André Arnold, Helmut Seidl,
Bernhard Steffen (editors):

Algorithms in Automata Theory

Dagstuhl-Seminar-Report; ??
7.2.–11.2.1994 (9406)

Dagstuhl Seminar

on

Algorithms in Automata Theory

Organized by:

André Arnold (Université Bordeaux I)
Helmut Seidl (Universität des Saarlandes)
Bernhard Steffen (Universität Passau)

Schloß Dagstuhl, February 7 - 11, 1994

Contents

1 Preface

The meeting ALGORITHMS IN AUTOMATA THEORY was the fourth in a series of automata theory seminars held at Dagstuhl Castle. It was motivated by the observation that although many fields in computer science use automata as computational models there is only very little scientific exchange between the groups. Rather, one observes an independent development of similar algorithms for the various fields of application, which range from classical applications like language recognition, e.g. for syntax analysis, to the modelling of distributed systems and the generation of code selectors, the implementation of term-rewriting systems and the construction of type-checkers.

Historically, algorithms for the automatic analysis or transformation of automata, e.g. for the verification of the equivalence of two automata or for the minimization of a single automaton are well-known from classical automata theory. However, to cope with the new applications, such algorithms have in general to be modified. Particularly important is the refinement of the usual view of automata in terms of languages, which identifies those machines accepting the same language, to more discriminative criteria, which in particular take the branching potential of an automaton into account. On this basis, a number of equivalence-provers, model-checkers and static program-analyzers have been developed and implemented in the last few years. Also on the classical automata theoretical side several new methods and implementations have been developed, which raise the question of integration of available tools.

Therefore, the aims of the seminar were to investigate the rich panorama of automata-based algorithms, to compare implementations, to exchange experiences, and to enhance the cooperation among the active research groups. The lively discussions both during the sessions and in the evenings confirmed the need for such an exchange between the different research communities and they illustrated the success of the meeting. In fact, all participants agreed about the stimulating atmosphere of the seminar, both from the scientific point of view and the hospitality of the local organizers, and they are all looking forward to the next Dagstuhl meeting.

B. Steffen

2 Final Seminar Programme

Monday, February 7, 1994

8:50 Opening Address
Bernhard Steffen, Germany

Session 1, 9:00 – 12:00
Chair: Kim G. Larsen

9:00 *Model Checking in the Modal Mu-Calculus*
Rance Cleaveland, USA

9:40 *Incremental Model Checking in the Modal Mu-Calculus*
Scott A. Smolka, USA

10:20 Break

10:40 *Branching Time Temporal Logic and Amorphous Tree Automata*
Orna Grumberg, Israel

11:20 *On the Complexity of Deciding Bisimilarity of Normed Context-Free Processes*
Faron Moller, UK

Session 2, 15:30 – 17:45
Chair: Rance Cleaveland

15:30 *Undecidable Equivalences for Basic Parallel Processes*
Hans Hüttel, Denmark

16:10 *Decidability of Model Checking for Petri Nets*
Javier Esparza, UK

16:50 Break

17:05 *Complexity Theoretical Aspects of Problems/Algorithms in Automata Theory*
Klaus-Jörn Lange, Germany

Tuesday, February 8, 1994

Session 3, 9:00 – 12:00
Chair: Scott A. Smolka

9:00 *Algorithms for Verification of Real Time Systems*
Kim G. Larsen, Denmark

9:40	*Decidability for Real Time Systems over Dense Time-Domains* Carsten Weise, Germany
10:20	Break
10:40	*On the Cascaded Decomposition of Automata, its Complexity and its Application to Logic* Oded Maler, France
11:20	*Failure Detection in Automatic Control: A Case Study or Verification of a Real Example* Srećko Brlek, Canada

Session 4, 15:30 – 17:45

Chair: André Arnold

15:30	*Grail and its Consequences* Derick Wood, Canada
16:10	*AMORE: Automata, Monoids, and Regular Expressions* Wolfgang Thomas & Andreas Potthoff, Germany
16:50	Break
17:05	*Classes of Self-Stabilizing Protocols* Joffroy Beauquier, France

Wednesday, February 9, 1994

Session 5, 9:00 – 12:00

Chair: Bernhard Steffen

9:00	*Model Checking Context-Free Processes* Julian C. Bradfield, UK
9:40	*Pushdown Processes: Parallel Composition and Model Checking* Olaf Burkart, Germany
10:20	Break
10:40	*Model Checking of Higher-Order Processes* Hardi Hungar, Germany
11:20	*Formal Callability and its Application in Program Analysis* Jens Knoop, Germany

∗ ∗ ∗ Afternoon Excursion ∗ ∗ ∗

Wild Session I, 19:30 – 20:30

19:30 *The Mu-Calculus on Trees and Rabin's Complementation Theorem*
André Arnold, France

Thursday, February 10, 1994

Session 6, 9:00 – 12:00

Chair: Wolfgang Thomas

9:00 *Assumption/Commitment Specifications*
Bernhard Josko, Germany

9:40 *Abstract Interpretation and Verification of Parallel Programs*
Susanne Graf, France

