. The Oblog family of languages developed in cooperation with A. and C. Sernadas (Lisbon) aims at specifying information systems on a high level of abstraction. The development is rooted in conceptual modeling, abstract data type theory, concurrency theory, and temporal logic. The approach taken is object-oriented.

As in other object-oriented languages, the basic specification granules are objects and classes. Objects are granules of autonomy, while classes are granules of community. Types are specified in combination with classes, they are granules of uniformity.

This talk puts forward the idea of introducing explicitly two further granules of specification: *modules* as granules of reusability, and *nodes* as granules of locality.

Modules are reusable entries in software catalogues. They are generic and offer two kinds of interfaces: an internal interface for the services used, and one or more external interfaces for the services offered. These interfaces are internally related by reification. System components are produced by instantiation.

Nodes are units of distribution. Operations in different nodes are fully concurrent. Nodes and modules are orthogonal concepts, i.e., there may be several modules at a node, and a module may be distributed over several nodes.

For coping with distribution, an extension of temporal logic called DTL (distributed temporal logic) is being developed. Troll2, the most recent offspring of the Oblog family, and the *TBench*, a Case tool developed for using Troll2, will be extended to incorporate language features for modularization and distribution.

Constraint-based Reachability Analysis of Information Systems

Remco Feenstra (Free University of Amsterdam)

Abstract

When validating a formal specification of an information system, an analyst needs to communicate the meaning of the specification to domain specialists to enable them to judge the contents of the specification. Because they cannot be expected to understand the formal specification, the analyst will often use concrete examples of system behavior, also known as scenarios. Animation tools can be used to check whether a scenario is possible according to the formal specification. They also can aid in creating graphical presentations of the scenario. Typically, these tools present the current system state graphically, and can show the effect of a user-selected transaction on it.

We propose extending animation tools with a capability to perform automatic reasoning in order to solve reachability queries. A reachability query asks for a scenario that satisfies constraints on the states and transactions occurring in it, while being possible according to the formal specification. The analyst may then use reachability queries to characterize the scenarios of interest for a specific validation discussion, having the animation tool search for an example scenario that satisfies the query. Thus, he no longer needs to specify a starting state in detail and select the appropriate transaction instances. Moreover, the animation system may use the reachability query and information gained during the process of finding a scenario to improve the understandability of the presentation of the scenario by automatically leaving out details that are less relevant for showing this reachability property.

Although solving reachability queries is an undecidable problem for most logic-based specification languages suitable for specifying information system behavior, we might still try to solve them for simple cases. This could be very useful in validation practice. We currently are developing a prototype system that attempts to do this. We will sketch how this system uses various techniques, including planning by goal regression and model generation theorem proving, to solve reachability queries.

Using a Modal Logic for Component Library Management

Burkhard Freitag (University of Passau)
joint work with Bernhard Steffen, Tiziana Margaria-Steffen, Ulrich Zukowski

Abstract

Logic is ubiquitous in information systems and appears in many different shapes and at many different levels, e.g. as a query language, as a specification and verification language, or as an implementation language. The DaCapo system presented in this talk uses a modal logic as a query language and a deductive database language as both the data description and the implementation language.

The DaCapo system aims at the *automatic construction of sequences of components* from a repository of existing components according to a (possibly incomplete) specification. A wide range of application domains can be accommodated. Components can belong to as different domains as (descriptions of) software modules, pieces of course material, or direct flights, just to name a few. Correspondingly, the answers to a query may denote sequences of software modules linked together via input/output channels, relevant paths through a network of text documents, or a sequence of direct flights forming an allowable flight route.

While most repository management systems concentrate on the retrieval of single components, the DaCapo system is an attempt towards the integration of component retrieval and component composition. Sequences of components are described by a single specification formula along three "dimensions": a *module constraint* restricts the set of candidate components by semantic properties whereas an *interface constraint* defines the compatibility of adjacent components, e.g. type compatibility. Both are applied to guide the retrieval of single components. The third kind of constraints are *ordering constraints* expressed by modal operators, that specify a relative order between (groups of) components.

