Dagstuhl Seminar 9743: Applications of Tree Automata in Rewriting, Logic and Programming.

October 20-24, 1997

Schloss Dagstuhl, Germany

Organizers:

Hubert Comon
Ecole Normale Supérieure
Lab. Specification et Verification
61 Ave. du Président Wilson
F-94235 Cachan Cedex
hubert.comon@lsv.ens-cachan.fr

Dexter Kozen
Cornell University,
Dept. of Comp. Sci.
5145 Upson Hall
Ithaca, NY 14850-7501
kozen@cs.cornell.edu

Helmut Seidl
Universität Trier
Fachbereich IV-Informatik
D-54286 Trier
seidl@uni-trier.de

Mosche Y. Vardi
Rice University
Dept. of Comp. Sci.
6100S Main Street
Houston, TX
vardi@cs.rice.edu

1 Overview

The idea of organizing this seminar came during the Federated Logic Conference, New Brunswick, 1996, which brought together the Symposium on Logic in Computer Science, the Conference on Rewriting Techniques and Applications, the Conference on Automated Deduction and the Conference on Computer Aided Verification. We realized there that, though logic was the straightforward intersection between these four communities, there were also common underlying themes and techniques of more specific nature. An important example was, applications of tree automata". This computational model is indeed used in various domains of applications which, to some extent, have ignored each other.

In automated deduction, it seems that general purpose fully automated theorem provers are reaching a limit; they still improve, but the factor of improvement is decreasing. The idea which was raised to combine specialized theorem provers which provide efficient techniques for some specific problems, together with a general purpose prover. An idea would be to use tree automata techniques each time this is possible. For example, equality modulo ground term equations can handled in this way.

Most of the properties of term rewriting systems (like reachability, word problem, sequentiality,...) are undecidable. Hence we are led either to consider only subclasses or to design sufficient conditions guaranteeing these properties. Tree automata play an important role in the design of such classes (or sufficient conditions), which has only been realized recently. For instance, by forgetting (some of) the relations between the variables, we obtain a binary relation which can be recognized by a Ground Tree Transducer. Then all properties of rewrite systems become decidable.

Computer Aided Verification, and especially model checking, became very popular in the recent years because of its security applications; some critical applications might have disastrous consequences if the software managing them is not correct. Several logics and several models have been proposed in the last years. For instance, μ-calculus and CTL on the logical side, and Kripke structures and their unfoldings in tree form on the model side. Already in the sixties, it was shown that monadic second-order formulas (and hence μ-calculus expressions and CTL-formulas) can be converted into automata. The model checking problem was thus reduced to an inclusion test for languages defined by (sequential or tree) automata, possibly consisting also of infinite strings or trees. Although efficient model checking procedures are by now a standard tool, refinements of the theory are needed to overcome several practical problems, notably the, state explosion problem and deficiencies in expressiveness.

Also compiler construction turns out to be a surprisingly wide area of applications for tree automata. These range from backend generation over generation of attribute evaluators to program analysis. In the case of program analysis, the basic idea is to approximate the program by some formal device for which there are tractable algorithms. Tree automata or, equivalently, regular tree grammars and various subclasses of set constraints have attracted attention for such an approximation. While simple forms of set constraints are directly equivalent to finite tree automata, it turns out that more general forms also require advanced tree automata techniques.

In order to stimulate cross-fertilization, this seminar brought together researchers who are involved in such applications. A special session discussed the impact of our results on education. In our opinion, the workshop was a success. Exciting discussions took place until late in the evenings – which has been especially supported by this form of meeting as well as by facilities offered at Dagstuhl, like open access to seminar rooms or the library. By a group of French researchers, a first draft of a basic textbook on tree automata was presented. The community is invited to comment on details or contribute sections.

Acknowledgement: This seminar has been sponsored by the European Community through the TMR program as well as by the National Science Foundation.

We also like to thank Reinhard Wilhelm and the people at Dagstuhl who really made it a pleasure to organize the workshop. We all felt that Dagstuhl did a very good job in providing us with a friendly and stimulating surroundings for our work.

2 Schedule

Monday, Oct. 20, 1997

Morning Session – *Program Analysis:*

Chair:	**D. Kozen**	
9:00 - 10:00	**N.D. Jones**:	Early Applications of Tree Grammars and Tree Automata [overview]
10:00 - 10:30		*Coffee break*
10:30 - 11:05	**M.H. Sørensen**:	Regular Tree Grammars of Deforestation
11:05 - 11:40	**H. Vogler**:	Deforestation Versus Composition
11:40 - 12:15	**J. Palsberg**:	From Polyvariant Flow Information to Intersection and Union Types (joint work with C. Pavlopoulou)
12:15		*Lunch*

Afternoon Session – *Set Constraints*

Chair:	**N.D. Jones**	
2:00 - 3:00	**D. Kozen**:	Set Constraints: An Overview
3:00 - 3:30		*Coffee break*
3:30 - 4:05	**M. Tommasi**:	Set Constraints and Tree Automata (joint work with S. Tison)
4:05 - 4:40	**N. Heintze**:	Using bottom-up Deterministic Tree Automata for Efficient Solving of Set Constraints
4:40 - 4:50		*Coffee break*
4:50 - 5:25	**A. Podelski**:	Set-based Analysis of Concurrent Programs
5:25 - 6:00	**D. McAllester**:	The Greatest Fixed Point of the τ_P Abstraction is Regular (joint work with A. Podelski and W. Charatonik)
6:00		*Dinner*

Tuesday, Oct. 21, 1997

Morning Session – *Model-Checking/Program Analysis*

Chair:	**E.A. Emerson**	
9:00 - 10:00	**M.Y. Vardi**:	Unifying Truth and Validity for Temporal Logic [overview]
10:00 - 10:30		*Coffee break*
10:30 - 11:05	**N. Klarlund**:	Algorithms for Guided Tree Automata
11:05 - 11:40	**H. Seidl**:	Fixpoint Methods for Program Analysis and Model-Checking
11:40 - 12:15	**F. Nielson**:	Contraint-Based Flow Logics (joint work with H.R. Nielson)
12:15 - 2:30		*Lunch*

