Dagstuhl Workshop on Finite Model Theory, Databases, and Computer Aided Verification

03.10. - 08.10.1999

organized by

G. Gottlob, E. Grädel, M. Vardi, V. Vianu

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

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

Also the central logical formalisms in the three fields are of a very similar nature. Typically, a basic formalism like first-order logic, relational calculus or modal logic is extended by recursion in one form or another. In particular, fixed-point logics (formalisms that include least and/or greatest fixed points as their essential feature) play a central role in all three fields. In databases, fixed-point and while queries have been studied quite intensively and fixed-point query languages such Datalog and its extensions are central to the field. In CAV, the mu-calculus is in some sense the quintessential specification language, since it subsumes most of the other common formalisms like PDL, CTL, CTL* etc. The discovery of natural symbolic evaluation of the mu-calculus has lead to the industrial acceptance of computer-aided verification. In FMT, the most important logics are the fixed-point logics LFP, IFP, PFP with a very close relationship to the most important complexity classes. Hence fixed point logics, their expressive power and the algorithmic problems connected with them have been a central topic of this workshop.

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

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

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

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

We believe that this workshop has been a success. It has certainly helped to increase the awareness of the researchers working in one field of the problems and methods in the others and thus to increase the interaction and collaboration of the three fields, and the transfer of methodologies from one field to another.

Georg Gottlob
Erich Grädel
Moshe Vardi
Victor Vianu

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

Contents

1 **Martin Otto:** *Finite Model Theory* **6**

2 **Phokion G. Kolaitis:** *Database Query Languages - a finite model theory perspective* **6**

3 **Jan Van den Bussche:** *Constraint Databases* **7**

4 **Colin Stirling:** *Games in Verification* **7**

5 **Pierre Wolper:** *Infinite Structures in Databases and Verification* **8**

6 **Foto Afrati:** *CTL* vs. Monadic Datalog* **8**

7 **Michael Benedikt:** *Logical methods in pointer aliasing analysis* **9**

8 **Nicole Bidoit:** *Implicit temporal query languages : towards completeness* **9**

9 **Witold Charatonik:** *Stratified mu-calculus* **11**

10 **Rolf Drechsler:** *Automatic verification based on Decision Diagrams* **12**

11 **E. Allen Emerson:** *Aspects of Model Checking* **13**

12 **Javier Esparza:** *An automata-theoretic approach to interprocedural dataflow analysis* **13**

13 **Kathi Fisler:** *Computing Bisimulation Symbolically* **14**

14 **Erich Grädel:** *Automatic Structures* **14**

15 **Martin Grohe:** *Descriptive and Parameterized Complexity* **15**

16 **Neil Immerman:** *Reachability Logic* **16**

17 **Marcin Jurdziński:** *Small Progress Measures and Model Checking Games* **17**

18 **Leonid Libkin:** *Variable Independence* **17**

19 Johann A. Makowsky: *Meta-Finite Monadic Second Order Logic* **18**

20 Frank Neven: *Unranked trees in database theory* **18**

21 Andreas Podelski: *Temporal Properties as Models of Constraint Data Bases* **19**

22 Michel de Rougemont: *Probabilistic Model Checking and Compression* **20**

23 Vladimir Sazonov: *Capturing LOGSPACE over Hereditarily-Finite Sets* **21**

24 Helmut Seidl: *Inter-Procedural Analysis of Parallel Programs - the Abstract Interpretation Perspective* **21**

25 Wolfgang Thomas: *Recognizability and Constraint Satisfaction* **22**

26 Moshe Y. Vardi: *Emptiness of Tree Automata* **23**

27 Helmut Veith: *Linear Time Datalog: Temporal versus deductive reasoning in verification* **23**

28 Victor Vianu: *Verifying protocols for electronic commerce* **24**

29 Igor Walukiewicz: *Logic on Traces* **24**

30 Thomas Wilke: *CTL^+ Is Exponentially More Succinct Than CTL* **25**

1 Finite Model Theory

Martin Otto

This is a survey talk on some of the central issues in finite model theory. I focus on the role of games and logical equivalences, and on ideas and results in descriptive complexity. The discussion of games and equivalences is related to some families of logics that feature prominently in finite model theory, in particular finite-variable logics and fixed-point extensions of first-order logic. Fundamental notions od equivalence, as mediated by the games, may also be regarded as providing specif semantic frameworks (e.g. of a modal, bounded-variable, or guarded flavour) which reflect levels of abstraction at which finite structures are to represent typical problem instances in corresponding applications. Of the algorithmic and model theoretic issues that can usefully be studied within these specific semantic frameworks I here consider the capturing of complexity classes and model-checking issues as well as the analysis of least-fixed point recursion.
A revised and extended version of this talk will be available from my Aachen homepage: www-mgi.informatik.rwth-aachen.de/∼otto