One of the interesting properties of this approach is its flexibility: The atomic propositions of the modal query language are the answers to conventional deductive database queries. Therefore, that part of the specification language that serves to specify the local properties of components, i.e. the interface and module constraints, is completely adaptable to a specific application domain.

Identifying Objects by Declarative Queries

Martin Gogolla (University of Bremen)

Abstract

One of the central issues in object-orientation is object identity. In object models with identity, objects can exist independent of the values of their attributes. Advantages of such identities are object sharing and object updates. The importance of the identity concept with different perspectives from programming and database languages and philosophy has also been emphasized.

In this talk we study object identification in data models, especially in semantic, Entity-Relationship, and Object-Oriented data models. Various known approaches to object identification are shown, and an alternative proposal to the topic is put forward. The main new idea is to attach to each object type an arbitrary query in order to observe a unique, identifying property of objects of the corresponding type. This proposal seems to be more general than the well-known "deep equality" approach.

The talk is organized as follows. First we introduce an example schema which will be used throughout the paper, and sketch the interpretation of object schemas in an informal way. Several known and our new proposal (the so-called observation term approach) to object identification are studied afterwards. The advantages of our new proposal are discussed, and its formal definition is mentioned. The talk ends with some concluding remarks.

Airbus A320 Braking as Predicate-Action Diagrams

Peter Ladkin (University of Bielefeld)

Abstract

In September 1993, a Lufthansa Airbus A320 landed in a rainstorm in Warsaw. It landed faster than normal, but the spoilers didn't deploy until 9 seconds after landing and the brakes were not effective until 4 seconds after that. The aircraft ran off the end of the runway, surmounted an earth bank, and finally burned. Luckily, only two people died.

It turns out that some of this behavior was predictable from the design, and from the Flight Crew Operating Manual. It was not immediately clear following the accident that the operator, Lufthansa, the manufacturer, Airbus, and the Luftfahrtbundesamt had the same view concerning how the aircraft should be flown and how it should behave in certain circumstances.

Could we improve the way in which the aircraft behavior is specified to the operators? We use the Predicate-Action Diagrams of Lamport to express the description of the operation of the Airbus A320 braking systems contained in the Flight Crew Operating Manual. This helps identify ambiguities and incompleteness. Furthermore, we argue that the information present in the predicate-action diagrams is easier to read and understand than its textual counterpart.

Towards Logical Foundations of Active Rules

Bertram Ludäscher (University of Freiburg)
joint work with U. Hamann and G. Lausen

Abstract

We present a state-oriented extension to Datalog called *Statelog* which comprises two kinds of rules within a simple, yet flexible unified framework, i.e. (i) passive *query rules* used to express queries and static integrity constraints, and (ii) active *transition rules* for defining state-changing operations. We show that Statelog is powerful enough to capture many typical active rule applications (including complex updates, integrity enforcement and incremental view maintenance strategies) while retaining a declarative, set-oriented, and deterministic semantics thus reconciling active and deductive databases. Using the Statelog framework, formal results on termination, expressiveness and complexity of active rules are established which carry over to related state-oriented languages like XY-stratified Datalog. We prove that it is undecidable for general Statelog programs whether a program terminates on all databases. We then develop syntactic criteria which guarantee program termination and investigate the impact of these restrictions on expressive power. The class of Δ-*monotone programs* extends previous results on Statelog and is of practical importance since it includes self-triggering active rules and expresses all Fixpoint transformations. In addition, our syntactic conditions yield an alternative static analysis technique based on the logical notion of monotony whereas previous approaches use execution and triggering graphs.