10:20 Break

10:40 *Regularly Controlled Term Rewriting*
Helmut Emmelmann, Germany

11:20 *Solving Systems of Set Constraints Using Tree Automata*
Remi Gilleron

Session 7, 15:00 – 18:00

Chair: Helmut Seidl

15:30 *Module Charts*
Andreas Claßen, Germany

16:10 *Mu-CRL, Protocol Verification, Proof Checking*
Jan Friso Groote, The Netherlands

16:50 Break

17:05 *Pumping and Cleaning*
Max Dauchet, France

Wild Session II, 19:30 – 20:45

19:30 *A Preorder whose Kernel is Strong Bisimulation*
Joachim Parrow, Sweden

20:00 *On the Analysis of Hybrid Systems*
Oded Maler, France

Friday, February 11, 1994

Session 8, 9:00 – 11:30

Chair: Derick Wood

9:00 *Efficient Transformations of Context-Free Grammars*
Didier Caucal, France

9:40 *Learning Picture Sets from Examples*
Anne Brüggemann-Klein, Germany

10:20 Break

10:40 *Practical Computations of the Syntactic ω-Semigroup*
Jean Eric Pin, France

3 Abstracts of Presentations

The following abstracts appear in alphabetical order of speakers.

Classes of Self-Stabilizing Protocols

Joffroy Beauquier
Université Paris Sud
Orsay Cedex, France

Self-stabilization is an abstraction of failure tolerance for distributed systems in which transient failures can corrupt data memories, processor registers and communication channels, but not the program code. Between two successive failures, the system has to regain its consistency by itself, without any external intervention. A system is said to be self-stabilizing if, starting from any possible state, it eventually reaches a correct state. Using the Arnold-Nivat model of synchronized automata, we formalize the notions of locality degrees for detection and correction and we give a condition for a protocol to be automatically transformed into a self-stabilizing one. This result yields a classification of distributed protocols according to the values of their locality degrees.

Model-Checking Context-Free Processes

Julian Bradfield[1]
University of Edinburgh
Edinburgh, Scotland, UK

The problem of determining whether a context-free process (*alias* a BPA process) satisfies a property expressed in the propositional modal mu-calculus, has been known for some years to be decidable. However, the proof is difficult, proceeding via deep results of Muller and Schupp and ultimately Rabin, and gives a non-elementary complexity. In this talk, we give a tableau method for deciding the problem. A tableau has sequents $\alpha\Phi \vdash \phi$, which can be understood as "if a CF process ω satisfies all the formulae in the set Φ (which will be a subset of the closure of the formula ϕ_0 being checked), then the process $\alpha\omega$ satisfies ϕ (a sub-formula of ϕ_0)". Tableaux are constructed by rules which are for the most part obvious; the important rule is the 'Hoare rule', which splits a sequent involving a sequential composition $\alpha = \alpha' P$ (for P a process variable) into one for α' and one for P, by 'guessing' intermediate formulae, as in the sequential composition rule of Hoare logic. Once a tableau has been constructed, some extra conditions on it need to be checked to give 'success'. The system is sound, and complete in the sense that there exists a successful tableau for a true root sequent (under certain conditions, which obtain when model-checking a process against a formula). This gives decidability; moreover, there is an exponential bound on the possible size of a tableau, giving an elementary

[1] This is joint work with Javier Esparza and Colin Stirling.

complexity. Finally, the system transfers immediately to pushdown processes (finite state control plus a pushdown stack).

Failure Detection in a Control Process: A Case Study or Verification of a Real Example

Srećko Brlek
Université du Québec à Montréal
Montréal, Québec, Canada

We discuss a model presented in [1] concerning a HVAC (Heating, Ventilation and Air Conditioning) system for which the authors consider the problem of failure detection and control.

The system is modelized as the synchronized product of the finite labeled transition systems describing the behaviour of the components of the HVAC. A desirable property one expects in the behaviour of such a system is the following:

> From the (sufficiently long) trace of observable events one should be able to determine if it is the projection of a trace containing an unobservable failure event.

To achieve this goal a property of diagnosability is presented which ensures the existence of a non ambiguous failure detection. Partial verification was carried out using a transition based model checker [2] and also a μ-calculus based tool [3].

[1] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. *Failure d Diagnosis using DES models*. Tech. Rep. CGR 93-16, Dept. of EECS, Univ. of Michigan, MI 48109, USA, 1993.

[2] Arnold, A. *MEC: a system for constructing and analysing transition systems*. In Proceedings of the International Workshop on Automatic Verification of Finite State Systems, Grenoble, France, Springer-Verlag, LNCS 407 (1989), 117 - 132.

[3] Corsini, M. M., and Rauzy, A. *Symbolic Model Checking and Constraint Logic Programming*. To appear in Proceedings of the European Symposium on Programming (ESOP'94), Edinburgh, Scotland, UK, April 11 - 13, 1994.

Learning Picture Sets from Examples

Anne Brüggemann-Klein
Universität Freiburg
Freiburg, Germany

We study picture sets in the context of computational learning theory.

