Foundations of Semistructured Data

2.9.2001-7.9.2001

organized by

Alberto Mendelzon (Toronto), Thomas Schwentick (Marburg), Dan Suciu (Washington)

Traditional database systems rely on an old model: the relational data model. When it was proposed in the early 1970's by Codd, a logician, the relational model generated a true revolution in data management. In this simple model data is represented as relations in first order structures and queries as first order logic formulas. It enabled researchers and implementors to separate the logical aspect of the data from its physical implementation. Thirty years of research and development followed, and they led to today's mature and highly performant relational database systems.
The age of the Internet brought new data management applications and challenges. Data is now accessed over the Web, and is available in a variety of formats, including HTML, XML, as well as several application specific data formats. Often data is mixed with free text, and the boundary between data and text is sometimes blurred. The way the data can be retrieved also varies considerably: some instances can be downloaded entirely, others can only be accessed through limited capabilities. To accommodate all forms and kinds of data, the database research community has introduced the "semistructured data model", where data is self-describing, irregular, and graph-like. The new model captures naturally Web data, such as HTML, XML, or other application specific formats. While researchers mostly agree on a common definition of the semistructured data, there is still a lot of confusion about the logical foundations for representing and querying such data: several practical query languages have been proposed, but their formal foundations and their relationships to logical formalisms are poorly understood. This lack of understanding further prevents us from designing general solutions to typical data management problems, such as building indexes, optimizing queries, and designing storage structures. To add to the confusion, the structured document community has studied for several years "structured text", and proposed a number of algebraic operators and accompanying index structures to express queries over structured text. This work definitely has relevance to semistructured data, but their connections are still poorly understood. Current work in academia and research institutions is studying the nature of query languages for semistructured data, and proposing index structures, optimization techniques, and storage mechanisms to support those queries.

This seminar aims at bringing together database researchers, logicians, and researchers in structured documents. Furthermore, we would like to invite some people from other communities that are related to the area of semistructured data, like information retrieval, programming languages, and discrete algorithms. Besides the presentation of recent research results by the participants additional goals are:

- to identify the main issues for further foundational research on semistructured data,
- to improve the mutual understanding of the communities involved concerning their respective settings and needs.

The seminar will consist of presentations by the participants and of organized group discussions.

Contents

1 Simple First-Order Tree Query Languages

Michael Benedikt and Leonid Libkin

We present a family of straightforward extensions of the relational calculus to the context of relations on trees, which yields tree query languages with attractive closure properties. These languages share the basic property that the output of a query is always representable as an automaton running over a tuple of trees. We focus on two of these languages, giving a normal form for queries in each language and deriving from these normal forms bounds on expressive and complexity of queries. We also show that one can effectively enumerate the queries with finite output within these languages

2 Q&N: Combining Querying and Navigation on Structured Document Collections

Holger Meuss and Klaus Schulz

The use of markup languages like SGML, HTML, or XML for encoding the structure of documents or linguistic data has lead to many databases where entries are adequately described as trees. In this context querying formalisms are interesting that offer the possibility to refer both to textual content and logical structure. We consider models where the structure specified in a query is not only used as a filter, but also for selecting and presenting different parts of the data. If answers are formalized as mappings from query nodes to the database, a simple enumeration of all mappings in the answer set will often suffer from the effect that many answers have common subparts. From a theoretical point of view this may lead to an exponential time complexity of the computation and presentation of all answers. Concentrating on the language of so-called tree queries—a variant and extension of Kilpeläinen's Tree Matching formalism—we introduce the notion of a "complete answer aggregate" for a given query. This new data structure offers a compact view of the set of all answers and supports active exploration of the answer space. Since complete answer aggregates use a powerful structure sharing mechanism their maximal size is of order $\mathcal{O}(d \cdot h \cdot q)$ where d and q respectively denote the size of the database and the query, and h is the maximal depth of a path of the database. Complete answer aggregates can be computed in time

$\mathcal{O}(d \cdot log(d) \cdot h \cdot q)$.
We will show in examples that complete answer aggregates are particularly appropriate for answer searching and answer browsing, thus supporting the user in the task of localizing the relevant pieces of information in a complex environment as represented by a semistructured data.