Constraint Propagation and Intensional Answers in Deductive Databases

Robert Marti (ETH Zürich)

Abstract

The solving of constraint satisfaction problems is based on the ability to represent and manipulate partial information, i.e., constraints such as equalities and inequalities. Some practical applications in which constraint handling is a natural requirement include configuration tasks, circuit design, and temporal reasoning. In this project, we apply ideas from Constraint Logic Programming (CLP) to deductive database systems. Partial information is represented by non-ground facts and rules which are not range- restricted. Such facts and rules are rewritten at compile-time, using a combination of constraint propagation and simplification techniques. As a consequence, the system may not able to find concrete values for some of the variables occurring in the query. In this case, it will return answers which consist of an extensional part, i.e., bindings of concrete values to variables, and an intensional part, i.e., formulas which constrain the unbound variables. An interesting application of this technology is the representation of temporal knowledge, in particular of periodically recurring events.

The Role of Deontic Logic in the Specification of Information Systems

John-Jules Meyer (University of Utrecht)

Abstract

Deontic logic is a logic to reason about ideal versus actual behaviour. [It does so by means of operators for prohibition, permission and obligation.] As such it is very useful for the specification of "soft" integrity constraints which may be violated, since in deontic logic one can express that ideally these constraints are complied to, but actually violations may occur. Moreover, one has the ability to specify what should happen in case of a violation (in order to restore integrity or take some other compensating action, for instance).

However, (standard) deontic logic (SDL) itself is not without problems: it has been plagued by many paradoxes since its inception. When (proposing to) using deontic logic for specification, these should be dealt with one way or another, either by resolving the paradoxes or by explaining why they are harmless in the context of application. Our claim is that all paradoxes are harmless except for the ones concerning the so-called contrary-to-duty imperatives, such as the Chisholm paradox. An instance of this comprises the following assertions:

- there should be no error.
- if there is an error, there should be a warning.
- if there is no error, there should be no warning.
- there is an error.

In SDL this set cannot be represented consistently (without one of the assertions being redundant). Furthermore, we claim that because of the conceptual difference one should distinguish between the notions of ought-to-do ('Tunsollen') and ought-to-be ('Seinsollen'), to the extent of using different logics for these two notions. We present a multi-modal extension S5O(n) of SDL to represent the ought-to-be version of the Chisholm set properly. For the ought-to-do variant we employ a reduction to dynamic logic, and show how the Chisholm set can be treated adequately in this setting. Finally we propose a way of integrating both ought-to-be and ought-to-do reasoning, enabling one to draw conclusions from ought-to-be constraints to ought-to-do ones.

The Role of Formal Description Techniques in Requirements Engineering

Barbara Paech (Technical University of München)

Abstract

We introduce the SYSLAB-Project which aims at bridging the gap between formal and pragmatic software development methods. The notations of formal methods have a fixed syntax and a mathematical semantics, but the specifications are difficult to read and work with. On the other hand the graphical description techniques of pragmatic methods are easier to understand and offer structural operations for working with these notations, but lack precise semantics (and often even precise syntax). The SYSLAB-project develops a new software development method offering mathematically founded description techniques for various system aspects. In our approach a system is viewed as a set of communicating actors carrying out certain activities using data which is internally stored or communicated from other actors.

In the talk we introduce by way of example the framework of stream processing functions (FOCUS) which serves as the mathematical basis. Then we give an overview on the process model (which emphasizes business process modeling, (user) interface design and architectural design) and on the description techniques of the SYSLAB-method.

The Lexicographic Combination of Preference Relations

Mark Ryan (University of Birmingham)
joint work with Pierre-Yves Schobbens (University of Namur) and
Hajnal Andréka (University of Budapest)

Abstract

The preferential approach to the semantics of defaults assigns to each default formula ϕ a *preference relation* $\sqsubseteq_\phi$ on interpretation structures: $m \sqsubseteq_\phi n$ says that m satisfies ϕ at least as well as n does. The motivation for this is that, since in general defaults are not fully satisfied, we must examine cases in which they are satisfied only *as much as possible* subject to other constraints.