Afternoon Session – *Model Checking*

Chair:	**M.Y. Vardi**	
2:30 - 3:30	**W. Thomas**:	The Rabin Tree Theorem
3:30 - 4:15		*Coffee break*
4:15 - 4:50	**I. Walukiewicz**:	Automata on Tree-like Structures
4:50 - 5:25	**D. Niwinski**:	Relating Trees and Flowers (joint work with I.Walukiewicz)
5:25 - 6:00	**B. Courcelle**:	Logical Characterization of Recognizability for Finite Graphs
6:00		*Dinner*

Wednesday, Oct. 22, 1997

Morning Session – *Term Rewriting*

Chair:	**H. Ganzinger**	
9:00-10:00	**F. Jacquemard**:	Tree Automata and Term Rewriting: An Overview.
10:00-10:30		*Coffee break*
10:30-11:05	**A. Middeldorp**:	Decidable Call-by-Need Computations in Term Rewriting
11:05-11:40	**A. Bouhoula**:	Automata-Driven Automated Induction (joint work with Jean-Pierre Jouannaud)
11:40-12:15	**D. Lugiez**:	Constrained Automata and Some Applications
12:15		*Lunch*

Afternoon – *Excursion*

Thursday, Oct. 23, 1997

Morning Session – *Education/Term Rewriting*

Chair:	**H. Seidl**	
9:00–9:20	**N. Karlund**:	Teaching Logic, Automata Theory and Decidability to Shophomores
9:20–9:40	**J. Palsberg**:	Automata in Graduate Programming Language Course
9:40–10:00	**R. Wilhelm**:	More Trees than Strings in Compilation
10:00–10:30		Discussion
10:30-11:00		*Coffee break*
11:00–11:35	**G. Kucherov**:	Some Results on Context-free Tree Languages (joint work with D. Hofbauer, M. Huber)
11:35–12:10	**D. Hofbauer**:	Test Sets for Tree Languages
12:10:		*Lunch*

Afternoon Session – *Model-Checking/Rump Session*

Chair:	**I. Walukiewicz**	
14:00–14:35:	**A. Arnold**:	A Selection Property of the Boolean p Calculus
14:35–15:10:	**S. Rohde**:	Alternating Automata and the Temporal Logic of Ordinals
15:10–15:50:		*Coffee break*
15:50–16:25:	**A. Mader** :	Modal μ-Calculus, Model Checking and equivalent problems
16:25–17:00:	**M.Y. Vardi**:	Synthesis with Incomplete Information (joint work with Orna Kupferman)
17:00–17:10		*Short break*
17:10–17:30	**D. Muller**:	On Infinite Trees
17:30–17:45:	**M. Veanes**:	On Simultaneous Rigid E-unification with Restricted Number of Variables.
17:45–18:00	**S. Limet**:	Tree Tupled Synchronized Grammars.
18:00		*Dinner*

Friday, Oct. 24, 1997

Morning Session – *Term Rewriting*

Chair:	**S. Tison**	
9:00-9:35	**H. Comon**:	Higher-Order Matching and Tree Automata
9:35-10:10	**R. Matzinger**:	Using Tree Automata for Computational Representations in Automated Model Building
10:10-10:30		*Coffee break*
10:30-11:05	**M. Steinby**:	On One-Pass Term Rewriting (joint work with Z. Fülöp, E. Jurvanen, and S. Vagvölgyi)
11:05-11:40	**C. Weidenbach**:	Monadic Horn Problems, Tree Automata and Sorted Unification

3 Abstracts

For convenience, the abstracts of the talks have been grouped according to the four main research areas the workshop has been concerned with, namely, Logic and Model-Checking, Term-Rewriting, Program Analysis and Set Constraints. The fifth and last (but clearly equally important) subsection collects the abstracts on statements concerning the impact of tree automata research onto education.

3.1 Logic and Model Checking

Unifying Truth and Validity Checking for Temporal Logics

Moshe Y. Vardi

We describe an automata-theoretic approach to the automated checking of truth and validity for temporal logics. The basic idea underlying this approach is that for any formula we can construct an alternating automaton that accepts precisely the models of the formula. For linear temporal logics the automaton runs on infinite words while for branching temporal logics the automaton runs on infinite trees. The simple combinatorial structures that emerge from the automata-theoretic approach decouple the logical and algorithmic components of truth and validity checking and yield clean and essentially optimal algorithms for both problems.

Further material can be found at `http://www.cs.rice.edu/~vardi/papers`

A Selection Property of the Boolean Mu Calculus

André Arnold

We prove that every closed Boolean mu-term has the same value as a mu-term obtained by replacing each sum by one of its summands.

Logical Characterization of Recognizability for Finite Graphs

Bruno Courcelle

The notion of recognizability for words and trees is usually given in terms of finite automata, and basic results by Büchi and Doner state that a set of words (or of binary trees) is recognizable if and only if it is MS-definable (i.e. definable in Monadic Second-order logic). The concrete meaning is that a property is checkable by a finite automaton iff it is MS-definable. We consider possible the extensions of these notions and results for graphs.

The notion of a recognizable set of graphs is based on graph congruences with finitely many classes and is relative to operations on graphs that, typically, glue two graphs together or extend in some way a given graph.

Already for trees of unbounded degree, MS logic is insufficient to capture recognizability: one needs an extension of MS logic called Counting Monadic Second-order logic (CMS in short). Its formulas are written with special "counting modulo q"-quantifiers meaning that the number of elements x that satisfy a property is a multiple of q.

Result 1: CMS definability (restricting quantifications on sets of vertices in case graphs are simple) is equivalent to recognizability for the class of graphs of tree-width at most 3, and a few related classes (Courcelle, Kaller, Kabanets).

The method for proving these results is as follows. Let C be a class of graphs, let F be the set of graph operations on C involved in the intended notion of recognizability, let us also assume that every graph in C is the value of an F-expression, i.e., of a finite term over F. (The set F is in some sense a parameter: different sets F may yield different notions of recognizability). Assume we have a language L (say an extension of MS like CMS), for which we know that, if a subset of C is L-definable, then it is recognizable. Assume finally that for every graph G in C we can construct "in G" an F-expression that defines this graph. Then, if L is a (F-)recognizable subset of C, there exists a finite tree-automaton recognizing the set of F-expressions the value of which is in L. Given a graph G we can express that G belongs to L by means of a formula in L that works at follows :

(1) it defines in G an F-expression, the value of which is G,

(2) it checks whether the automaton accepts this expression: the graph G is in L if and only if the automaton accepts the expression, if and only if the MS-formula holds.