3 An Example-based Approach to Extracting Semistructured Data

Alberto H. F. Laender and Altigran S. da Silva and Berthier Ribeiro-Neto

We present DEByE - Data Extraction By Example, an approach to extracting data from Web sources, based on a small set of examples specified by the user. The novelty of this approach is the fact that the user specifies examples according to a structure of his liking and that this structure is described at example specification time. For the specification of the examples, the user interacts with a tool which adopts nested tables as its visual paradigm. Nested tables are simple, intuitive, and allow shielding the user from technical details (such as HTML tags, formatting operators, and learning automata) related to the extraction problem. The examples provided by the user are then used to generate patterns which allow extracting data from new documents. For the extraction, DEByE adopts a new bottom-up procedure we proposed which is very effective with various Web sources, as demonstrated by our experiments.

4 Information Monitoring on the Internet: The Continual Query Approach

Ling Liu

The World Wide Web (the Web) has made an enormous amount of data freely accessible over the Internet. However, finding the right information in the midst of this mountain of data has been likened to finding the proverbial needle in a haystack. The exponential growth of the Web is increasing the haystack rapidly. Commonly used search engines (e.g., AltaVista) and directory services (e.g., Yahoo) have practical but limited success in providing continual services. Instead of pull-based browsing, *information monitoring* is a promising area of research where the system brings the right information to the right user at the right time. In this talk I motivate the problems of information monitoring on the net with a number of applications. Then I present an overview of the continual query approach and the technical results produced from our research, including the concept of continual queries, the three continual query projects: OpenCQ [3,2], WebCQ [1,4], and Peer-to-Peer based information monitoring (PeerWatch). My talk ends with a discussion on a list of interesting research issues and challenges in monitoring information changes on the Net.

References

[1] D. Buttler, L. Liu, and C. Pu. A fully automated object extraction system for the world wide web. *Proceedings of IEEE International Conference on Distributed Computing Systems*, April 2001

[2] L. Liu, C. Pu, and W. Han. XWrap: An XML-enabled Wrapper Construction System for Web Information Sources. *Proceedings of the International Conference on Data Engineering*, 2000

[3] L. Liu, C. Pu, and W. Tang. Continual queries for internet-scale event-driven information delivery. *IEEE Knowledge and Data Engineering*, 1999. Special Issue on Web Technology.

[4] L. Liu, C. Pu, and W. Tang. WebCQ: Detecting and Notifying Information Changes on the Web. *Inernational Conference on Knowledge and Information Management*, 2000

[5] L. Liu, C. Pu, W. Tang, J. Biggs, D. Buttler, W. Han, P. Benninghoff, and Fenghua. CQ: A Personalized Update Monitoring Toolkit. In *Proceedings of ACM SIGMOD Conference*, 1998

5 Data Provenance

Peter Buneman and Sanjeev Khanna and Wang-Chiew Tan

When you find some data on the Web, are you concerned about out how it got there? Most of us have learned not to put much faith in what we find when we casually browse the Web. But if you are a scientist or any kind of scholar, you would like to have confidence in the accuracy and timeliness of the data that you

find. In particular, you would like to know how it got there – its provenance. In Bioinformatics there are literally hundreds of public databases. Most of these are not source data: their contents have been extracted from other databases by a process of filtering, transforming, and manual correction and annotation. Thus, describing the provenance of some piece of data is a complex issue. These "curated" databases have enormous added value, yet they often fail to keep an adequate description of the provenance of the data they contain.
In this talk I shall describe some recent work on a Digital Libraries project to investigate data provenance. I shall try to describe the general problem and then deal with some technical issues that have arisen, including (a) a semistructured data model that may help in characterizing data provenance (b) tracing provenance through database queries (c) efficient archiving of scientific databases and (d) keys (canonical identifiers) for semistructured data and XML.

6 Containment of Tree Pattern Queries

Gerome Miklau and Dan Suciu

We describe a class of pattern queries, over node-labeled trees, which contain label wildcards and can express descendant relationships between nodes. This class of patterns is equivalent to a fragment of XPath and a fragment of computation tree logic. We study the problem of deciding containment of patterns. If either the label wildcards or descendant edges are missing from such patterns containment is decidable in polynomial time. However, we show that the combination of these features makes the containment problem coNP-complete. We provide an algorithm that runs in exponential time in the size of the patterns, and show that when the number of label wildcards and/or descendant edges are bounded the algorithm runs in polynomial time.