We can present a prioritised bag of defaults by $(I, <, v)$ where $(I, <)$ is a poset and v maps I to formulas. Thus, $i < j$ means that the formula $v(i)$ has greater priority over the formula $v(j)$. To give a semantics to such a prioritised bag of defaults involves combining the relations $\sqsubseteq_{v(i)}$ $(i \in I)$ into a single 'consensus' relation $\sqsubseteq$ according to the priority $<$. This is done by the following lexicographic rule:

$$m \sqsubseteq n \Leftrightarrow \forall i \in I.(m \sqsubseteq_{v(i)} n \vee \exists j \in I.(j < i \wedge m < n))$$

In words: m must be better than n at each formula $(v(i))$ except possibly those at which there is another formula $(v(j))$ with greater priority at which m is strictly better.

We give several results about this definition, including the following.

1. Propagation of properties. If all the $\sqsubseteq_\phi$ have a certain property P and another condition holds, then the resulting relation $\sqsubseteq$ also has P. Example: $P =$ transitivity, $C = <$ is well-founded. Other examples of P considered are well-founded, Zorn, reflexive, symmetric, etc.

2. Proof rules. View $(I, <, v)$ as a graph denoting a $v(I)$-ary operator which takes relations to a relation. We show that certain syntactic conditions on the respective graphs are necessary and sufficient to prove equality (or inclusion) between two operators.

3. Algebraic theory. Consider two such graphs, given by $I = \{1, 2\}$ with $1 < 2$ and $I = \{1, 2\}$ with $1 \# 2$. Applying the lexicographic definition we see that these denote the binary operators on relations:

$$x/y = (x \cap y) \cup (y \cap \overline{y^{-1}})$$
$$x \| y = x \cap y$$

   A preferential algebra is a set of relations closed order these operations. Certain equations hold in all preferential algebras, such as $x/(y/z) = (x/y)/z$ and $x\|y = y\|x$. We present 7 equations which are sound and complete for all preferential algebras.

From Object Specifications towards Logic-based Agent Modelling

Gunter Saake (University of Magdeburg)

Abstract

Object-oriented specification languages specify the behaviour of objects by a set of behaviour-restricting formulae, for example behaviour axioms in temporal logic. Examples for such languages are TROLL, LCM or Albert (all presented in some detail on this workshop). An object specification restricts the whole life of an object and cannot be modified during object lifetime. This restriction to a fixed behaviour description motivates the search for a more flexible specification formalism allowing behaviour evolution during object existence.

The concept of agent as actor in an information system has several similarities to the object concept, for example the notion of internal state and behaviour driven by events. Agent behaviour, however, should be adaptable to changing situations, depending on former interactions with other agents. We propose to extend a temporal logic designed to specify objects towards an agent modelling logic. This extended logic allows to modify behaviour axioms during the lifetime of an object in a controlled way. This extension does not capture all facets of agent modelling but may be a first step towards a temporal logic based agent description.

The presented work has ist origins in discussions with Amílcar and Cristina Sernadas on object evolution as well as in the work in the European ModelAge working group (Modelling of Agents).

Parts of the presented ideas can be found in the following paper:

G. Saake, S. Conrad, and C. Türker. From Object Specification towards Agent Design. In *Proc. of the 14th Int. Conf. Object-Oriented & Entity-Relationship Modelling (OO&ER'95), Gold Coast, Queensland, Australia*, Springer-Verlag, December 1995. *To appear.*

Multi-Preferential Logics for Information Systems

Pierre-Yves Schobbens (University of Namur)

Abstract

A requirement specification language for Information Systems has to rest on a rich ontology, since a requirement specification has to be checked against informal requirements. It is thus important that the natural concepts in which the informal requirements are expressed, have their formal counterpart in the specification language, so as to avoid an error-prone "encoding" of these concepts. Concepts such as encapsulation, inheritance, behaviour, time are now receiving adequate formal counterparts. Yet, a study of informal requirements shows a large use of (somewhat anthropomorphic) agency concepts, such as intention, belief, goal, notification, etc. even for automated system parts. The main characteristics of such concepts is their continuity, or "softness": a goal isn't reached or missed, but rather satisfied as much as possible. It may interact with conflicting goals to reach a compromise. It is interesting to note that such concepts are also considered as paradigms for programming. In this talk we attempt at giving them a formal counterpart.