So the logical language L must not be too powerful (we want that every L-definable class of graphs be recognizable) but it must be powerful enough to do two things

(1) to "parse" the graph (i.e., to define an F-expression for it),

(2) to simulate the behaviour of a finite automaton on the obtained F-expression (which is often a tree of unbounded degree because it handles associative and commutative operations).

For cographs, a built-in (auxiliary and arbitrary) linear ordering of the given graph helps to parse it by MS-formulas.

Result 2: Every graph property that is expressible in MS logic with built-in linear order on the vertices is recognizable. (All properties expressible in CMS logic are of this form: a linear ordering is helpful to express that a set X has an even number of elements; the answer, namely the parity of the cardinality of X does not depend on the chosen ordering of the vertices although it is expressed logically in terms of this order).

Result 3: A set of cographs is recognizable iff it is definable in MS logic with built-in linear order.

Algorithms for Guided Tree Automata

Morten Biehl, Nils Klarlund, and Theis Rauhe

When reading an input tree, a bottom-up tree automaton is "unaware" of where it is relative to the root. This problem is important to the efficient implementation of decision procedures for the Monadic Second-order Logic (M2L) on finite trees. In [Reg=Alg+RDT], it is shown how exponential state space blow-ups may occur in common situations. The analysis of the problem leads to the notion of guided tree automaton for combatting such explosions. The guided automaton is equipped with separate state spaces that are assigned by a top-down automaton, called the guide.

In this paper, we explore the algorithmic and practical problems arising from this relatively complicated automaton concept.

Our solutions are based on a BDD representation of automata [Reg=Alg+RDT], which allows the practical handling of automata on very large alphabets. In addition, we propose data structures for avoiding the quadratic size of transition tables associated with tree automata.

We formulate and analyze product, projection (subset construction), and minimization algorithms for guided tree automata. We show that our product algorithm for certain languages are asymptotically faster than the usual algorithm that relies on transition tables.

Also, we provide some preliminary experimental results on the use of guided automata vs. standard tree automata.

See also: `http://www.brics.dk/~klarlund/MonaFido/papers/`

Modal μ-Calculus Model Checking and Equivalent Problems

Angelika Mader

In this talk the equivalence of problems in four frameworks is worked out: in modal μ-calculus the model-checking problem, for Boolean equation systems the solution of an equation system, for ω-automata with parity acceptance emptiness, and for parity games the existance of a winning strategy for one of the players. The problems can be shown to be contained in NP $\cap$ co-NP, but it is still open, whether they are contained in P. The different means that the four frameworks provide may help to answer this open question.

Relating Trees and Flowers

Damian Niwinski and Igor Walukiewicz

For a language of infinite words L, a derived tree language *Path(L)* is the language of trees having all their paths in L. We consider the hierarchies of deterministic automata on words and nondeterministic automata on trees with Rabin conditions in chain form. We show that L is on some level of the hierarchy of deterministic word automata iff *Path(L)* is on the same level of the hierarchy of nondeterministic tree automata.

We also note that this level can be computed by a polynomial time algorithm from a deterministic automaton recognizing L, by detecting substructures of the automaton's graph that we call flowers.

Full paper available from
`http://zls.mimuw.edu.pl/~niwinski/Prace/kwiatek.ps.gz`

Alternating Automata and the Temporal Logic of Ordinals

Scott Rohde

In their 1995 paper, Muller and Schupp define the concept of an alternating automaton, a sort of completion of the notion of a non-deterministic automaton, and show how the notion may be used to prove a number of major results in the theory of automata on infinite inputs in a unified way. We extend the notion of an alternating automaton to accommodate linear inputs of arbitrary ordinal length and then use these automata to prove a number of results about linear propositional temporal logic, a form of modal logic. First, we show that there is a natural interpretation of automaton inputs as structures for the logic and that under this interpretation, alternating automata and temporal logic are equally powerful. We then use this fact to investigate the satisfiability problem for temporal logic. In particular, not only do we look at the question of whether a given formula has a model, but we look at this question when the lengths of models is restricted to a specific ordinal. We show that the temporal logic of an arbitrary (but fixed) ordinal is decidable in exponential time, generalizing the known result that the temporal logic of ω is decidable in exponential time.

The Rabin Tree Theorem

Wolfgang Thomas

Rabin's Tree Theorem (1969) states that the monadic theory S2S of the infinite binary tree is decidable. The theorem is fascinating both for its wide applicability (it yields a large number of interesting decidability results) and for the difficulty of its proof. There are many more "users" of the result than people who have gone through one of the available proofs.

In this talk we consider the two main steps in Rabin's Tree Theorem, the complementation lemma for Rabin tree automata and the decidability of the emptiness problem. We outline a "modular" proof which seems simple enough to be presentable in a university course. It is based on a new mixture of well-known notions and constructions concerning infinite games, due to Büchi, Gurevich-Harrington, Mostowski, and others.

Synthesis with Incomplete Information

Orna Kupferman and Moshe Y. Vardi

In program synthesis, we transform a specification into a system that is guaranteed to satisfy the specification. When the system is open, then at each moment it reads input signals and writes output signals, which depend on the input signals and the history of the computation so far. The specification considers all possible input sequences. Thus, if the specification is linear, it should hold in every computation generated by the interaction, and if the specification is branching, it should hold in the tree that embodies all possible input sequences.

Often, the system cannot read all the input signals generated by its environment. For example, in a distributed setting, it might be that each process can read input signals of only part of the underlying processes. Then, we should transform a specification into a system whose output depends only on the readable parts of the input signals and the history of the computation. This is called synthesis with incomplete information. In this work we solve the problem of synthesis with incomplete information in its full generality. We consider linear and branching settings with complete and incomplete information. We claim that alternation is a suitable and helpful mechanism for coping with incomplete information. Using alternating tree automata, we show that incomplete information does not make the synthesis problem more complex, in both the linear and the branching paradigm. In particular, we prove that independently of the presence of incomplete information, the synthesis problems for CTL and CTL* are complete for EXPTIME and 2EXPTIME, respectively.