7 Semistuctured Data and Linguistics

Frank Morawietz and Uwe Moennich

In this talk we show that the datastructures used by linguists can be considered as semi-structured data. In particular, we show how to encode a Tree Adjoining Grammar [1] into a weakly equivalent monadic context-free tree grammar (MCFTG). This is particulary interesting since the resulting string languages have a descriptive complexity which is non-context-free. By viewing MCFTG-rules as terms in a free Lawvere theory, we can translate a given MCFTG into a regular tree grammar. The latter is characterizable by both a tree automaton and a corresponding formula in monadic second-order (MSO) logic. The trees of the resulting regular tree language are then unpacked into the intended "linguistic" trees through an MSO transduction based upon tree-walking automata. This two-step approach gives a logical as well as an operational description of the tree sets involved.

References

[1] Aravind K. Joshi and Yves Schabes. Tree Adjoining Grammars. In *Handbook of Formal Languages*, Springer 1997.

8 From XML Schema to Relations: A Cost-Based Approach to XML Storage

Juliana Freire

XML has become an important medium for data representation, particularly when that data is exchanged over or browsed on the Internet. As the volume of XML data increases, there is a growing interest in storing XML in relational databases so that the well-developed features of these systems (e.g., concurrency control, crash recovery, query processors) can be re-used. However, given the wide variety of XML applications and the mismatch between XML's nested-tree structure and the flat tuples of the relational model, storing XML documents in relational databases presents interesting challenges.
LegoDB is a *cost-based* XML-to-relational mapping engine that addresses this problem. It explores a space of possible mappings and selects the best mapping for a given application (defined by an XML Schema, XML data statistics, and an XML query workload). LegoDB leverages existing XML and relational technologies: it represents the target application using XML standards and constructs the space of configurations using XML-specific operations, and it uses a traditional relational optimizer to obtain accurate cost estimates for the derived configurations. In this talk, I will describe the LegoDB mapping engine and provide

experimental results that demonstrate the effectiveness of this approach.

9 XFiles: A Grammatical Approach to XML Engineering

Anne Brüggemann-Klein and Derick Wood

XFiles project provides means to deal with the large variety of grammars that are in use for language specifications on the Web. Examples are: The XML grammar, a metagrammar for the XML grammar and other grammars on the Web, XML DTDs and schemas, XML DTDs and Quasi-DTDs with complex semantics such as XSLT, XSL-FO, XML Schema and XLink, the grammar for CSS, the grammar for Minimal XML, and the grammar for Relax NG schemas.
Most of these grammars are extended context-free grammars. In the XFiles project, we have defined the notion of extended context-free eLL(1) grammars and we have developed an algorithm that builds a parsing table for eLL(1) grammars for efficient parsing. These foundations of the project are based on—among others—Heckmann's work on extended context-free grammars and our own work on 1-unambiguous regular expressions.
We are currently designing and implementing a software component ECFG that lets us parse encodings of extended context-free grammars into an object representation, compute parsing tables and generate parses and that provides access to the parse tree for further processing. A metagrammar for encodings of extended context-free grammars facilitates the bootstrapping of ECFG. Furthermore, ECFG can be used as a parser generator for classes of grammars whose encoding form context-free languages.
ECFG is an enabling technology to application programmers who deal with Web languages. It encapsulates proven techniques to deal with the syntactical aspects of language parsing, leaving developers to deal with application-specific processing on a high level of representation of the languages they are dealing with.
We envision the following first uses of ECFG:

1. Parse the XML grammar, analyze it, simplify it and generate parsers for XML DTDs and XML instances.

2. Build a tool that reads in a (possibly empty or incomplete) DTD and generates a nonvalidating parser and a nonvalidating parser for the "instances" of the DTD.

3. Build tools for analysis and manipulation of DTDs.

As asides, I have also mentioned in the talk a mathematical data model for XML (PODDP 2000, Munich, Germany) and a result on two-way tree automata over unranked alphabets (CIAA 2000, London, Ontario). Online versions of these papers as well as the synopsis on tree automata over unranked alphabets (with Makoto Murata and Derick Wood) are available from Derick Wood's Web site `http://www.cs.ust.hk/~dwood`.