2 Database Query Languages - a finite model theory perspective

Phokion G. Kolaitis

The aim of this talk was to present an overview of database query languages from a finite model theory. Particular emphasis was placed on the trade-off between expressive power and complexity of query evaluation.
The first part of the talk focused on first-order logic as a database query languages and on fragments of first-order logic, such as conjunctive queries. The second part of talk contained a survey of extensions of first-order logic

with fixed-point operators. Topics covered included least fixed-point logic, Datalog, inflationary fixed-point logic, and the complementation problem for least fixed-point logic.

3 Constraint Databases

Jan Van den Bussche

The concept of constraint database was introduced 10 years ago by Kanellakis and his collaborators. We give an elementary introduction to the concept, and present a selection of results that have been obtained in constraint database research. Topics discussed include quantifier elimination; o-minimality; natural-active collapse; safety; arithmetical collapse; spatial databases; topological connectivity; geometrical and topological queries; linear constraint databases; and recursion. A book on the subject is due to appear in December 1999 (Constraint Databases, edited by Libkin, Kuper, and Paredaens; Springer).

4 Games in Verification

Colin Stirling

A brief introduction to the use of games of verification was presented. Property checking games for modal mu-calculus were described. In the case of finite state processes these are parity games. We considered some algorithms for this game. The second part of the talk concentrated on bisimulation equivalence which is essentially game theoretic.Some decidability results were mentioned. In particular a simpler proof of decidability of DPDA equivalence using bisimulation of strict deterministic grammars was mentioned.

5 Infinite Structures in Databases and Verification

Pierre Wolper

After introducing a simple view of verification, this talk discusses an approach for dealing with systems having an infinite number of states. The central element of the methodology that is outlined is a suitable representation formalism for infinite sets of states. Examples are used to show that this representation can often be finite automata. Moving on to constraint databases, it is then shown that automata can also be used as a convenient constraint language for temporal database. The general conclusion is that finite automata are a useful and recurring formalism for representing infinite structures.

6 CTL* vs. Monadic Datalog

Foto Afrati

This is joint work with Th. Andronikos, E. Foustoucos, and I. Guessarian.

We investigate the relationship between CTL* and Monadic Datalog$^\neg$. We give a translation of CTL*(without-the-existential-quantifier) into piecewise linear monadic Datalog$^\neg$ over acyclic finite Kripke structures. Then, by using two successor-like binary predicates, we also give a translation of CTL* into monadic Datalog$^\neg(succ_0, NextSucc)$ over finite tree Kripke structures.

7 Logical methods in pointer aliasing analysis

Michael Benedikt

Much recent work in pointer analysis can be seen as algorithms for producing a finitely-representable database of memory storage graphs. This database is an approximation to the storage structures that can be produced by a source program; once produced, this database can be mined for information relevant to the correctness or optimization of the source program.
Some of the representations produced in program analysis are quite familiar to existing verification tools (e.g. grammars, constraint sets). However, there are some exceptions: representations of infinite classes of stores that do not map easily onto standard formalisms in theoretical computer science. I'll discuss one family of representations that has this property: shape graphs. I'll give some idea of the kinds of properties that can be expressed in these formalisms that can't be expressed in classical formalisms like tree automata. I'll then give a new logic in which several variations of shape graph can be embedded, and which has a decidable satisfaction problem. The embedding of shape graphs in a logic gives an effective means for querying the shape graph for information about the source program, and give some idea of the expressive power of these graphs.
This is joint work with Tom Reps and Mooly Sagiv.

8 Implicit temporal query languages : towards completeness

Nicole Bidoit

This is joint work with Sandra de Amo.