Automata on Tree-like Structures

Igor Walukiewicz

We present some tools for showing decidability and examining expressibility of (monadic second order) theories.

Rabin's definability is a very easy and sometimes sufficiently strong tool. As an example of its application we will show how to define pushdown graphs in the full binary tree (result of D. Caucal). The technique has unfortunately its limitations as already an unwinding of a pushdown graph cannot be coded into the binary tree by means of monadic second order (MSO) formulas.

To overcome this problem we set off to find more structures having decidable MSOL theory. We consider a family of tree-like structures. These are the structures obtained by making a copy of the structure for every finite sequence of elements of the structure, and arranging these copies in a tree. We will recall Muchnik's theorem saying that if MSO-theory of a structure M is decidable then the MSO-theory of the tree-like structure obtained from M is decidable. This in particular will allow us to handle unwindings of pushdown graphs (result of B. Courcelle). We will finish with one more example of Rabin's definability. We will show how one can define the structures $\langle \mathbb{N}, \{n^2 : n \in \mathbb{N}\}\rangle$ and $\langle \mathbb{N}, \{n! : n \in \mathbb{N}\}\rangle$ in tree-like structures with decidable MSO theory.

3.2 Term Rewriting and Automated Deduction

Tree Automata and Applications to Rewriting

Florent Jacquemard

We present a collection of recognizability results for various class of tree automata with/without constraints and show how they can help to solve some problems concerned with term rewriting. The recognition of binary relations on ground terms, especially rewriting relations and the conservation of recognizability under rewriting are also treated.

A postscript version of the slides is available at:
`http://www.mpi-sb.mpg.de/~florent/art/slide-note/dagstuhl.ps.gz`

Automata-Driven Automated Induction

Adel Bouhoula and Jean-Pierre Jouannaud

This work investigates inductive theorem proving techniques for first-order functions whose meaning and domains can be specified by Horn Clauses built up from the equality and finitely many unary membership predicates. In contrast with other works in the area, constructors are not assumed to be free. Techniques originating from tree automata are used to describe ground constructor terms in normal form, on which the induction proofs are built up. Validity of (free) constructor clauses is checked by an original technique relying on the recent discovery of a complete axiomatisation of finite trees and their rational subsets. Validity of clauses with defined symbols or non-free constructor terms is reduced to the latter case by appropriate inference rules using a notion of ground reducibility for these symbols. We show how to check this property by generating proof obligations which can be passed over to the inductive prover.

See also: `http://www.loria.fr/ bouhoula/publications.html`

Tree Automata and Lambda Calculus

Hubert Comon

We consider the "higher-order matching problem" for the simply typed lambda calculus: given two terms u and t, is there a replacement σ of the free variables of u such that u and t become β-equivalent ?

We show actually that, when the order of the free variables of u is smaller or equal to 4, then the set of solutions of the above matching problem is accepted by a finite tree automaton. This allows to derive a new proof of decidability for fourth order matching. We can also derive that third-order matching is NP-complete and give some hint on why 5th order is so hard.

Finally, we will try to relate general higher-order matching and tree automata.

Tree Languages and Test Sets

Dieter Hofbauer and Maria Huber

Test sets are a useful tool when handling the membership problem for the universal closure of tree languages. For a tree language L the universal closure $L^{\forall}$ is the set of those terms whose ground instances are all in L. A test set for the universal closure of L must serve the following purpose: In order to decide whether a term is in $L^{\forall}$ it is sufficient to check whether all its test set instances belong to L.

A possible application – and our main motivation – is ground reducibility: If $Red(R)$ denotes the set of ground terms reducible by a rewrite system R, then its universal closure $Red(R)^{\forall}$ is the set of ground reducible terms. Test sets for ground reducibility always exist and there are several approaches to construct them. The resulting sets, however, often are unnecessarily large.

In this talk we consider regular languages L and linear terms in their closure. Here, membership for $L^{\forall}$ (being regular as well) is decidable, and test sets always exist. By relating test sets to tree automata, we solve the following problems:

- How to characterize test sets?
- How to compute minimal test sets?
- How to minimize given test sets?

Some Results on Context-free Tree Languages

Gregory Kucherov, Dieter Hofbauer and Maria Huber

First, we study the class of top-context-free tree languages (called *corégulier* by Arnold and Dauchet (1976)) – a natural subclass of context-free tree languages. In particular, we concentrate on the yet more narrow class of regular top-context-free languages and give various characterizations of this class. We also study closure properties of top-context-free tree languages and give a criterion for a language to be not top-context-free.

In the second part, we talk about projection (erasing) rules in general context-free tree grammars. It is known that in contrast to the word case, in a context-free tree grammar these rules cannot always be eliminated. We show, however, that an equivalent grammar can be constructed such that application of these rules can be avoided for any given set of positions. As an immediate consequence we get a new direct proof for the decidability of the membership problem for context-free tree languages.

Constrained Automata and Some Applications

Denis Lugiez

We define a new class of constrained tree automata to deal with terms containing syntactical functions and interpreted ones. First we define an abstract class of automata with constrained rules which enjoys boolean closure properties, then we specialize the class for the case of multisets and the case of non-negative integers. In each case we give a decision procedure for the emptiness of the language accepted by an automaton. We conclude by showing how these automata can be applied to inductive theorem proving for testing sufficient completeness and inductive reducibility.

Full paper available from: `http://www.loria.fr/~lugiez`

Using Tree Automata for Computational Representations in Automated Model Building

Robert Matzinger

Automated Model Building - the attempt to automatically construct models for formulas that are found to be satisfiable - raises increasing interest in the automated deduction community. Clearly, representing particular models of first-order formulas in a computational feasible way is an important prerequisite for Automated Model Building, but plays a role in many other fields too, e.g. semantic resolution, model checking, etc. As the specification of a Herbrand model is nothing else than the specification of a (potentially infinite) set of atoms, we are lead to investigating model properties in terms of syntactical properties of the true ground atom set. Tree automata and their relatives turn out to be a major tool for such an investigation. Therefore I will try to advertise automated model building and model representations as an interesting field for applying tree automata theory.

In particular I start with giving an introduction to automated model building and to computational requirements in this field, and I show that with relatively simple means we can already obtain interesting results about representable models, e.g. I show how Herbrand models can be represented with regular tree automata.