We consider pictures in the Cartesian plane that can be traced by a finite number of consecutive one-unit length movements to the left, to the right, up, and down. Such pictures are represented by words over the alphabet $\{l, r, u, d\}$ of single-step movements.

For a fixed picture q, we introduce the class $\mathcal{B}_q$ of regular languages that represent q; that is, each word in each language of $\mathcal{B}_q$ represents q. We show that this class is exactly as hard to learn as the full class of all regular languages.

We also investigate the class $\mathcal{B}_c$ which contains, for each picture q, the full set of all words over $\{r, l, u, d\}$ that represent q.

Pushdown Processes: Parallel Composition and Model Checking

Olaf Burkart[2]
Rheinisch-Westfälische Technische Hochschule Aachen
Aachen, Germany

In this talk, we consider a strict generalization of context-free processes, the *pushdown processes*, which are particularly interesting for three reasons: First, in contrast to context-free processes that do not support the construction of distributed systems, they are *closed under parallel composition* with finite state systems. This is shown by proving a new expansion theorem, whose implied 'representation explosion' is no worse than for finite state systems. Second, they are the *smallest* extension of context-free processes allowing parallel composition with finite state processes, which we prove by showing that every pushdown process is bisimilar to a (relabelled) parallel composition of a context-free process (namely a stack) with some finite process. Third, they can be *model checked* by means of an elegant adaptation to pushdown automata of the second order model checker introduced in [1]. As arbitrary parallel composition between context-free processes provides Turing power, and therefore destroys every hope for automatic verification, pushdown processes can be considered as the appropriate generalization of context-free processes for frameworks for automatic verification.

[1] Burkart, O., and Steffen, B. *Model Checking for Context-Free Processes*. In Proceedings of the 3rd International Conference on Concurrency Theory (CONCUR'92), Stony Brook, NY, USA, Springer-Verlag, LNCS 630 (1992), 123 - 137.

[2] Christensen, S, Hüttel, H., and Stirling, C. *Bisimulation Equivalence is Decidable for all Context-Free Processes*. In Proceedings of the 3rd International Conference on Concurrency Theory (CONCUR'92), Stony Brook, NY, USA, Springer-Verlag, LNCS 630 (1992), 138 - 147.

[3] Caucal, D., and Monfort, R. *On the Transition Graphs of Automata and Grammars*. In Proceedings of the 16th International Workshop on Graph-Theoretic Concepts in Computer Science, Berlin, Germany, Springer-Verlag, LNCS 484 (1990), 311 - 337.

[2]This is joint work with Bernhard Steffen.

[4] Hungar, H., and Steffen, B. *Local Model-Checking for Context-Free Processes*. In Proceedings of the 20th International Colloquium on Automata, Languages, and Programming (ICALP'93), Lund, Sweden, Springer-Verlag, LNCS 700 (1993), 593 - 605.

[5] Muller, D. E., and Schupp, P. E. *The Theory of Ends, Pushdown Automata, and Second-Order Logic*. Theoretical Computer Science 37, 1 (1985), 51 - 75.

Efficient Transformations of Context-Free Grammars

Didier Caucal
Université de Rennes
Rennes Cedex, France

Given a context-free grammar G of size n (the length of description), we show that the minimization of the number of non-terminals,

a) using the equivalence between words, can be done polynomially in n when G is any simple grammar;

b) using the equivalence between finite sets of words, can be done exponentially in n when G is any parenthesis grammar or any tree grammar;

c) using the contexts between sets of non-terminals, can be done exponentially when G is any parenthesis grammar, and polynomially when G is an invertible parenthesis grammar.

Finally, we show that the time complexity

i) to put G in pre-Chomsky normal form: every right hand side is a letter or two non-terminals, is in $O(n)$;

ii) to put G in Chomsky normal form is in $O(n^2)$;

iii) to put G in Greibach normal form is in $O(n^4)$.

Module Charts

Andreas Claßen
Universität Passau
Passau, Germany

This talk presents a hierarchical extension of the automata formalism designed for the specification of reactive systems. The main motivation for the extensions comes from the fact that large systems can only be specified in a structured, modular way. Module interfaces, local event memories and a special communication mechanism guarantee modularity for our formalism. In these concepts, Module Charts differ profoundly from Statecharts, another hierarchical extension of automata. To emphasize the modularity of our formalism, refinement and abstraction are studied in detail. Several concepts are introduced to flexibly guarantee the completeness of semantic refinement.

Model Checking in the Modal Mu-Calculus

Rance Cleaveland
North Carolina State University
Raleigh, NC, USA

The modal mu-calculus is a very expressive modal logic that provides fixed-point constructs for defining formulas. In a certain sense it may be viewed as the "assembly language" of automatic verification for finite-state systems, since different verification approaches may be efficiently reduced to determining whether a system satisfies a certain mu-calculus formula. This talk surveys recent results on efficient algorithms for "model checking" in this logic—i.e. for checking whether a given finite-state system satisfies a given formula.