There are two alternative ways of extending the relational model in order to represent temporal data. The first approach captures time in an implicit manner: a relationnal temporal database instance is then a finite sequence of relational instances. The second approach relies on augmenting each relation with a "timestamp" column storing the time instants of validity of each tuple. In the context of an implicit representation of time, query languages, called implicit temporal query languages, are usually based on first order temporal logic. When time is explicitly represented, queries are specified using the standard relational query languages with built-in linear order on the timestamps. One of these languages, called TS-FO, is the relational calculus (i.e. first order logic) with timestamps. A non trivial question arises then: how implicit temporal languages and explicit temporal languages relate to each other with respect to expressive power. It has been studied from various angles.

Herr & all provide a hierarchy of temporal languages[1] with respect to expressivity: (1) it is shown that *future* first order temporal logic (FTL) is strictly weaker than first temporal order logic (TL) and (2) that TL is strictly weaker than TS-FO. The first result (FTL $\subset$ TL) follows from the fact that the query `Does there exist an instant (in the future) whose state equals the initial state?` cannot be expressed in FTL but is expressible in TL. The second result (TL $\subset$ TS-FO) is derived by showing that the query `Does there exist two distinct instants (in the future) whose respective states are equal?` cannot be expressed in TL but is expressible in TS-FO. These two results are of major interest and stand in contrast with the propositional case.h The notion of complete temporal language is introduced via equivalence with TS-FO and the authors show that propositional TL is complete. Moreover, it is shown that propositional FTL is equivalent to propositional TL.

In the present paper, we study the following open problem : find an implicit first order temporal language which is complete i.e. equivalent to TS-FO. We enrich the hierarchy described above by investigating two languages NTL and RNTL. Both languages are shown to be more powerful than TL. The language NTL is not complete. The language RNTL is more expressive than NTL. Completeness of RNTL remains a conjecture.

The language NTL is the first order linear version of NCTL*. It extends TL by a temporal modality $\aleph$ ("From Now On"). Intuitively, the modality $\aleph$ allows one to choose a new initial time instant (called relative origin)

[1] Other languages investigated by these authors are stronger with respect to expressivity than TS-FO and TL.

and forget about all previous instants. In this sense, it can be said that NTL introduces a notion of relative past: the past modalities (Previous and Since) are evaluated with respect to the relative origin. This stands in contrast with TL whose modalities are of course always evaluated with respect to the absolute origin. Relative past increases the expressive power of TL in the first order case even if in the propositional case relative past is redundant with the other temporal operators. However, the language NTL is not complete: we show that the query `Does there exist 3 distinct instants whose respective states (let say` S_1, S_2 `and` S_3`) satisfy` $S_1 \cap S_2 = \emptyset$ `and` $S_1 \cup S_2 = S_3$`?` is expressible in TS-FO but not in NTL[2]. This result is proved by extending the proof technique based on communication protocol.

Because NTL fails to be complete, we extend it by investigating a rather simple idea: allowing one to forget the past is coupled together with allowing one to restore the past. The implicit language RNTL is defined by introducing a temporal modality $\Re$ whose task is to restore the segment of the past which has been "removed" by the last "application" of $\aleph$. Once again, the ability to restore the past does not add any expressive power in the propositional case. However, in the first order case, RNTL is strictly more expressive than NTL. A strict hierarchy in expressive power among fragments RNTL^i of RNTL is established. The fragment RNTL^i is defined by restriction on the maximal number of $\Re$ operators in formulas. This provides a new perspective towards proving the well-known conjecture that there is a strict hierarchy in expressive power among the i time-variable fragments TS-FO^i of TS-FO. The fragment TS-FO^i is the subclass of TS-FO formulas built by restricting the number of distinct time-variables to be at most i.

9 Stratified mu-calculus

Witold Charatonik

This is joint work with Supratik Mukhopadhyay and Andreas Podelski.

[2]One says that the query separates NTL and TS-FO.

We introduce a fragment of alternation-free modal mu-calculus and show its connections to stratified logic programs. Using this correspondence we are able to apply methods available in logic programming (in particular tabled resolution and constructive negation) for model checking of infinite-state systems specified by [constraint] logic programs, including timed automata and pushdown systems. As a consequence, we obtain model-checking procedures based on both forward and backward analysis.
The formulas of stratified mu-calculus express properties definable in terms of reachability. In the case of finite-state systems the model-checking problem for this fragment has NLOGSPACE program complexity as opposed to the PTIME-completeness for the alternation-free mu-calculus.