Computational aspects of this approach are considered and the class of models representable by this approach is characterized. I survey some key approaches for automated model building. In particular I introduce the key ideas of the "clauses with constraints"-approach of Caferra, Zabel, Peltier, et al. and the "choose and resolve"-approach of Fermueller, Leitsch et al., both valuable work of people not present at the workshop. I compare the models generated with these methods and relate them to tree automata representations, concluding with hints to open problems and future ideas.

See also: `http://www.kr.tuwien.ac.at/~matzi`

Decidable Call-by-Need Computations in Term Rewriting

Aart Middeldorp and Irene Durand

The following theorem of Huet and Levy forms the basis of all results on optimal normalizing reduction strategies for orthogonal term rewriting systems (TRSs): every reducible term contains a needed redex and repeated contraction of needed redexes results in a normal form, if the term under consideration has a normal form. Unfortunately, needed redexes are not computable in general. Hence, in order to obtain a computable optimal reduction strategy, we are left to find

(1) decidable approximations of neededness and

(2) decidable properties of TRSs which ensure that every reducible term has a needed redex identified by (1).

Starting with the seminal work of Huet and Levy on strong sequentiality, these issues have been extensively investigated in the literature. In all these works Huet and Levy's notions of index, omega-reduction, and sequentiality figure prominently.

We present an approach to decidable call by need computations to normal form in which issues (1) and (2) above are addressed directly. Besides facilitating understanding this enables us to cover much larger classes of TRSs. For instance, an easy consequence of our work is that every orthogonal right-ground TRS admits a computable call by need strategy whereas none of the sequentiality-based approaches cover all such TRSs.

Full paper available from
`http://www.score.is.tsukuba.ac.jp/~ami/papers/dcbn.dvi`

On One-Pass Rewriting

Magnus Steinby

Two restricted ways to apply a term rewriting system (TRS) to a tree are considered. When the one-pass root-started strategy is followed, rewriting starts from the root and continues stepwise towards the leaves without ever rewriting a part of the current tree produced in a previous step. One-pass leaf-started rewriting is defined similarly, but rewriting begins from the leaves. In the sentential form inclusion problem one asks whether all trees obtained with a given TRS from the trees of some regular tree language T belong to another given regular tree language U, and in the normal form inclusion problem the same question is asked about the normal forms of T. We show that for a left-linear TRS these problems can be decided for both of our one-pass strategies. In all four cases the decision algorithm involves the construction of a suitable tree recognizer.

Monadic Horn Problems, Tree Automata and Sorted Unification

Christoph Weidenbach

We study the complexity of sorted unifiability, the complexity of emptiness and the cardinality of complete sets of well-sorted unifiers with respect to order-sorted, shallow and ordered sort theories. These sort theories are closely related to regular bottom-up tree automata, automata with (dis)equality constraints between brother terms and generalized encompassment automata. We investigate similarities and differences between these approaches by a relativization of both frameworks to monadic Horn problems. Finally, we show that the extension of shallow sort theories by shallow equations still yields a decidable unification (emptiness) problem.

3.3 Program Analysis

Early Applications of Tree Grammars and Tree Automata for Program Analysis and Compiling

Neil D. Jones

An approach using tree grammars to analyze programs manipulating tree-structured data was developed by Jones and Muchnick in 1978. Correctness (more properly safety) was expressed by reachability: the set of all possible values assigned to variables at each program point is a subset of the set of values generated by a nonterminal in the tree grammar. A key to constructing the grammar was a technique from regular canonical systems (Büchi, 1964). Just as the work was completed, it was discovered that Reynolds had similar results for functional programs in a much overlooked article from 1968.

Extensions of this idea were used for flow analysis of: lambda expressions (ICALP 81); flexible interprocedural analysis (POPL 82); higher-order lazy programs (Abramsky and Hankin collection, 1987); and by several researchers since then.

Tree transducers (actually, derivors) were applied to compiler generation (and not just compiler construction) in the CERES system, developed by Jones and Tofte and the subject of Tofte's EATCS Monograph. Key ideas included expressing a derivor itself as a tree; and "self-composition" to transform a tree transducer defining a language's semantics into one defining a compiler for the same language, also by means of a derivor. Annette Wagner (a student of Ganzinger) extended the self-composition concept to apply to Ganzinger and Giegerich's "attribute-coupled grammars". Self-composition was a precursor to the idea of "generating extension", as used in compiler generation by partial evaluation.

Constraint Based Flow Logics

Flemming Nielson and Hanne Riis Nielson

This talk gives an overview of how constraints over finite sets, elements of complete lattices, and languages may arise when performing program analysis. We first cover syntax-directed approaches to constraint based Control Flow Analysis and illustrate the relationship to the equational formulation of Data Flow Analysis; next we cover abstract approaches motivated by Abstract Interpretation and observe the need for coinductive methods. Then we augment the Control Flow Analysis with components from Data Flow Analysis: a value based analysis using complete lattices, and an analysis tracking the calling history using formal languages. We conclude with discussing the need to "overcome" the undecidability of many questions related to formal languages.

From Polyvariant Flow Information to Intersection and Union Types

Jens Palsberg and Christina Pavlopoulou

Many polyvariant program analyses have been studied in the 1990s, including k-CFA, poly-k-CFA, and the cartesian product algorithm. The idea of polyvariance is to analyze functions more than once and thereby obtain better precision for each call site. In this talk we present the first formal relationship between polyvariant analysis and standard notions of type. In the spirit of Nielson and Nielson, we study a parameterized flow analysis which can be instantiated to the analyses of Agesen, Schmidt, and as a simple case also 0-CFA. Extended with safety checks, the flow analysis accepts and rejects programs, much like a type checker. We prove that if a program can be safety-checked by a finitary instantiation of the flow analysis, then it can also be typed in a type system with intersection types, union types, subtyping, and recursive types, but no universal or existential quantifiers. This provides a framework for designing and understanding combinations of flow analyses and type systems.

Full paper available from
`http://www.cs.purdue.edu/homes/palsberg/draft/palsberg-pavlopoulou97.ps.gz`

Fixpoint Methods for Program Analysis and Model-Checking

Helmut Seidl

We review current developments in the area of local fixpoint iterators. We apply these methods to obtain efficient interprocedural analyzers for procedural languages, fast control-flow analyzers as well as fast model-checkers for the full μ-calculus.