10 Stream Semantics (for Semistructured Data)

David Maier and Peter Tucker and Kristin Tufte

Data streams arise in many contexts – financial feeds, telephone call records, network monitoring, environmental sensing. We would like to adapt database query processing techniques to work with streams. The situation at first glance seems good, as many query evaluators actually operate in a pipelined fashion. Any monotone algebraic operator potentially has a non-blocking, stream-based version. However, monotonicity depends on what definition of "greater than" one takes on operator inputs. There are several possible definitions one could use when generalizing from flat data to structures found in semi-structured models, and the choice affects which operators are monotone. (For example, nest is not monotone under "subset" order, but it is under "substructure" order.) The definition in turn influences stream semantics in that it governs what is the "reconstitution" function that maps a stream back to its tabular or document form. One part of our work looks at a merge operator (similar to the "deep union" of others) that recursively combines XML fragments into larger documents. Merge makes use of a merge template that specifies when to combine elements and what combination function to use on their contents. We hypothesize that merge is a lattice theoretic join operator on the subspace of documents that satisfy keys implicit in the merge template.
Even with a broader notion of monotone operator, there are still blocking operators (e.g., group-by) and non-blocking operators that accumulate unbounded amounts of internal state (e.g., join). A second part of our work addresses such limitations by supplementing stream semantics with punctuation. A punctuation can be viewed as a predicate; its appearance in a stream indicates that no more data matching the predicate will be seen on the stream. With punctuated streams, an operator can be modified to partially unblock its output or discard some of its internal state early. We characterize how operators work with punctuation using three kinds of rules:

- Pass rules specify when output can be released based on punctuation seen so far.
- Purge rules govern the discarding of state.
- Propagate rules specify when an operator can emit punctuation to be used by other operators.

11 Automata for Querying XML Documents

Helmut Seidl and Andreas Neumann and Alexandru Berlea

The goal of this research is to design and implement an XML querying tool which supports an as expressive language of query patterns as possible — which still can be implemented with "decent" efficiency. Here, "decent" efficiency means that querying should be essentially no more exprensive than parsing.
In order to achieve this goal, we apply classical automata theory on unranked trees. We present a notion of alternation in grammars and an extension of finite tree automata by pushdowns which allows to locate matches of full regular patterns (both for context and structure) in at most two passes over the document. Furthermore, we briefly sketch practical implementation issues such as dealing with don't cares, external predicates, say for text nodes, and state explosion. The outcome of this research is the very efficient querying tool Fxgrep which can be downloaded from:

`http://www.informatik.uni-trier.de/~berlea/Fxgrep`

12 Fxt - A Transformation Tool for XML Documents

Alexandru Berlea

XML document processing is a subarea of tree processing for which the functional programming style is very natural. A pattern-matcher is necessary for identifying parts of the tree to be processed. The functional style implies a processing model in which navigation is possible only to subtrees of a tree. This restriction can be compensated if a tree pattern-matcher is used which is able to relate to ancestors as well as to siblings of a match. On top of the powerful fxgrep XML pattern-matcher, we build fxt, a transformation tool for XML documents. The functional processing model that fxt uses, allows an implementation much more efficient than implementations permitted by the processing model of the popular XSLT, where navigation in the input tree can proceed in arbitrary directions. The fxt transformations are specified in an intuitive, declarative way. Flexibility is provided by hooks to the full functionality of the SML programming language, as well as by the fxt's variable mechanism.

13 Visual Web Information Extraction with Lixto

Georg Gottlob and Robert Baumgartner and Sergio Flesca

We present and demonstrate new techniques for supervised wrapper generation and a system called *Lixto* implementing these techniques, in particular for generating wrappers which translate relevant pieces of HTML files into XML. One core component of this effort is a new declarative, logic-based language called *ELOG* that offers quite advanced functionality. *Lixto*, of which an entirely functional prototype has been implemented, assists the user to semi-automatically create *ELOG* wrapper programs by providing a fully visual and interactive user interface. Users never have to deal with the internal language and even familiarity with HTML is not required. Moreover, *Lixto* is portable, allows for expressive and flexible data extraction and uses intuitive hierarchical extraction, as well as string extraction techniques. *Lixto* can be used to create an "XML-Companion" for an HTML web page with changing content, containing the continually updated XML translation of the relevant information. Empirical results emphasise that our wrapper generator has been successfully tested in real life scenarios.
We also explain and demonstrate advanced features of Lixto such as the possinbility to crawl the web while wrapping, composing XML data structures that integrate data from several web pages, and recursive wrapping. Finally, we exhibit a complex application done for a telecom company which uses Lixto to wrap and integrate data from the web and presenting the data on a PDA with a wireless connection to the internet.