Inheritance of Business Rules

Michael Schrefl (University of Linz)

Abstract

Life cycle rules, which define in what order work steps of a business process may be applied, and decision rules, which define which one of alternative work steps must be or may be or may not be performed in specific processing stages of a business case, are important types of business rules. Using an object-oriented modeling approach, business processes are represented by object classes, where work steps are represented by activities (methods), life cycle rules by pre- and post-states of activities, and decision rules by different types of preconditions (prohibitions, obligations, and permissions) associated with activities.

Decision rules are specialized in class hierarchies by extending (i.e. weakening) inherited prohibitions, obligations, and permissions at subclasses. Probibitions F as well as permissions P can be extended at subclasses, as contrary to the usual axiom of deontic logic $P \leftrightarrow F$, we require only $P \rightarrow F$, but not $F \rightarrow P$. A conflict resolution strategy is used to decide among conflicting rules. For example, if an activity that is permitted under some condition at a superclass is prohibited under this condition at a subclass, the permission of the superclass overrides the prohibition of the subclass. Note, this is contrary to the usual approach of overriding in class hierarchies, but corresponds to policies usually followed in business and administrative organizations.

Temporal Triggers in Active Databases Systems

Prasad A. Sistla (University of Illinois, Chicago)
joint work with Ouri Wolfson

Abstract

In this talk, we present a unified formalism, based on Temporal Logic, for specifying conditions and events in the rules for active database system. We introduce two languages FTL and PTL based on future and past temporal logics, respectively. These language permit specification of many time varying properties of database systems. They also permit specification of temporal aggregates.

We define various semantics for rules and integrity constraints with respect to two notions of time — valid time and transaction time. We also show that the temporal conditions can be used for the specification of temporal actions, which are a form of extended transactions.

In addition to the languages, we present algorithms for processing the trigger conditions specified in these languages, namely, procedures for determining when the trigger conditions are satisfied. These methods can be added as a "temporal" component to an existing database management systems.

Preliminary prototypes of the temporal component that use fragments of FTL and PTL languages have been built on top of Sybase running on SUN workstations.

Constraints in Database Models

Bernhard Thalheim (Technical University of Cottbus)

Abstract

Integrity constraints are one of the main and surely the most difficult parts of database models. High abstraction capabilities are required by the user and the modeller. Different database models use different kinds of constraints and are based on different semantics. The talk gives an overview on different semantics which are used in value-oriented and object-oriented models, surveys main classes of integrity constraints and develops an approach how type constructors, semantics and integrity constraints can be defined and used in a compatible manner. It is shown that based on this approach, the theory of integrity constraints can be inherited by other models. This approach is based on algebraic properties of constraints classes and on properties of the type constructors. According to the type constructors different structural constraints have to be considered. Based on this approach the third component of the semiotical triangle can be developed. Pragmatics aims to encorporate syntactical and semantical parts of logical theories by notions of meaning, methodologies of utilization and considerations of implementational complexity. It is demonstrated how design principles can be extended to design heuristics and complete methodologies. This leads to database design which considers all three kinds of abstraction (data abstraction, localization abstraction, implementation abstraction). Therefore, the approach takes integrity enforcement into consideration. Further, it is demonstrated by different results on scheme transformation and normalization how constraints can be applied. Finally, it is discussed how wrong assumptions - like completeness of constraint sets and normalization - centered approaches - can be corrected and it is demonstrated how this theory can be encorporated into database design tool boxes.