Pumping and Cleaning

Max Dauchet[3]
Université de Lille I
Villeneuve d'Ascq Cedex, France

We explain why new classes of tree automata can be tools for rewriting and symbolic computation. We introduce encompassment automata, which we use to prove decidability of the encompassment theory (the first order theory with monadic atomic predicates $Et(u)$ = " a (variable) subterm of u is an instance of a term t "). We illustrate on this class how pumping technics are used to prove termination of an efficient cleaning algorithm.

[1] Caron, A. C., Comon, H., Coquidé, J. L., Dauchet, M., and Jacquemard, F. *'Pumping, Cleaning and Symbolic Constraints Solving*. To appear in Proceedings of the 21st International Colloquium on Automata, Languages, and Programming (ICALP'94), Jerusalem, Israel, July 11 - 15, 1994.

[3]This is joint work with A. C. Caron (LIFL, Lille), H. Comon (LRI, Orsay), J-L. Coquide (LIFL, Lens), and F. Jacquemard (LRI, Orsay).

Regularly Controlled Term Rewriting

Helmut Emmelmann
Universität Karlsruhe
Karlsruhe, Germany

Let T be a universe of ground terms, $L_{\mathcal{G}} \subset T$ a regular tree language given by a tree grammar G, and TRS a left linear term rewriting system. We study the problem of rewriting a term $i \in T$ into $t \in L_{\mathcal{G}}$. It occurs (in a modified version using cost values) in the context of code selection.

Regularly controlled term rewriting constructs, when applicable, a non–deterministic bottom up tree transducer that performs the desired rewrites. First the problem is reduced to the following acception problem: rewrite a term t to a final state Z using a term rewriting system TRS'. This problem is solved by simulating TRS' with a ground term rewriting system GTRS. GTRS is constructed by instanciating the variables of TRS'-rules with ground terms. GTRS is said to be complete, iff the acception problem using GTRS has the same solutions as for TRS'. We use tree automata theory to decide whether GTRS is complete. Then we give an algorithm that determines a complete GTRS if it does exist. The question if it exists is undecidable.

Finally the acception problem for GTRS can be solved by constructing and implementing a tree transducer.

Decidability of Model Checking for Petri Nets

Javier Esparza
University of Edinburgh
Edinburgh, Scotland, UK

In this talk I present a small survey on the decidability of the model checking problem for several temporal logics and Petri nets. The main conclusion is that, loosely speaking, linear time logics are 'more decidable' than branching time ones. In particular, the linear time mu-calculus without atomic sentences (i.e., considering only closed formulae), which is a rather powerful linear time logic, is decidable. On the contrary, a weak branching time logic which can only express possibility properties is already undecidable. The talk was based on the references [1] and [2] given below.

[1] Esparza, J. *On the Decidability of Model Checking for Several Mu-Calculi and Petri Nets*. To appear in Proceedings of the Colloquium on Trees in Algebra and Programming (CAAP'94), Edinburgh, Scotland, UK, April 11 - 13, 1994.

[2] Esparza, J, and Nielsen, M. *Decidability Issues for Petri Nets – A Survey*. To appear in EATCS Bulletin 52 (Concurrency column).

Set Constraints and Automata

Remi Gilleron [4]
Universitè de Lille I
Villeneuve d'Ascq Cedex, France

Set constraints have been studied for their ability to describe properties of programs or more generally to describe relationships between sets of terms of a free algebra. We consider positive set constraints of the form $exp \subseteq exp'$ and negative set constraints of the form $exp \not\subseteq exp'$. The set expressions exp and exp' are built from set variables, function symbols and the set union, intersection and complement. We present a decision procedure for satisfiability of such systems of set constraints. This decision procedure is an automata-based algorithm. Moreover we prove in a constructive way that a non empty set of solutions always contains a regular solution, i.e. a tuple of regular tree languages. Our new class of automata can be viewed as an acceptor model for mappings from $T(\Sigma)$ into $\{0,1\}^n$ and we think that this new class of automata could be interesting in its own.

Using Abstract Interpretation for the Verification of Parallel Programs

Susanne Graf
Verimag - Grenoble
Miniparc Zirst, Montbonnot Saint Martin, France

The framework of abstract interpretation allows to define a notion of abstraction (respectively refinement) between programs parametrized by a Galois connexion relating the powersets of concrete and abstract states; we have proven this notion to be exactly Milner's simulation relation parametrized by a relation ρ relating abstract and concrete states. An important question is, for which properties f the satisfaction of f on the abstract program implies (is equivalent to) the satisfaction of f on the concrete program; we say that f is (strongly) preserved from the abstract to the concrete program. We have obtained (strong) preservation results for important fragments of the propositional μ-calculus with past modalities (all formulas containing only universal path quantifiers and negations only on atomic propositions are preserved from the abstract to the concrete program).

Given a concrete program (represented by a finite transition relation R) and a relation ρ between concrete and abstract states, we obtain also, in an almost straightforward manner, a way of computing a corresponding exact abstract transition relation $R_\rho = \rho^{-1}R\rho$. Furthermore simulation is preserved under union and intersection of transition relations and — under some conditions, satisfied in most practical applications — also under parallel composition. This is an important property in practice.