14 Integration of Data and Web Services

Serge Abiteboul and Omar Benjelloun and Tova Milo

We presented a data-oriented approach to web services integration. To this end, we introduced a new framework based on a model that embeds service calls within semistructured data. The framework captures various integration scenarios including mediation and warehousing, and provides fusion based on object identifiers. Moreover, by allowing service call parameters and responses to contain calls to other services, the framework enables distributed computation over the web. We described our ongoing work on the subject, mainly through examples.

15 Finite State Machines for Infinite Alphabets

Frank Neven

Motivated by formal models recently proposed in the cot of XML, we study automata and logics on strings over infinite alphabets. These are conservative extensions of classical automata and logics defining the regular languages on finite alphabets. Specifically, we consider register and pebble automata, and extensions of first-order logic and monadic second-order logic. For each type of automaton we consider one-way and two-way variants, as well as deterministic, non-deterministic, and alternating control. We investigate the expressiveness and complexity of the automata, their connection to the logics, as well as standard decision problems

16 The XML Query Formal Semantics: State and Challenges

Peter Fankhauser

The XML Query Formal Semantics [1] is designed as part of the W3C-Activity

XML Query [2] to provide a formal basis for an XQuery. It is guided by three principles. Full support of wellformed and valid XML, compositionality, and strong typing. This talk introduces the overall design, the underlying datamodel, the type system, and the operational semantics of the algebra. Furthermore, the talk exemplifies how to map surface syntaxes such as XPath [3] or XQuery to the algebra, and discusses the relationship between the algebra's typesystem and XML Schema's [4,5,6] typesystem.

References

[1] `http://www.w3.org/TR/query-algebra/`
[1] `http://www.w3.org/XML/Query`
[1] `http://www.w3.org/TR/xpath.html`
[1] `http://www.w3.org/TR/xmlschema-0/`
[1] `http://www.w3.org/TR/xmlschema-1/`
[1] `http://www.w3.org/TR/xmlschema-2/`

17 Grouping Constructs for Semistructured Data

Franois Bry and Dan Olteanu and Sebastian Schaffert

Markup languages for semistructured data like XML are of growing importance as means for data exchange and storage. In this paper we propose an enhancement for the semistructured data model that allows to express more semantics and to enhance query answering. A data model is proposed and the implications on pattern matching are investigated.
Paper:
`www.pms.informatik.uni-muenchen.de/publikationen/PMS-FB/PMS-FB-2001-7.pdf`
Slides:
`www.pms.informatik.uni-muenchen.de/publikationen/PMS-FB/PMS-FB-2001-7/slides.pdf`

18 A Query Language Based on the Ambient Logic

Giorgio Ghelli and Luca Cardelli

The ambient logic is a modal logic proposed to describe the structural and computational properties of distributed and mobile computation. The structural part of the ambient logic is, essentially, a logic of labeled trees, hence it turns out to be a good foundation for query languages for semistructured data, much in the same way as first order logic is a fitting foundation for relational query languages. We present a query language for semistructured data that is based on the ambient logic, and we outline an execution model for this language. The language turns out to be quite expressive. Its strong foundations and the equivalences that hold in the ambient logic are helpful in the definition of the language semantics and execution model.

19 A Logic for Trees

Laks V.S. Lakshmanan and Alberto Mendelzon

In recent times, there has been a significant spurt of interest in trees in the database community, much of it sparked by the advent of modern applications such as LDAP style network directories and XML. In this talk, I will outline a generic model for tree structured data and describe a logic for reasoning about such trees. The vision that is driving this work is a "wide spectrum" logic for accommodating multiple data stores such as relational databases, network directories, and XML repositories. I will describe both the syntax and semantics and illustrate the logic with a number of examples. The Horn clause fragment of this logic serves as a declararive query language for tree structured data. Finally, I will conclude with open problems.