10 Automatic verification based on Decision Diagrams

Rolf Drechsler

This is joint work with Bernd Becker.

Nowadays modern circuit design can contain several million transistors. For this, also verification of such large designs becomes more and more difficult, since pure simulation can not guarantee the correct behavior and exhaustive simulation is too time consuming. But many designs have very regular structures, like ALUs, that can be described easily on a higher level of abstraction. This talk gives an introduction to formal verification in VLSI CAD. Decision Diagrams (DDs) are introduced and it is shown how they can be applied in equivalence checking. Techniques are outlined how DDs can make use of information provided by hardware description languages, like VHDL. A verification tool is presented that is totally automatic and experimental results are given to demonstrate the efficiency of the approach.

11 Aspects of Model Checking

E. Allen Emerson

This is a talk in two parts. The first part is a historical retrospective. It is suggested that expressiveness is at least as fundamental to the sucess of model checking as efficiency. In the second part, it is demonstrated how certain asymmetric systems are "nearly" symmetric, and amenable to symmetry reduction, which can provide an exponential compression of the state space. This latter is joint work with Richard Trefler.

12 An automata-theoretic approach to interprocedural dataflow analysis

Javier Esparza

In the talk I first presented a solution to the model-checking problem for LTL and pushdown automata. (Notice that pushdown automata are infinite state systems, since stacks has unbounded length.) The solution uses automata techniques in two different ways. First, following the automata-theoretic approach of Vardi and Wolper, the model-checking problem is reduced to the emptiness problem for Buechi-pushdown automata. Second, finite word automata are used to finitely represent infinite sets of stack contents.
In the second part of the talk I showed that our solution to the model-checking problem finds a natural application in the area of interprocedural dataflow analysis. The goal of interprocedural dataflow analysis is to obtain information about runtime properties of a procedural program without executing it. It operates by abstracting from information about data; for basic dataflow analysis problems, only information concerning which variables are accessed and modified at each program point is retained. The abstracted program can then be faithfully modelled as a pushdown automaton, and many dataflow

analysis problems can be reduced to model-checking problems.
In the final part of the talk, I showed that our results can be extended to programs with both procedures and parallelism. This requires to replace the finite word automata used in the case without parallelism by tree automata.

13 Computing Bisimulation Symbolically

Kathi Fisler

Bisimulation is a well-known equivalence relation between models. As it preserves all properties expressible in the mu-calculus, bisimulation is attractive for minimizing state-spaces in model checking. Many model checking approaches represent state spaces symbolically. Efficient symbolic bisimulation algorithms are therefore desirable in model checking. This talk explores symbolic algorithms for bisimulation minimization and discusses our experience with such algorithms in practice. It also provides a theoretical explanation for our observation that bisimulation minimization worsens, rather than improves, the performance of symbolic model checking on invariant properties.

14 Automatic Structures

Erich Grädel

We study definability and complexity issues for automatic and ω-automatic structures. These are, in general, infinite structures but they can be finitely presented by a collection of automata. Moreover, they admit effective (in fact automatic) evaluation of all first-order queries. Therefore, automatic structures provide an interesting framework for extending the approach and methods of finite model theory from finite structures to infinite ones.

While *automatic groups* have been studied rather intensively in computational group theory, a general notion of automatic structures has only been defined recently and the theory of these structures is not well-developed yet. Informally, a relational structure $\mathfrak{A} = (A, R_1, \ldots, R_m)$ is automatic if we can find a regular language $L_\delta \subseteq \Sigma^*$ (which provides names for the elements of $\mathfrak{A}$) and a function $\nu : L_\delta \to A$ mapping every word $w \in L_\delta$ to the element of $\mathfrak{A}$ that it represents. The function ν must be surjective (every element of $\mathfrak{A}$ must be named) but need not be injective (elements can have more than one name). In addition it must be recognisable by finite automata whether two words in L_δ name the same elements, and, for each relation R_i of $\mathfrak{A}$, whether a tuple of words in L_δ names a tuple belonging to R_i.
In the talk the notion of automatic and ω structures was explained, examples were presented and the relationship to automatic groups was discussed. Complexity results for *model checking* and *query evaluation* problems on automatic structures were presented. Further, *closure properties* and *definability issues* on automatic structures were studied, and a technique for proving that a structure is not automatic was explained. Finally, *model-theoretic characterisations* for automatic structures were given, via interpretations into suitable expansions of Presburger arithmetic or into tree structures. Similarly ω-automatic structures can be characterized via interpretability into a suitable expansion of the additive real group.