[4] This is joint work with S. Tison and M. Tommasi.

These theoretical results allow to use the following verification method: given a program given as a parallel composition of component programs and a property f to be verified on it, define an abstract domain and a relation ρ between the concrete and the abstract main, abstract states such that for every atomic proposition p occurring in f, the abstract interpretation of p and $\neg p$ are disjoint. Then compute an abstract program by abstracting and composing components in any order. If the property holds on the obtained abstract program by using the abstract interpretation function of atomic propositions, then it holds on the concrete program using the original interpretation function. If the property doesn't hold, nothing about the validity of the property can be concluded; however, using a debugging tool, it is possible to either find a counter example invalidating the property or to obtain information about how to refine the abstraction relation ρ.

We have implemented this method for programs which are parallel compositions of guarded command programs defined on boolean variables, where internally programs are represented by a set of BDDs — one for the representation of the transition relation R_i of each guarded command. For non-trivial examples, we were able to compute the quite exact abstraction obtained by computing for every command BDD representing the exact abstract transition relation R_{i_ρ}, and to verify their correctness on these abstractions by using our symbolic model checker.

In the case of programs over infinite domains, this method is not applicable directly, but one may still obtain an abstract program on some finite domain by using the method proposed by abstract interpretation, which consists in replacing each operation on the concrete domain by some operation on the abstract domain. This yields a much looser abstraction, but it is the best one can do. We have applied this method to some distributed Cache memory system, where the used data structures are essentially infinite buffers of integers, memories of integers, ... For each property to be verified, it was quite straightforward to guess a small finite abstract domain and the associated operations. This allowed us to verify in a quite simple manner the distributed cache memory, which is known to be a difficult problem.

μCRL, Correctness Proofs of Protocols and Proof Checking

Jan Friso Groote
Utrecht University
Utrecht, The Netherlands

μCRL has been defined in 1990 as a language that combines process algebra (ACP) with a generalized sum ($\sum_{d:D}$), a then-if-else construct ($\_ \lhd \_ \rhd \_$) and data (equationally specified abstract datatypes). In 1991 a formal proof system for μCRL has been defined. This proof system has been used to prove Milner's Scheduler, a Bakery Protocol, a Bounded Retransmission Protocol and Sliding Window Protocols correct. These protocols are now among the most complicated that have been proven correct in process algebra. But the proofs are straightforward, due to newly developed techniques about invariants and 'Cones and Foci'.

The proofs of most of the above mentioned protocols have been checked using the proof checker Coq. From this experience we draw the conclusion that it is in principle perfectly possible to formally check proofs, thereby guaranteeing almost absolute correctness. A bottleneck in the application of proof checkers is the enormous number of axioms that need to be applied. Currently, work is being done to let proof checkers generate large parts of the proofs automatically, leaving only the crucial steps of the proof to be provided explicitly. Techniques that are being used for that purpose are term rewriting and resolution.

Branching Time Temporal Logic and Amorphous Tree Automata

Orna Grumberg[5]
The Technion
Haifa, Israel

An automata-theoretic framework for branching-time temporal logics is presented. We introduce a new type of finite automata on infinite trees, the Amorphous Automata, and use them as a formalism to represent efficiently CTL formulas. In addition, we introduce simultaneous trees, and associate with every model for CTL, a simultaneous tree that enables a tree automaton to visit different nodes on the same path of the tree simultaneously. With every formula f we associate an amorphous automaton $U(f)$ that accepts exactly those simultaneous trees (of any branching degree) that originate from models that satisfy the formula. This enables to use the automaton for model checking which is reduced to the membership problem, and for satisfiability decision, which is reduced to testing the nonemptiness of an extension of $U(f)$ that does not assume simultaneous input trees.

The amorphous automata for CTL use the Buchi acceptance condition. The size of an automaton is linear in size of the formula and the extension required for satisfiability is exponential. Based on that, we get a polynomial model checking procedure and an exponential decision procedure for CTL, both match the known lower bounds. This is the first time that a model checking algorithm for a branching-time temporal logic is placed in the automata-theoretic framework.

Model Checking of Higher-Order Processes

Hardi Hungar
Universität Oldenburg
Oldenburg, Germany

Model checking provides a powerful tool for the automatic verification of behavioral systems. There are iterative algorithms and tableaux-based algorithms. At first sight, both

[5]This is joint work with Orna Bernholtz (The Technion, Haifa, Israel).

kinds of algorithms seem to be restricted to finite systems. But by using second-order assertions Burkart and Steffen were able to handle infinite systems given in the form of *context-free* process systems with an iterative algorithm, and also a local model checking procedure has been developed.

Context-free processes (CFPs) are labeled transition systems generated by edge replacement systems: edges labeled with a nonatomic action are to be replaced by the transition system defining the action. Because the defining systems may contain nonatomic actions as well, the resulting expansion might be infinite. One of the main observations leading to the decidability result is that due to the context-free nature of the replacement process, an edge labeled with a nonatomic action stands for the same transition system wherever it occurs. And only finitely many different nonatomic actions do exist.