20 Queries under Compression

Hans Leiß and Michel de Rougemont

To save storage space, data are often archived in compressed form. We considered the problem of answering queries about archived data in situations where decompression is not an option.
Specifically, we considered the Lempel-Ziv-78 scheme for compression of strings. We think of uncompressed strings as finite linear orders with unary predicates P_a telling at which positions the letter a occurs. LZ-compressed strings are enumerated tries, i.e. finite trees in which different outgoing edges of a node carry different letters a and whose nodes are numbered in a order-compatible way. We model these as total orders with a parent-node-relation and unary predicates P_a telling which node is an a-child of its parent.
Thus, a string w and its compression $LZ(w)$ are relational structures (of different signatures). As query languages, we consider formulas $\phi(x, X)$ of first- or higher-order logic for such structures.
We can code a position i in w by a pair (i_1, i_2) in $LZ(w)$ and prove: (i) Not every first-order query about strings w can be translated to an equivalent first-order query about $LZ(w)$, and conversely. (ii) Every first-order query about w can be translated to an equivalent query about $LZ(w)$ in first-order extended by a (deterministic) transitive closure operator. (iii) Every monadic second order (i.e. regular) property of strings can be translated to a dyadic second order property of LZ-compressed strings.
Finally, we looked for a variation of tree-automata that can be used to evaluate monadic second-order properties ϕ of LZ-compressed strings. We showed that the k-MSO-equivalence of enumerated tries is not a congruence with respect to concatenation of LZ-compressed strings, so there is no notion of LZ-automaton that captures monadic second-order of enumerated tries. However, there is an approximate notion of automaton whose transitions take (in)equations between the children of a node and its successor in the enumeration into account.

21 View-Based Query Processing for Variants of Two-Way Regular Path Queries

Diego Calvanese and Maurizio Lenzerini

View-based query processing is the problem of computing the answer to a query based on a set of materialized views, rather than on the raw data in the database. The problem comes in two different forms, called query rewriting (QR) and query answering (QA), respectively. In the first form, we are given a query and a set of view definitions, and the goal is to reformulate the query into an expression that refers only to the views. In the second form, besides the query and the view definitions, we are also given the extensions of the views and a tuple, and the goal is to check whether the knowledge on the view extensions logically implies that the tuple satisfies the query.
In this talk we concentrate on the problem of view-based query answering in the context of semistructured data, in particular for the case of variants of two-way regular-path queries (2RPQs), where one can specify traversal of the edges of the database in both directions. We present techniques based on the use of two-way finite-state automata, which allow us to prove optimal upper bounds for the problem. More precisely, we show that for 2RPQs, QA is PSPACE-complete wrt combined complexity and coNP-complete wrt data complexity. For conjunctions of 2RPQs, QA is EXPSPACE-complete wrt combined complexity. Using a technique based on alternating two-way automata we are also able to show that QA is still in PSPACE when the conjunctive 2RPQ is tree-shaped.

22 Optimizing Queries Using a Meta-level Database

Christoph Koch

This talk addresses the problem of optimizing queries over an object-oriented data model extended by mechanisms for explicitly representing meta-data. Data are stored using a schema with several layers, which describe each other by the use of *meta-classes* and *homomorphisms* between relationships between classes on the various layers. We characterize a fragment of algebraic queries over our data model, the *described queries*, for which a simple one-to-one relationship with *description queries* over meta-data exists such that each tuple in a described query corresponds to exacly one tuple in the corresponding description query – with the nice property that the describing tuple holds all of the meta-data of the described tuple. We show how this can be used for semantic query optimization and provide an algorithm in the case of conjunctive queries that is based on the Chase, having favorable optimality properties. Next, we discuss recursive and regular path queries, where our approach is closely related to and a generalization of well-known work based on *graph simulation*. Finally, we also put our work into the context of semistructured data management research and propose it as a new approach towards adding structure to semistructured data.