(This is joint work with Achim Blumensath)

15 Descriptive and Parameterized Complexity

Martin Grohe

Descriptive Complexity Theory studies the complexity of problems of the following type:

> Given a finite structure A and a sentence ϕ of some logic L, decide if A satisfies ϕ?

In this talk we discuss the *parameterized complexity* of such problems. Basically, this means that we ask under which circumstances we have an algorithm solving the problem in time $f(|\phi|)|A|^c$, where f is a computable function and $c > 0$ a constant. We argue that the parameterized perspective is most appropriate for analyzing typical practical problems of the above form, which appear for example in database theory, automated verification, and artificial intelligence.

16 Reachability Logic

Neil Immerman

We define Reachability Logic ($\mathcal{RL}$), a sublanguage of transitive closure logic (with boolean variables) that naturally expresses a wide class of path queries. $\mathcal{RL}$ admits efficient model checking: linear in the size of the structure being checked. Model checking of $\mathcal{RL}$ is also in NSPACE[$\log n$] and thus parallelizable, unlike the alternation-free mu-calculus which is PTIME complete. Modal logics PDL and CTL* can be linearly embedded in $\mathcal{RL}$. The model checking algorithm is also linear in the size of the formula, but exponential in the number of boolean variables occurring in it. In practice this number is very small. In particular, for CTL and PDL formulas the resulting model checking algorithm remains linear. For CTL* the complexity of model checking — which is PSPACE complete in the worst case — can be read from the face of the translated formula.
This is joint with Natasha Alechina.

17 Small Progress Measures and Model Checking Games

Marcin Jurdziński

In this talk I have presented a new algorithm for solving parity games, and hence for the modal μ-calculus model checking. The problem is known to be in NP $\cap$ co-NP, and even in UP $\cap$ co-UP, and it is an intriguing open problem whether there is a polynomial time algorithm for it. Our algorithm has running time $O\big(m\cdot(2n/k)^{k/2}\big)$, where n and m are the numbers of vertices and edges, respectively, of the game graph, and k is the number of priorities in the parity game. This matches the running time of the algorithms due to Browne *et al.*, 1997, and Seidl, 1996, with the best running time bounds previously known for the problem. Moreover, our algorithm works only in space $O(kn)$, while the other two algorithms need space exponential in k in worst case. The algorithm is based on computing game progress measures, which witness existence of winning strategies for players in parity games. We show existence of least game progress measures and a way to compute them as the least fixed points of certain monotone operators.

18 Variable Independence

Leonid Libkin

Given a model M whose theory admits quantifier elimination, a first-order formula $\phi(x_1, \ldots, x_n)$ and a partition P on $\{x_1, \ldots, x_n\}$, we say that ϕ conforms to P if ϕ is equivalent to a Boolean combination of formulae, each using variables from at most one block of P. The motivation for considering formulae of this kind comes from quantifier-elimination algorithms, which in many important cases are exponential in the number of variables; thus, detecting independence of variables helps reduce the cost of such algorithms.

We prove that if ϕ conforms to partitions P_1 and P_2, then it also conforms to their greatest lower bound in the partition lattice. In particular, this implies that there exists a unique minimal partition a formula conforms to. (A special case $n = 3$ was proved earlier by Cosmadakis and Kuper.) We further show that it is decidable, in polynomial time for each fixed P, whether ϕ conforms to P, if M is the real field or the real ordered group. These two cases have applications in constraint databases, as was shown earlier by Grumbach, Rigaux and Segoufin.

19 Meta-Finite Monadic Second Order Logic

Johann A. Makowsky

This is joint work with K. Meer.

In this talk we study questions related to polynomials over the reals $\mathbb{R}$. These questions are important in computational geometry but found also applications in constraint databases. We are looking for combinatorial properties of the tuples of indices of the variables which appear as non-vanishing monomials. We suggest that if this is considered as a hypergraph, then its tree width allows us to solve many questions about the polynomials in polynomial time or, equivalently, that the tree width k makes it parametrically tractable. Allthough both the problem and its solution do not mention logic, our analysis relies on the fact that many of these problems are expressible in Meta-finite Monadic Second Order Logic.