The effect of each of those transition systems to the validity of formulae, if the attention is restricted to a finite set of formulae, can be computed in a mutual recursive way. I.e. model checking of alternation-free mu-calculus formulae is possible.

CFPs can model infinite structures which are similar to stacks, but this is roughly the borderline of their modeling power. Still more general structures allow model checking, as it is shown by this work.

Just like parameterless procedures can be generalized to higher-order procedures (and still are axiomatizable), higher-typed expansion rules for the generation of processes introduce *higher-order processes* (HOPs) which generalize CFPs. From context-free systems, higher-order process systems arise like macro grammars from context-free grammars: Nonatomic actions may be parametrized. This provides the power to model *nested stacks*, which are outside the scope of CFPs.

The parametrization of nonatomic actions leads to an infinite number of transition systems represented by nonatomic actions. But finite higher-order assertions enable compositional reasoning, extending the second-order reasoning which is sufficient for CFPs.

The main result of this work is that alternation-free mu-calculus formulae can be decided for higher-order processes. This holds even in the presence of unguarded recursion, and extends in a natural way if processes are enriched by a finite control (higher-order pushdown processes).

Undecidable Equivalences for Basic Parallel Processes

Hans Hüttel
Aalborg University
Aalborg, Denmark

Much attention has been devoted to the study of process calculi and in particular to behavioural semantics for these calculi. In order to capture the behavioural aspects of processes, a variety of equivalences have been proposed. A systematic approach consists of classifying the equivalences according to their coarseness. For this purpose van Glabbeek proposed the *linear/branching time spectrum*.

Various criteria exist for comparing the merits and deficiencies of these equivalences. One of these is whether or not an equivalence is *decidable*.

Recent results show that strong bisimilarity is decidable for the class of Basic Parallel Processes (BPP), which corresponds to the subset of CCS definable using recursion, action prefixing, nondeterminism and the full merge operator. In this paper we examine all other equivalences in the linear/branching time spectrum and show that *none* of them are decidable for BPP. Most of these results follow from a reduction from the empty input halting problem for two-counter Minsky machines to the preorder from which the equivalence is induced.

Assumption/Commitment Specifications

Bernhard Josko
OFFIS
Oldenburg, Germany

Assumption/commitment specifications are used to describe reactive systems, systems which interact which their environments. Assumption/commitment specifications are given by pairs $(assm,\ comm)$ where

- *assm* describes the expected behaviour of the environment and
- *comm* is the commitment which should be satisfied by the module under consideration provided the environment guarantees *assm*.

We compare different approaches to give a formal semantics of the validity of assumption/commitment specifications given by temporal logic formulae:

$$M \models (assm,\ comm)$$

We consider linear time as well as branching time temporal logic specifications.

$M \models tr(assm, comm)$: On the one hand we give a translation of pairs $(assm,\ comm)$ into a single temporal logic formula. For linear time temporal logic this is simply the implication $assm \rightarrow comm$. For branching time temporal logic this can not be done in such an easy manner. A translation is obtained by using the logic ECTL which involves automata constructives which are used to keep track of the assumption information when evaluating a commitment. It is well known how ECTL can be translated into the μ-calculus.

$M \times \mathcal{A}_{assm} \models comm$: On the other hand we can construct a product of the given module M and an automaton derived from the assumption *assm*. Then the commitment is interpreted w.r.t. this product automaton.

It can be shown that both definitions are equivalent. The first approach is useful to compare the expressive power of assumption/commitment specifications with other logics whereas the later one is more appropriated regarding verification (e.g. model checking).

Formal Callability and its Application in Program Analysis

Jens Knoop
Universität Passau
Passau, Germany

The notion of *formal callability* is introduced, which determines for a formal procedure call occurring in a program the set of procedures that may actually be called by it. In interprocedural *program analysis* working on push-down automata this reduces the investigation of programs with formal procedure calls to the analysis of programs without formal procedure calls by treating formal procedure calls as higher-order branch statements. It turns out that formal callability is a natural refinement of the well-known formal reachability problem, and like formal reachability, it is not decidable in general. It is shown that formal callability is decidable for programs without global formal procedure parameters, but within a time bound that is exponential in the program size. Therefore, additionally the notion of *potential passability* is introduced, which can *efficiently* be computed and is a *correct* approximation of formal callability. Moreover, for programs of mode depth 2 without global formal procedure parameters it is even *complete*.

Complexity Theoretical Aspects of Problems in Automata Theory

Klaus-Jörn Lange
Technische Universität München
München, Germany

The talk centers around two topics: density of information in the input and types of functions we want to compute. In both cases the change of models deeply affects the corresponding complexities.

Changing the amount of information by *padding* directly decreases the complexity and leads to the well-known phenomena of embedding and downward separation. Changing the information amount by compression shows a less uniform pattern. In general, we get higher complexities depending very much in the choice of the model and the language of representation.