Fundamentals of Object-Oriented Database Systems

Bernhard Thalheim (Technical University of Cottbus) joint work with Catriel Beeri, Klaus-Dieter Schewe, Joachim Schmidt, David Stemple and Ingrid Wetzel

Abstract

Although object-oriented database systems have been implemented and object-oriented technology got a wide acceptance, formal foundations of object-oriented databases are still missing. The research summarized during this talk tries to propose a foundation for current and future object-oriented database technology. We shall present a type system whcih can be used for the definition of the structural aspects of database applications. Based on the type system classes, integrity constraints, methods, queries, views etc. can be defined since the type system can be extended in various ways, most of the known object-oriented database models can be founded in a similar way. The power of the approach is demonstrated by solutions to three problems: A) It can be shown that the major property of object-oriented database schemes – the identifiability – can be based on the properties of the type system. Weak value-responsibility is a necessary and sufficient property for identification in schemes. For object societies this notion can be accombined by 5 equivalent other notions (automorphism groups, query identifiability, ...). B) The existence of generic update operations is based on value-representability in schemes. C) A generalization of integrity-enforcing methods has been developed. This method – the generalization of greatest consistent specializations of operations – overcomes the pitfalls of rule-triggering systems like such proposed for active databases. The semantics of the presented approach is based on intuitionistic logics. Thus, the corresponding algebraic structures are topoi. The object identifier concept is a nice implementation concept which does not relieve us from unique identification of object. The designer is, therefore, responsible for defining object societies which are identifiable.

A Modal Predicate Calculus

Yde Venema (University of Amsterdam)
joint work with Maarten Marx

Abstract

Treating existential quantifiers as modal diamonds, we study the n-variable fragment L_n of first-order logic, as if it were a modal formalism. In order to deal with atomic formulas adequately, to the modal version of the language we add operators corresponding to variable substitution.

Since every modal language comes with an abstract Kripke-style semantics, this modal viewpoint on L_n provides an alternative, far more general semantics for the latter. One may impose conditions on the Kripke models, for instance approximating the standard Tarskian semantics. In this way one finds that some theorems of first-order logic are 'more valid' than others.

As an example, we define a class of generalized assignment frames called *local cubes*; here the basic idea is that only certain assignments are admissible. We show that the theory of this class is finitely axiomatizable and decidable.

A Comparison of the Notations Used in the Shlaer-Mellor Method and in TCM

Roel Wieringa (Free University of Amsterdam)
joint work with Gunter Saake (University of Magdeburg)

Abstract

In this talk we compare two notation systems for requirements model, the notations used in the Shlaer-Mellor method for object-oriented analysis and the notations uses in TCM (Toolkit for Conceptual Modeling). The notations used in the Shlaer-Mellor method are semi-formal, i.e. they consist of diagrams annotated by natural language text. The notations used in TCM are semi-formal, as in the Shlaer-Mellor notation, but there is also a formal part. The formal part of a TCM specification is written down in LCM (Language for Conceptual Modeling), a language based on order-sorted dynamic logic. The formal and semi-formal parts of a TCM specification supplement each other and each can be used without using the other. Because the semi-formal and formal notations in TCM are precisely related, the semi-formal notations have unambiguous definitions, and the formal notations have simple and clear diagram representations.

The report analyzes the notations used for information model, state model, process model and communication model used in the Shlaer-Mellor method and shows how these relate to the notations used for the class model, life cycle model and communication model in TCM. It is shown that the notations of the Shlaer-Mellor method contain ambiguities and redundancies, that are resolved when we transform these notations into TCM notations. However, some of these redundancies provide useful information and TCM is extended with a simplified form of some of the Shlaer-Mellor notations.

Working Group: Logic as a Query Language

Burkhard Freitag (University of Passau)

Abstract

It is well known that (fragments of) logic can be given an operational semantics and thus can be used as a query language. In fact, the success of relational database systems shows that logic as a query language is not merely of theoretical interest to a small community of researchers but has practical relevance.