20 Unranked trees in database theory

Frank Neven

joint work with Sebastian Maneth and Thomas Schwentick

We recall the automaton model of Br uggemann-Klein, Murata, and Wood for unranked trees. Unranked trees are trees where the arity of nodes is not bounded by any constant. We argue that such trees form natural abstractions of XML documents. Next, we define query automata for expressing unary queries and point out the differences between the model for the ranked and the unranked case. Finally, we discuss a formal model for the XML transformation language XSL.

21 Temporal Properties as Models of Constraint Data Bases

Andreas Podelski

We present our approach to infinite-state model checking. This approach is based on a novel view of model checking; this view uniformly accounts for the two cases of programs over symbolic and numeric data structures, respectively. We translate a guarded command program together with a temporal logic property into a constraint data base such that its specific model is exactly the set of program states that is defined by the temporal logic property (the model is the least, greatest or some intermediate model, according to the mu-calculus formula representing the temporal logic property). Model checking thus amounts to transforming the constraint data base into an equivalent one in closed form (equivalent wrt. the specific solution). The two cases of symbolic and numeric data structures are accounted for by different closed forms. In the case of symbolic data structures (represented by strings or trees), the closed form corresponds to a finite automaton (over strings or trees). Alternatively, we can view a constraint data bases as a constraint whose variables range over sets of states; then model checking amounts to constraint solving, i.e. the procedure of transforming the constraint into an equivalent one in solved form. In most interesting cases, the procedure is possibly non-terminating; several experiments, however, have been successful and demonstrate the practical potential of this approach.

22 Probabilistic Model Checking and Compression

Michel de Rougemont

This is joint work with joint work with R. Lassaigne.

Given a probabilistic system (presented as a program P), let S be the state space, λ the accessibility function ($\lambda : S.S \rightarrow [0,1]$ such that $\sum_j \lambda(i,j) = 1$) and $P_1...P_k$ unary predicates on S ; Let $\mathcal{M} = (S, \lambda, P_1...P_k, s_0)$.

We want to check if the system satisfies a CTL formula ψ in the probabilistic sense, i.e. if $\mathcal{M} \models Prob(\psi) > \delta$. We introduce a notion of compression where C is a mapping from finite structures of a class K to finite structures of a class K', such that $C(U_n) = V_m$ where $m \leq n$.

Given a formula we want to check on $\mathcal{M}$, we can in some cases design a compression algorithm C and a formula φ in the language of K' such that:

$$\mathcal{M} \models Prob(\psi) > \delta \ \leftarrow \ C(U_n) \models \varphi$$

We give two examples : the random walk and the perfect matching on graphs.

23 Capturing LOGSPACE over Hereditarily-Finite Sets

Vladimir Sazonov

This is joint work with Alexander Leontjev.

Two versions of a set theoretic Δ-language are considered as theoretical prototypes for "nested" data base query language where data base states and queries are represented, respectively, as hereditarily-finite (HF) sets and set theoretic operations. It is shown that these versions correspond exactly to (N/D)LOGSPACE computability over HF, respectively. Such languages over sets capturing also PTIME were introduced in previous works, however, the case of LOGSPACE [A.Lisitsa and V.Sazonov, TCS (175) 1 (1997) pp. 183–222] was not completely satisfactory: problems with the closure under compositions in one approach and unnatural, however formally effective syntax in another. Here we overcome those drawbacks due to some new partial result on definability of a linear ordering over finite extensional acycling graphs and present a unified and simplified approach.
Cf. also an extended abstract in
`http://www.math.unipr.it/~gianfr/ppdp99.html`.

24 Inter-Procedural Analysis of Parallel Programs - the Abstract Interpretation Perspective

Helmut Seidl

We give a general framework for the analysis of programs with procedures and explicit parallelism. The framework is general enough to derive precise and efficient algorithms not only for bitvector problems like reaching definitions or life variables but also for non-bitvector problems like simple constant propagation. Our work is based on the work of Knopp, Steffen and Vollmer on the efficient *intra-procedural* analysis of parallel programs. Our contribution is a completely algebraic reformulation which not only simplifies proofs, presentation and algorithms but also reveals that the approach can be naturally extended to the inter-procedural setting.