Given a language $L \subseteq X^*$, it is often necessary to compute more than just the characteristic function $c_L : X^* \to \{0,1\}$ (i.e.: to solve the membership problem of L). Typical objects of interest are derivation trees and accepting computations (*Parsing*) or numbers of output or degrees of ambiguity (*Counting*). While parsing usually is comparable to recognition w.r.t. complexity, counting problems are often of a high complexity yielding complete problems for function classes of the #-, opt-, and span-type.

Algorithms for Verification of Real–Time Systems

Kim Guldstrand Larsen
Aalborg University
Aalborg, Denmark

This talk describes algorithmic techniques underlying the verification tool EPSILON. That is we describe techniques for deciding various behavioural equivalences and preorders between real–time systems, in particular we discuss time–abstracting, time–sensitive and speed–relating equivalences/preorders. In all cases the algorithmic technique is based on a reduction to that of deciding a "well–behavedness" property of an induced finite–state symbolic transition system. The checking for "well–behavedness" may be performed using one of the numerous existing modelchecking algorithms though we advocate the use of local checking techniques due to the expected high number of inaccessible states. The reduction algorithms presented are due to Cerans and Larsen/Wang, and in all cases the induced symbolic transition system is based on the idea of region graphs by Alur and Dill. We also show how the algorithms may be extended with time–quantitites sufficient for generating distinguishing formulae.

On the Cascaded Decomposition of Automata, its Complexity and its Application to Logic

Oded Maler
Verimag
Grenoble, France

In this talk I described and proved the Krohn-Rhodes decomposition theorem. This theorem states that every finite-state deterministic automaton is homomorphic to a cascade product of permutation-reset automata, and in particular that a counter-free automaton can be decomposed into a cascade of reset-identity automata. I gave an exponential algorithm (based on Eilenberg's proof of the theorem) as well as an exponential lower-bound on the size of the decomposition. I showed that a language accepted by cascade products of reset-identity automata admit a natural description by linear past temporal logic formulae, and hence there is an exponential algorithm for translating from counter-free automata to temporal logic. This procedure can be extended easily to translate star-free ω-languages from ω-automata into future temporal logic.

On the Analysis of Hybrid Systems

Oded Maler
Verimag
Grenoble, France

Hybrid system are systems where discrete state-transitions and continuous dynamics interact. Their investigations is motivated both by practical concerns (analysis of embedded real-time systems) and by theoretical interest.

In this talk I defined a class of hybrid systems, namely dynamical systems with piecewise-constant derivatives (PCD systems). Such systems consist of a partition of the Euclidean space into a finite set of polyhedral sets (*regions*). Within each region the dynamics is defined by a constant vector field, hence the discrete transitions occur only on the boundaries between the regions where the trajectories change their direction.

With respect to such systems we investigate the reachability question: *Given an effective description of the systems and two points x_1 and x_2 in the space, is there a trajectory starting at x_1 that reaches x_2?* The main results presented in this talk were a decision procedure for two-dimensional systems (joint work with A. Pnueli), and an undecidability result for three or more dimensions (joint work with E. Asarin).

On the Complexity of Deciding Bisimilarity of Normed Context-Free Processes

Faron Moller
University of Edinburgh
Edinburgh, Scotland, UK

In this month's Journal of Theoretical Computer Science, Dung T Huynh and Lu Tian demonstrate that the problem of deciding bisimilarity between normed context-free processes is in $\Sigma_2^{\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}}$. Briefly, they present an algorithm which guesses a proof of the equivalence of two process terms, and then verifies this proof in polynomial time using oracles freely answering questions which are in NP. We demonstrate that the questions asked of these oracle calls are actually in P, so the problem actually lies in NP. More than this, we demonstrate their initial guess can be constructed in polynomial time, so the decision problem can actually be solved in polynomial time.

A Preorder whose Kernel is Strong Bisimulation

Joachim Parrow
SICS
Kista, Sweden

We define the relation $\prec$ on CCS agents by $P \prec Q$ if for some $A \overset{\mathrm{def}}{=} E(A)$ it holds $P \sim A$ and $Q \sim E(Q)$. So for example $P \prec P + R$ for all P, R. It is not known if $\prec$ is transitive, but its transitive closure is a preorder and has strong bisimulation as its kernel. It is not a precongruence; the greatest contained and least containing precongruences are unknown.

Practical Computation of the Syntactic ω-Semigroup of a Recognizable ω-Language

Jean Eric Pin
Université Paris VI
Paris Cedex 05, France

The notion of syntactic semigroup is well-known for languages of finite words. The syntactic semigroup of a recognizable (= regular) language given by its (finite) minimal deterministic automaton is easy to compute. There is a similar notion of syntactic congruence, introduced by Arnold, for recognizable ω-languages. This leads to a notion of syntactic ω-semigroup. An ω-semigroup is a two-sorted algebra $S = (S_f, S_\omega)$ equipped with a product $S_f \times S_f \to S_f$, a mixed product $S_f \times S_\omega \to S_\omega$ and an infinite product $S_f^\omega \to S_\omega$. A Ramsey-type argument shows that the finite ω-semigroups are completely determined by their product, mixed product and the ω-power (corresponding to the infinite product $(s, s, s, \ldots) \to s^\omega$). The problem is to effectively compute these objects given a finite non-deterministic Büchi automaton recognizing the regular ω-language. This can be done by computing matrices over the semiring $(\{-\infty, 0, 1\}, max, x)$ where x is defined by $(-\infty) \times x = x \times (-\infty)$, $0 \times 1 = 1 \times 0 = 1 \times 1 = 1$, $0 \times 0 = 0$.