Further reading can be found at
`http://www.informatik.uni-trier.de/~seidl/publications.html`

Program Transformers and Tree Automata

Morten Heine Sørensen

This talk presents a brief overview of how such notions as set constraints, tree grammars, and tree automata arise naturally in several lines of work on transformation techniques for functional programs.

It is then argued that a deeper understanding of such program transformers may be acquired by emphasizing the use of, e.g., tree automata. For example, Wadler's deforestation algorithm, which eliminates intermediate data structures from functional programs, turns out to be very directly related to the product construction for composing deterministic top-down tree transducers, and this equivalence sheds some new light on deforestation and the class of treeless program, for which deforestation is known to terminate.

Benefits of Hypergraphs for Program Transformations

Andrea Mößle and Heiko Vogler

We show an automatic program transformation strategy which allows to remove multiple calls of functions on the same argument. The transformation strategy works on every constructor-based term rewriting system. In particular, we indicate that using the framework of hypergraphs makes the presentation of the transformation strategy very easy. The resulting term rewriting system M is at least as efficient as the original one N. Moreover, it is decidable whether M is more efficient on infinitely many arguments than N.

3.4 Set Constraints

Set Constraints: An Overview

Dexter Kozen

Set constraints are equations between expressions denoting sets of elements of an algebraic structure, usually ground terms. They have been used extensively in program analysis and type inference for many years. Considerable recent effort has focussed on the computational complexity of the satisfiability problem. Set constraints have also recently been used to define a constraint logic programming language over Herbrand domains.

Set constraints exhibit a rich mathematical structure. There are strong connections to automata theory, type theory, first-order monadic logic, Boolean algebra with operators, and modal logic. There are algebraic and topological formulations, corresponding roughly to "soft" and "hard" typing respectively, which are related by Stone duality.

In this survey talk I will define the general set constraint satisfaction problem and describe several applications; outline the connection to tree automata; discuss how the automata-theoretic formulation is used to obtain complexity results; and discuss algebraic and topological formulations and their relationship.

The Greatest Fixed Point of the τ_p Abstraction is Regular

David McAllester

In a well known POPL paper in 1990 Heintze and Jaffar introduced the τ_p abstraction of logic programs. This abstraction can be phrased as a simple source to source transformation mapping logic program P to program P' where the meaning of P', i.e., the set of atoms provable from P', is a regular set, i.e., a set definable by a regular tree automata, and contains the meaning of P. Here we show that the τ_p abstraction can be used for analyzing greatest fixed points as well least fixed points — the greatest fixed point of P' is a regular set containing the greatest fixed point of P. This fact can be used to verify certain liveness conditions by proving that a "nontermination predicate" is empty in a greatest fixed point.

Set-based Analysis of Reactive Programs

Andreas Podelski

We present several ideas which together yield a verification method for reactive programs via set-based analysis. CTL-properties are described by least or greatest models of logic programs (and logic programs describe transition systems, possibly with an infinite number of states). The set-based approximation of the least or greatest model of a logic program can be computed by the iteration of a functional over sets of Horn clauses (in single exponential time). Various kinds of automata over infinite trees (which are used to encode various kinds of temporal logic properties) correspond to various classes of logic programs. For example, weak alternating Büchi automata correspond to stratified logic programs (without negation, but with calls to the greatest model which corresponds to double negation). Rabin tree automata correspond to logic programs with atoms marked by priorities (the Horn μ-calculus).

More information to be found at `http://www.mpi-sb.mpg.de/~podelski`

Set Constraints and Tree Automata

Sophie Tison and Marc Tommasi

Generalized Tree Set Automata (GTSA) are recognizers of mappings from the Herbrand Universe to a finite set of labels. When the set of labels is $\{0,1\}^n$, GTSAs accept tuples of tree languages.

This new kind of automaton has been introduced to solve positive and negative set constraints. However, because of the high complexity of the satisfiability problem of solving set constraints, it is interesting for some special classes to build directly some special regular solutions.

In this talk, we will present GTSAs, some important properties that they fulfill, and their application to set constraints. In a second part, we will expose an algorithm for constructing the least (resp. the greatest) solution of (resp. co-)definite set constraints. This algorithm is based on tree automata and the computed solution is regular.

3.5 Education

Sophomores, Logics, and Automata

Nils Klarlund

During a second-year course on fundamental models in computer science, I tried to link logic, automata, concurrency, and verification. The goal was to make clearer to the students that logic and automata have computational meanings. To do this, I showed how mutual exclusion protocols can be modeled in an automata-theoretic framework. I also showed how first-order logic is useful in formalizing desired properties protocols. The main theoretical idea gluing automata and properties together is the connection between automata and monadic second-order logic on finite strings—admittedly a big mouthful for sophomore students.

I have drawn two conclusions,

- The automaton-logic connection should be taught to sophomores, but only as a decision procedure for Propositional Logic (and maybe also for Quantified Propositional Logic). There are many interesting points to be made about such a procedure relative to the naive algorithm by enumeration. And there is a strong practical motivation: BDDs, which are very similar to automata representations of the satisfying truth assignments, have become very important in the development and testing of digital hardware.

- Logic and automata should be illustrated early on by easy-to-use, interactive programs that hammer in the fact that these concepts are firmly rooted in computation and programming. Too large a fraction of students feels very estranged when these matters are taught. (WS1S, which is a fragment of arithmetic, could be a useful logic, since its decision procedure allows immediate answers to simple questions about natural numbers.)

Automata in a Graduate Programming Language Course

Jens Palsberg

Tree automata plays an important role in the required graduate course on programming languages at Purdue University. Such automata are used to represent types. The students learn and implement subtyping, type inference, and mappings between types and flow using that representation. The representation is particularly useful for Java interfaces which can be recursive.

Applications of Tree Automata in Compilation

Reinhard Wilhelm

Two applications of tree automata in compilers are described, one in the strategy phase of attribute evaluation, the other in code selection.