On the other hand, however, most practitioners rather rely on conventional imperative programming and embedded SQL than adopting any of the more advanced notions developed in the field of logic databases such as views and recursion, powerful pattern matching, and integrity constraints.

Among the questions discussed in the working group have been

- Is SQL with recursive views enough from a practical point of view? Are logic query languages beyond DATALOG relevant in practice?
- If yes, what must be done to make them a success?
- Do we need a full-fledged declarative (logic) programming language with persistence?
- What higher-order features are necessary?
- Should the logic databases community care about the *implementation* of future relational database systems?

4 List of Participants

Klemens **Böhm**
GMD-IPSI-Darmstadt
Dolivostr. 15
D-64293 Darmstadt, Germany
kboehm@darmstadt.gmd.de
Phone: +49-6151-869-963
Fax: +49-6151-869-966

Stefan **Brass**
Universität Hannover
Institut für Informatik
Lange Laube 22
D-30159 Hannover, Germany
sb@informatik.uni-hannover.de
Phone: +49-511-762-4953
Fax: +49-511-762-4961

Jan **Chomicki**
Monmouth University
Howard Hall
Dept. of Computer Science
West Long Branch, NI 07764, USA
chomicki@cis.ksu.edu

Stefan **Conrad**
Universität Magdeburg
Inst. für Technische Informationssysteme
Postfach 4120
D-39016 Magdeburg, Germany
conrad@iti.cs.uni-magdeburg.de
Phone: +49-391-67-18066
Fax: +49-391-67-12020

Hendrik **Decker**
Siemens - München
Zentralabt. Forschung und Entwicklung
Corporate Research and Development
Otto-Hahn-Ring 6
D-81730 München, Germany
hendrik@zfe.siemens.de
Phone: +49-89-636-47533
Fax: +49-89-636-45111

Jürgen **Dix**
Universität Koblenz - Landau
Fachbereich Informatik
Rheinau 1
D-56075 Koblenz, Germany
dix@informatik.uni-koblenz.de
Phone: +49-261-9119-420
Fax: +49-261-9119-496

Philippe **Du Bois**
University of Namur
Facultis Universitaires
Notre Dame de la Paix
CEDITI
Av. Jean Mermoz 30
B-6041 Gosselies, Belgium
pdu@cediti.be
Phone: +32-71-259415
Fax: +32-81-724-967

Eric **Dubois**
University of Namur
Facultis Universitaires
Notre Dame de la Paix
Institut d'Informatique
Rue Grandgagnage 21
B-5000 Namur, Belgium
edu@info.fundp.ac.be
Phone: +32-81-724-986
Fax: +32-81-724-967

Hans-Dieter **Ehrich**
Technische Universität Braunschweig
Abteilung Datenbanken
Postfach 3329
D-38023 Braunschweig, Germany
HD.Ehrich@tu-bs.de
Phone: +49-531-391-7442
Fax: +49-531-391-3298

Remco **Feenstra**
Vrije Universiteit Amsterdam
Faculteit Wiskunde en Informatica
De Boelelaan 1081a
NL-1081 Amsterdam, The Netherlands
rbfeens@cs.vu.nl
Phone: +31-20-444-7742
Fax: +31-20-444-7653

Burkhard **Freitag**
Universität Passau
Fakultät für Mathematik u. Informatik
D-94030 Passau, Germany
freitag@fmi.uni-passau.de
Phone: +49-851 509-3130
Fax: +49-851 509-3092

Martin **Gogolla**
Universität Bremen
Fachbereich 3
AG Datenbanksysteme
Postfach 330440
D-28359 Bremen, Germany
gogolla@informatik.uni-bremen.de
Phone: +49-421-218-7469
Fax: +49-421-218-4269

Peter **Ladkin**
Universität Bielefeld
Technische Fakultät
Postfach 100131
D-33501 Bielefeld, Germany
ladkin@techfak.uni-bielefeld.de
Phone: +49-521-106-2951
Fax: +49-521-106-2962