Incremental Model Checking in the Modal Mu-Calculus

Scott A. Smolka[6]
SUNY At Stony Brook
New York, NY, USA

We present an incremental algorithm for model checking in the alternation-free fragment of the modal mu-calculus, the first incremental algorithm for model checking of which we are aware. The basis for our algorithm, which we call MCI (for Model Checking Incrementally), is a linear-time algorithm due to Cleaveland and Steffen that performs global (non-incremental) computation of fixed points. MCI takes as input a set Δ of *changes* to the labeled transition system under investigation, where a change constitutes an inserted or deleted transition; with virtually no additional cost, inserted and deleted states can also be accommodated.

The main technique utilized by MCI is to first compute the immediate effects of the LTS updates on the results of the previous computation and then restart the fixed-point iteration. We show, however, that it is safe to restart the iterations only after making certain adjustments to the current variable assignment. The required adjustments are achieved by making *assumptions* about the existence of strongly connected components in a graph capturing all dependencies between pairs of the form $< s, X_i >$, for LTS state s and logical variable X_i.

[6]This is joint work with Oleg V. Sokolsky.

Like Cleaveland-Steffen, MCI requires time linear in the size of the LTS and logical formula in the worst case, but only time linear in Δ in the best case. We give several examples to illustrate MCI in action, and discuss its implementation in the Concurrency Factory, an interactive design environment for concurrent systems.

The AMORE-System

Wolfgang Thomas & Andreas Potthoff
Christian-Albrechts-Universität zu Kiel
Kiel, Germany

AMORE is a program for computing <u>a</u>utomata, <u>mo</u>noids, and <u>re</u>gular expressions. Its main functions are: conversion of (generalized) regular expressions into minimal deterministic finite automata, the computation of the associated syntactic monoid, its decomposition by Green's relations, tests for language properties (like "nonempty","locally testable","starfree"), and operations on regular languages (like Boolean operations, quotients, shuffle). Experiences with (so far experimental) extensions of the system were reported, concerning Büchi automata, tree automata, cascade decomposition, and transformation into and from temporal logics. In the final part of the talk, the minimization of nondeterministic finite automata was discussed. The algorithm rests on the construction of a canonical "fundamental automaton" $\mathcal{F}$ into which any nondeterministic automaton accepting the considered language can be homomorphically embedded. It is as yet open how the search for a minimal automaton within $\mathcal{F}$ is done most efficiently.

Decidability in Real-Time Systems over a Dense Time-Domain

Carsten Weise
Rheinisch-Westfälische Technische Hochschule Aachen
Aachen, Germany

A sketch of algorithms for the decision of strong and weak bisimulation for real-time systems with a dense time-domain was given. The model used for real-time systems are *timed graphs*, together with *timer evaluation graphs* giving the semantics of timed graphs, as introduced by Dill and Alur.

The basic idea of the decision procedure is to use *mutually extended timer evaluation graphs*, for which bisimulation can be characterized by looking at states with same timer evaluations only. For weak bisimulation, an extension by special τ-steps (*τ-extended timer evaluation graphs*) and adding a new timer T "measuring" the time elapsed since process start (*T-measurability*) are necessary additionally.

Then *timer region graphs*, which are a finite abstraction of the timer evaluation graphs also introduced by Dill and Alur, are used for the decision procedure. All extensions for timer evaluation graphs can easily be defined for timer region graphs as well. On the extended graphs, relations $\bowtie$ and $\bowtie_\tau^Z$ can be given, which characterize strong and weak bisimulation, respectively. These relations relate nodes with identical regions only. Due to the finiteness of timer region graphs the relations are computable.

Grail and its Consequences

Derick Wood
University of Western Ontario
London, Ontario, Canada

Grail is a symbolic manipulation system for expressions, automata, grammars, and other language-theory objects. Currently it supports finite languages, regular expressions, and finite-state automata. Our approach (Darrell Raymond and I) is to provide the basic tools that are needed in all such systems—carefully crafted implementations of the basic operations. For example, it includes minimization and star for finite-state automata; transfer functions from nondeterministic to deterministic automata, from automata to regular expressions, and from regular expressions to automata; and, lastly, predicates that, for example, test whether an automaton is deterministic or a regular expression contains the empty-string symbol. The system is currently written in C++ with extensive use of templates. Each operation is a standalone process with standard in and standard out that communicates with textual representations of the objects. Thus, the standard UNIX utilities can be used.

As a consequence we have been forced to reexamine the standard algorithms for many of the operations since they are often too inefficient to be implemented as is. In addition, Grail has opened up new avenues of theoretical and algorithmic research, has provided a testbed for new algorithms and software engineering principles and is a neverending source of undergraduate and graduate projects.

List of Participants