The evaluation of semantic attributes usually happens in two phases; the strategy phase determines an evaluation order for the attributes, and the evaluation phase proper computes the attribute values according to the evaluation order. Both phases are automatically generated from an attribute grammar. The following preprocessing happens at compiler generation time. A set of lower and upper global dependencies for each nonterminal are computed from the local attribute dependencies of productions. Evaluation plans for the productions are computed for all combinations of such global dependencies. At compile time, the global dependencies for the given tree are determined by a deterministic bottom up tree automaton and a deterministic top down tree automaton. Their results determine the evaluation plans at the productions instances in the tree.

Code selection for a given target machine from an intermediate representation (IR) of programs can be described by a regular tree grammar. Nonterminals in the grammar are recources of the target machine, terminals operators of the IR. A powerset construction generates a deterministic bottom up tree automaton. This automaton annotates the tree with its states from which the applied productions can be derived. Goal is the selection of a "cheapest" derivation corresponding to a (locally) optimal code sequence. The selection of a cheapest derivation can be done by running a dynamic programming method over the tree. Hwever, the selection of the cheapest derivation can also be integrated into the automaton if the cost measure satisfies the following criterium: the differences in cost of the cheapest and all other derivations are bounded by a constant. The powerset construction terminates under this condition.

Both applications are described in the text book:
R. Wilhelm, D. Maurer: Compiler Design, Addison Wesley, 1997

Participants of Seminar 9743:

Prof. Dr. André Arnold
Université Bordeaux I
Laboratoire Bordelais de
Recherche en Informatique
351 Cours de la Libération
F-33405 Talence Cedex
France
arnold@labri.u-bordeaux.fr
tel.: +33-5-56.84.60.94
fax.: +33-5-56.84.66.69

Prof. Dr. David Basin
Universität Freiburg
Institut für Informatik
Geb. 52 - Zi. 00-017
Am Flughafen 17
79110 Freiburg
Germany
basin@infomatik.uni-freiburg.de
tel.: +49-761-203-82 40
fax.: +49-761-203-82 42

Adel Bouhoula
INRIA Lorraine and CRIN
BP 101
F-54602 Villers-lès- Nancy Cedex
France
Adel.Bouhoula@loria.fr
tel.: +33-3-83 59 30 55
fax.: +33-3-83 27 83 19

Dr. Witold Charatonik
MPI - Saarbrücken
Im Stadtwald
D-66123 Saarbrücken
Germany
witold@mpi-sb.mpg.de

Prof. Dr. Hubert Comon
Ecole Normale Supérieure
Lab. Specification et Verification
61 Ave. du Président Wilson
F-94235 Cachan Cedex
France
hubert.comon@lsv.ens-cachan.fr
tel.: +33-1-47.40.24.30
fax.: +33-1-47.40.24.04

Prof. Dr. Bruno Courcelle
Université Bordeaux - I
LaBRI
351 Cours de la Libération
33405 Talence Cedex
France
courcelle@labri.u-bordeaux.fr
tel.: +33-5-56 84 60 86
fax.: +33-5-56 84 66 69

Prof. Dr. E. Allen Emerson
University of Texas
Department of Computer Sciences
Taylor Hall 2.124
Austin TX 78712-1188
USA
emerson@cs.utexas.edu
tel.: +1 512 471 9537
fax.: +1 512 471 8885

Dr. Kathi Fisler
Rice University
Dept. of Computer Science
Mail Stop 132
6100 S. Main. St.
Houston TX 77005-1892
USA
kfisler@cs.rice.edu
tel.: +1-713-527-8750 x 3834
fax.: +1-713-285-5930

Prof. Dr. Harald Ganzinger
Max-Planck-Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken
Germany
hg@mpi-sb.mpg.de
tel.: +49-681-9325-201
fax.: +49-681-9325-299

Dr. Remi Gilleron
Université de Lille I
LIFL
Bâtiment M 3
F-59655 Villeneuve d'Ascq Cedex
France
gilleron@lifl.fr
tel.: +33-3-20.41.61.78
fax.: +33-3-20.43.65.66

Dr. Nevin Heintze
BELL Laboratories
600 Mountain Ave.
Murray Hill NJ 07974
USA
nch@research.bell-labs.com
tel.: +1-908-582-6419

Dr. Dieter Hofbauer
TU Berlin
Theoretische Informatik
Algorithmik und Logik - FR 6-2
Franklinstrasse 28-29
10587 Berlin
Germany
dieter@cs.tu-berlin.de
tel.: +49-30-314-22624

Maria Huber
Universität-Gesamthochschule
FB 17 - Informatik
HPS
34109 Kassel
Germany
maria@theory.informatik.uni-kassel.de
fax.: +49-561-804-4008

Dr. Florent Jacquemard
Max-Planck-Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken
Germany
florent@mpi-sb.mpg.de
fax.: +49-681-9325-299

Prof. Dr. Neil D. Jones
University of Copenhagen
Department of Computer Science / DIKU
Universitetsparken 1
DK-2100 Copenhagen
Denmark
neil@diku.dk
tel.: +45-35 32 14 10
fax.: +45-35 32 14 00

Prof. Dr. Nils Klarlund
AT & T Research
600 Mountain Avenue
Murray Hill NJ 07974-0636
USA
klarlund@research.att.com
fax.: +1-908-582-7550

Prof. Dr. Dexter Kozen
Cornell University
Dept. of Computer Science
5143 Upson Hall
Ithaca NY 14853 -7501
USA
kozen@cs.cornell.edu
tel.: +1-607-255-5143
fax.: +1-607-255-9209

Gregory Kucherov
CRIN & INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
kucherov@loria.fr

Sebastien Limet
Université d'Orléans
LIFO
B.P. 6759
F-45067 Orléans Cedex 2
France
limet@lifo.univ-orleans.fr
tel.: +33-2-38 49 47 22
fax.: +33-2-38 41 71 37

Denis Lugiez
CRIN & INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
lugiez@loria.fr
tel.: +33-3-83.59.30.20

Dr. Angelika Mader
University of Nijmegen
Computing Science Institute
P.O. Box 9010
NL-6500 GL Nijmegen
The Netherlands
mader@cs.kun.nl

Oliver Matz
Universität Kiel
Inst. für Informatik und
Prakt. Mathematik, Haus I
Olshausenstrasse 40
D-24098 Kiel
Germany
oma@informatik.uni-kiel.de
tel.: +49-431-880-4389