23 Chase & Backchase: A Versatile Tool for Optimizing Queries of all Kinds

Val Tannen and Alin Deutsch and Lucian Popa and Arnaud Sahuguet

We have previously proposed *chase and backchase* as a novel method for using materialized views and integrity constraints in query optimization. In this presentation we show that the method is widely usable by extending it to bag and mixed (i.e. bag-set) semantics as well as to grouping views and by showing how to integrate it with standard cost-based optimization and by extending it to XML queries.
We understand materialized views broadly, including user-defined views, cached queries and physical access structures (such as join indexes, access support relations, and gmaps). Chase and backchase supports a very general class of integrity constraints, thus being able to find execution plans using views that do not fall in the scope of other methods. In fact, we prove completeness theorems that show that our method will find the best plan in the presence of common and practically important classes of constraints and views, even when bag and set semantics are mixed, as well as when queries are extended with regular path features. We report on a series of experiments that demonstrate the practicality of our new ideas.

24 Polymorphic Queries and Polymorphic Query Languages

Alberto Mendelzon and Gösta Grahne and Laks Lakshmanan

Motivated by information integration and semistructured data applications, we investigate the foundations of polymorphic queries. We define the *uniformly polymorphic queries* that in a certain sense compute the same query on databases with many different schemas. We introduce the notion of *meta-schema*: classical schemas correspond to "instances" of meta-schemas. We propose two polymorphic query languages; a rule-based language called *Polylog* and a polymorphic algebra. We give a natural definition of first-order uniformly polymorphic queries and show that Polylog and the polymorphic algebra are equivalent and complete for this class.

25 Using Agents for Concurrent Querying of Web-like Databases via a Hyper-Set-Theoretic Approach

Vladimir Sazonov

The aim of this talk is to present a brief outline of a further step in the ongoing work concerning the hyper-set-theoretic approach to (unstructured) distributed Web-like databases and corresponding query language Δ. The novel idea in this approach consists in using dynamically created mobile agents (processes) for more efficient querying such databases by exploiting concurrently distributed computational resources, potentially over the whole Internet. A fragment of a calculus of querying agents based on Δ is presented. A corresponding paper (for PSI'01) will be accessible via `http://www.csc.liv.ac.uk/~sazonov`.

26 A Graph-Based Update Language for Object-Oriented Data Models

Jan Hidders

We present a graph-based data model GDM where database instances and database schemas are described by certain types of labeled graphs called instance graphs and schema graphs. For this data model we introduce an update language GUL that is based on pattern matching and whose operations are an addition and a deletion that are represented in a graphical way. For this language it is investigated if operations can be typed such that it is guaranteed for well-typed operations that the result belongs to a certain database schema graph, and what the complexity of deciding this notion well-typedness is. Finally, we characterize the expressive power of GUL and compare it to other graph manipulation languages.

27 Streaming XML data

Luc Segoufin

An XML document is viewed as a stream of opening and closing tags. We study the scenario where queries over streams of XML data have to be answered using a constant memory (that is depending on the query only and not on the data). Obviously with those restrictions it is no longer possible to answer many queries. Our goal is to analyze what is still possible to do. We illustrated this by giving a characterization of DTDs that can be validated in this scenario.

28 A Formal Analysis of LDAP

Fang Wei

LDAP (Lightweight Directory Access Protocol) directories are being widely used on the Web, for white pages information, user profiles, etc. The advantages LDAP offers are (i) the support for highly distributed data on the Web while still keeping a uniform data model; (ii) the flexibility of a semi-structured data model, i.e. a flexible data type definition enabling the presentation and manipulation of heterogeneous data entries in a natural manner. Although many implementations of the LDAP protocol exist, the still lacking logical formalization prohibits a formal analysis and makes it difficult to make use of the numerous results developed for relational databases. In this talk, we give a first-order logic semantics of LDAP and discuss the expressive power of LDAP. In particular, schema typing constraints are interpreted as semantic integrity constraints. We apply our framework to the containment problem of LDAP queries with schema constraints; we reduce this problem to the containment problem of Datalog in the presence of integrity constraints.

29 A Reformulation of the XDuce Type System

Jan Van den Bussche and Stijn Vansummeren

XDuce, introduced by Hosoya and Pierce, is a very interesting n ML-like programming language for computing with XML data. The XDuce type system features type inference of pattern variables. Weak points of the type system are that it uses a grammar-based formalism, necessitating a difficult well-formedness condition; that the type inference algorithm is formulated on the level of an encoding in binary trees, which makes it hard to understand the algorithm and prove it correct; and that type inference is complete only for variables in tail position. We offer a reformulation of the XDuce type system, entirely based on unranked hedge automata, that avoids these weak points.