25 Recognizability and Constraint Satisfaction

Wolfgang Thomas

A common framework for deterministic and nondeterministic notions of finite-state recognizability is developed.
For the deterministic case, the classical view of Büchi, Eilenberg and others, further developed by Courcelle, is recalled: The objects under consideration (words, labeled trees, labeled graphs) are represented by terms over a functional signature Σ, and a recognizable set is obtained via a homomorphism from the term algebra T_Σ into a finite Σ-algebra.
In the nondeterministic case, we refer to relational structures and relational homomorphisms into a finite structure, where each input structure requires its own homomorphism (as in the theory of constraint satisfaction proposed by Féder and Vardi). It is shown that nondeterministic word automata, nondeterministic tree automata, tiling systems (as acceptors of labeled grids), and nondeterministic graph acceptors can be considered as special cases of finite relational structures which accept their inputs via relational homomorphisms.
So in this unifying framework deterministic and nondeterministic recognizability arise by the uniform, respectively nonuniform application of homomorphisms into finite structures. Moreover, a number of results on deterministic versus nondeterministic recognizability and on the connection to monadic second- order logic can be verified on this general level as metatheorems, covering corresponding results over the domains of finite words, trees, and

graphs.

26 Emptiness of Tree Automata

Moshe Y. Vardi

Testing emptiness of parity tree automata is a basic algorithmic building block in many decision procedure for satisfiability of program logics. In this talk we describe a simple algorithm that runs in time $n^O(k)$, where n is the size of the transition table of the automaton and k is the parity index. The key to the simplicity of the algorithm is the use of alternation.
Work done jointly with Orna Kupferman.

27 Linear Time Datalog: Temporal versus deductive reasoning in verification

Helmut Veith

This is joint work with Georg Gottlob and Erich Grädel.

We show that Datalog is well-suited as a temporal verification language. Datalog is a well-known database query language relying on the logic programming paradigm. We introduce Datalog LITE, a fragment of Datalog with negation, and present a linear time model checking algorithm for Datalog LITE. We show that Datalog LITE subsumes temporal languages such as CTL and the alternation-free mu-calculus, and in fact give easy syntactic characterizations of these temporal languages. We prove that Datalog LITE has the same expressive power as the alternation-free portion of guarded fixed point logic.

28 Verifying protocols for electronic commerce

Victor Vianu

Electronic commerce is emerging as one of the major Web-supported applications requiring database support. We introduce and study high-level declarative specifications of protocols for business transactions, called *relational transducers*. These are devices that map sequences of input relations into sequences of output relations. The semantically meaningful trace of an input-output exchange is kept as a sequence of *log* relations. We consider problems motivated by electronic commerce applications, such as log validation, verifying temporal properties of transducers, and comparing two relational transducers. Positive results are obtained for a restricted class of relational transducers called *Spocus transducers* (for semi-positive outputs and cumulative state). We argue that despite the restrictions, these capture a wide range of practically significant business models.

29 Logic on Traces

Igor Walukiewicz

30 CTL^+ Is Exponentially More Succinct Than CTL

Thomas Wilke

In the talk I present a sketch of a proof that CTL^+ is exponentially more succinct than CTL. More precisely, I sketch a proof of the following statement. Every CTL formula equivalent to the CTL^+ formula

$$\mathsf{E}(\mathsf{F}p_0 \wedge \cdots \wedge \mathsf{F}p_{n-1})$$

is of length at least $\binom{n}{\lceil n/2 \rceil}$, which is $\Omega(2^n/\sqrt{n})$. This matches almost the upper bound provided by Emerson and Halpern, which says that for every CTL^+ formula of length n there exists an equivalent CTL formula of length at most $2^{n \log n}$.

The proof also shows that an exponential blow-up as incurred in known conversions of nondeterministic Büchi word automata into alternation-free μ-calculus formulas is unavoidable. This answers a question posed by Kupferman and Vardi.

The proof of the above lower bound exploits the fact that for every CTL (μ-calculus) formula there exists an equivalent alternating tree automaton of linear size. The core of this proof is an involved cut-and-paste argument for alternating tree automata.

This is going to be published in the proceedings of FST & TCS '99; a detailed account is available as a technical report at `ftp://ftp.informatik.rwth-aachen.de/pub/reports/1999/index.html`.