Robert Matzinger
TU Wien
Institut 184/3
Treitlstr. 3
A-1040 Wien
Austria
matzi@kr.tuwien.ac.at
tel.: +43-1-588-0181 96
fax.: +43-1-581-79 66

Dr. David McAllester
AT & T Labs-Research
180 Park Avenue
Florham Park NJ 07932-0971
USA
dmac@research.att.com

Prof. Dr. Aart Middeldorp
University of Tsukuba
Institute of Information Sciences
and Electronics
Tennodai 1-1-1
Tsukuba 305
Japan
ami@score.tsukuba.ac.jp
tel.: +81-298-53-55 38
fax.: +81-298-53-52 06

Prof. Dr. David Muller
New Mexico State Univ. - Las Cruces
4200 Sotol Drive
Las Cruces NM 88011
USA
dmuller@nmsu.edu
tel.: +1-505-522-5738
fax.: +1-505-522-5738

Prof. Dr. Flemming Nielson
Aarhus University
Dept. of Computer Science
Bldg. 540
Ny Munkegade
DK-8000 Aarhus C
Denmark
fnielson@daimi.aau.dk
tel.: +45-89423363
fax.: +45-89423255

Prof. Dr. Maurice Nivat
Université Paris VII
LIAFA
2 Place Jussieu
F-75252 Paris Cedex 05
France
nivat@litp.ibp.fr
tel.: +33-1-44 27 68 45
fax.: +33-1-44 27 68 49

Dr. Damian Niwinski
University of Warsaw
Inst. of Informatics
Ul. Banacha 2
02-097 Warsaw
Poland
niwinski@mimuw.edu.pl
tel.: +48 22-658-4377
fax.: +48 22-658-3164

Prof. Dr. Jens Palsberg
Purdue University
Department of Computer Sciences
West Lafayette IN 47907-1398
USA
palsberg@cs.purdue.edu
tel.: +1-765-494-6012
fax.: +1-765-494-0739

Dr. Andreas Podelski
Max-Planck Institut für Informatik
Im Stadtwald
66123 Saarbrücken
Germany
podelski@mpi-sb.mpg.de
tel.: +49-681-9325-204
fax.: +49-681-9325-299

Prof. Dr. Hanne Riis Nielson
Aarhus University
Dept. of Computer Science
Bldg. 540
Ny Munkegade
DK-8000 Aarhus C
Denmark
hrn@daimi.aau.dk
tel.: +45-89 42 32 76
fax.: +45-89 42 32 55

Scott Rohde
2 Shuman Circle.
Urbana IL 61801-6219
USA
rohde@math.uiuc.edu
tel.: +1-217-344-3126

Dr. Michaël Rusinowitch
CRIN & INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
rusi@loria.fr
tel.: +33-3-83.59.30.20

Prof. Dr. Helmut Seidl
Universität Trier
Fachbereich IV - Informatik
LST fr Programmiersprachen und Compiler
54286 Trier
Germany
seidl@uni-trier.de
tel.: +49-651-201-2835
fax.: +49-651-201-3822

Dr. Morten Heine Sørensen
University of Copenhagen
Department of Computer Science / DIKU
Universitetsparken 1
DK-2100 Copenhagen
Denmark
rambo@diku.dk
tel.: +45-35 32 14 05
fax.: +45-35 32 14 01

Prof. Dr. Magnus Steinby
University of Turku
Dept. of Mathematics
20014 Turku
Finland
steinby@utu.fi
tel.: +358-2-633 5618

Jean-Marc Talbot
Université de Lille I
LIFL - USTL
Bâtiment M 3
F-59655 Villeneuve d'Ascq Cedex
France
talbot@lifl.fr
tel.: +33-3-20 43 47 18
fax.: +33-3-20 43 65 66

Prof. Dr. Wolfgang Thomas
Universität Kiel
Inst. für Informatik und
Prakt. Mathematik, Haus I
Olshausenstrasse 40
D-24098 Kiel
Germany
wt@informatik.uni-kiel.de
tel.: +49-431-880-4480
fax.: +49-431-880-4054

Prof. Dr. Sophie Tison
Université de Lille I
LIFL
Bâtiment M 3
F-59655 Villeneuve d'Ascq Cedex
France
tison@lifl.fr
fax.: +33-3-20.43.65.66

Dr. Marc Tommasi
Université de Lille I
LIFL
Bâtiment M 3
F-59655 Villeneuve d'Ascq Cedex
France
tommasi@lifl.fr
tel.: +33-3-20.43.42.55
fax.: +33-3-20.43.65.66

Dr. Ralf Treinen
Université de Paris Sud
LRI
Bt. 490
F-91405 Orsay Cedex
France
ralf.treinen@lri.fr
tel.: +33-1-69.15.65.92
fax.: +33-1-69.15.69.86

Prof. Dr. Moshe Y. Vardi
Rice University
Dept. of Computer Science
Mail Stop 132
6100 S. Main. St.
Houston TX 77005-1892
USA
vardi@cs.rice.edu
tel.: +1-713-285-5977
fax.: +1-713-285-5930

Margus Veanes
Max-Planck-Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken
Germany

Prof. Dr. Heiko Vogler
TU Dresden
Fakultät für Informatik
Institut fr Softwaretechnik I
D-01062 Dresden
Germany
vogler@inf.tu-dresden.de
tel.: +49-341-463-8232
fax.: +49-341-463-8504

Dr. Sergei Vorobyov
MPI - Saarbrücken
Im Stadtwald
D-66123 Saarbrücken
Germany

Dr. Igor Walukiewicz
University of Warsaw
Institute of Informatics
Ul. Banacha 2
02-097 Warszawa
Poland
igw@mimuw.edu.pl
tel.: +48-22-658-31 65
fax.: +48-22-658-31 64

Christoph Weidenbach
Max-Planck-Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken
Germany
weidenb@mpi-sb.mpg.de
tel.: +49-681-9325-221
fax.: +49-681-9325-299

Prof. Dr. Reinhard Wilhelm
Universität des Saarlandes
Fachbereich 14 - Informatik
Postafach 15 11 50
66041 Saarbrücken
Germany
wilhelm@cs.uni-sb.de
tel.: +49-681-302 3434
fax.: +49-681-302 4421