Udo **Lipeck**
Universität Hannover
Institut für Informatik
Lange Laube 22
D-30159 Hannover, Germany
ul@informatik.uni-hannover.de
Phone: +49-511-762-4950
Fax: +49-511-762-4961

Bertram **Ludäscher**
Universität Freiburg
Institut für Informatik
Am Flughafen 17
D-79110 Freiburg, Germany
ludaesch@informatik.uni-freiburg.de
Phone: +49-761-203-8124
Fax: +49-761-203-8122

Robert **Marti**
ETH Zürich
Department of Computer Science
CH-8092 Zürich, Switzerland
marti@inf.ethz.ch
Phone: +41-1-632-7260
Fax: +41-1-632-1172

John-Jules **Meyer**
University of Utrecht
Department of Computer Science
De Uithof
Padualaan 14, Postbus 80.089
NL-3508 TB, Utrecht, The Netherlands
jj@cs.ruu.nl
Phone: +31-30-534117
Fax: +31-30-513791

Barbara **Paech**
Technische Universität München
Institut für Informatik
Arcisstr. 21
D-80290 München, Germany
paech@informatik.tu-muenchen.de
Phone: +49-89-2105-8186
Fax: +49-89-2105-8183

Mark **Ryan**
University of Birmingham
School of Computer Science
Edgbaston, Birmingham, B15 2TT, UK
M.D.Ryan@cs.bham.ac.uk
Phone: +44-121-414-73 61
Fax: +44-121-414-42 81

Gunter **Saake**
Universität - Magdeburg
Inst. für Technische Informationssysteme
Postfach 4120
D-39016 Magdeburg, Germany

`saake@iti.cs.uni-magdeburg.de`
Phone: `+49-391-67-18800`
Fax: `+49-391-67-12020`

Pierre-Yves **Schobbens**
University of Namur
Facultis Universitaires
Notre Dame de la Paix
Institut d'Informatique
Rue Grandgagnage 21
B-5000 Namur, Belgium

`pys@info.fundp.ac.be`
Phone: `+32-81-724-990`
Fax: `+32-81-724-967`

Michael **Schrefl**
Universität Linz
Institut für Wirtschaftsinformatik
Abt. Data and Knowledge Engineering
Altenbergerstr. 69
A-4040 Linz, Austria

`schrefl@dke.uni-linz.ac.at`
Phone: `+43-732-2468-9490`
Fax: `+43-732-2468-9471`

Prasad A. **Sistla**
University of Illinois at Chicago
Dept. of Electrical Engineering &
Computer Science
851 South Morgan Street
Chicago, IL 60607, USA

`sistla@surya.eecs.uic.edu`
Fax: `+1-312-413-0024`
Phone: `+1-312-996-8779`

Bernhard **Thalheim**
Technische Universität Cottbus
Institut für Informatik
Postfach 101344
D-03013 Cottbus, Germany

`thalheim@informatik.tu-cottbus.de`
Phone: `+49-355-692700`

Can **Türker**
Universität - Magdeburg
Inst. für Technische Informationssysteme
Postfach 4120
D-39016 Magdeburg, Germany

`tuerker@iti.cs.uni-madgeburg.de`
Phone: `+49-391-67-12994`
Fax: `+49-391-67-12020`

Yde **Venema**
Vrije Universiteit Amsterdam
Faculteit Wiskunde en Informatica
De Boelelaan 1081a
NL-1081 Amsterdam, The Netherlands

`yde@cs.vu.nl`
Phone: `+31-20-444-7735`
Fax: `+31-20-444-7653`

Roel **Wieringa**
Vrije Universiteit Amsterdam
Faculteit Wiskunde en Informatica
De Boelelaan 1081a,
NL-1081 Amsterdam, The Netherlands

`roelw@cs.vu.nl`
Phone: `+31-20-444-7771`
Fax: `+31-20-444-7653`