Report from Dagstuhl Seminar 13192

Tree Transducers and Formal Methods

Edited by

Sebastian Maneth[1] and Helmut Seidl[2]

1   **University of Oxford, GB, sebastian.maneth@gmail.com**
2   **TU München, DE, seidl@in.tum.de**

—— Abstract ——

The aim of this Dagstuhl Seminar was to bring together researchers from various research areas related to the theory and application of tree transducers. Recently, interest in tree transducers has been revived due to surprising new applications in areas such as XML databases, security verification, programming language theory, and linguistics. This seminar therefore aimed to inspire the exchange of theoretical results and information regarding the practical requirements related to tree transducers.

1   Executive Summary

Sebastian Maneth

The Dagstuhl seminar 13192 "Tree Transducers and Formal Methods" was a short two and a half day seminar that took place from May 5th to 8th, 2013. The aim was to bring together researchers from various research areas related to the theory and application of tree transducers. Tree transducers are a classical formalism in computer science, dating back to the early days of compilers and syntax-directed translation. Recently, interest in tree transducers has been revived due to surprising new applications in areas such as XML databases, security verification, programming languages, and linguistics. This seminar was meant to inspire the exchange of theoretical results and practical requirements related to tree transducers. These points were addressed in particular:

- Expressiveness versus Complexity: Which transducers offer the best trade-offs between expressiveness and complexity?
- Implementability under Resource Restrictions: Which transducer models can be executed by devices with bounded resources, e.g., for processing XML streams?
- New Applications: What new challenges do the different application areas of tree transducers raise? What new solutions have been found?
- Open Problems: Which are the most pressing open problems in tree transducer theory?

The seminar fully satisfied our expectations. The 33 participants from 13 countries (Australia, Belgium, Canada, Czech, France, Germany, Great Britain, Hungary, Japan, Poland, Slovakia, Sweden, and the US) had been invited by the organizer Sebastian Maneth to give particular survey talks about their recent research on applications and theory of tree transducers.

There were talks focusing on very practical issues such as Margus Veanes' talk on software verification using symbolic tree transducers (which kicked off the meeting), and also talks on highly challenging theoretical results such as the talk by Emmanuel Filiot on their recent breakthrough of proving that one-wayness of a two-way word automaton is decidable. The other application areas, besides verification, were (1) tree processing (related to databases and search) (2) learning, and (3) linguistics.

The first talk by Veanes on symbolic transducers was followed by Jan Janousek about using pushdown automata to search for tree patterns, in linear order of trees. Symbolic transducers, from a theoretical point of view, were discussed in Heiko Vogler's talk in the afternoon. Input driven pushdown automata, also known as nested word automata or visibly pushdown automata, were discussed with respect to descriptional complexity by Kai Salomaa. The second morning session of the first day was devoted to MSO translations, first about its theory with respect to word and tree translations by Bruno Courcelle, and then concerning a one-pass and linear time implementation model for MSO tree translations: the streaming tree transducer by Loris d'Antoni. The first afternoon section was about higher-order transducers, recursion schemes, and verification, given by Kazuhiro Inaba, Luke Ong, and Naoki Kobayashi. They discussed the open problem of proving context-sensitivity of the unsafe OI-hierarchy, results on model checking of higher-order recursion schemes, and practical approaches to type checking unsafe higher-order tree transducers.

The second day started with theoretical results about word and tree transducers by Emmanuel Filiot and Sebastian Maneth. The latter one was about deciding two database notions, namely determinacy and rewriting, for top–down and MSO tree transducers. Next was a sequence of talks about streaming, by Joachim Niehren, Pavel Labath, and Keisuke Nakano. They discussed practical aspects of early query answering, streaming of macro tree transducers using parallel streams, and stack attributed tree transducers, respectively. Related to streaming was the following talk by Frederic Servais which surveyed recent results on visibly pushdown transducers. The following three talks discussed learning algorithms: first about tree series by Johanna Björklund and Frank Drewes, and then about top–down tree transformations by Adrien Boiret. The last talk of the second day was Florent Jacquemard and Sophie Tison's survey about tree automata with constraints.

The final day started with a talk about natural language processing using transducers, given by Daniel Gildea. It presented applications of multi bottom-up tree transducers to machine translation of natural language. It was followed by a talk by Uwe Mönnich on logical definitions of mildly context-sensitive grammar formalisms. A survey on "the tree-based approach" to natural language grammars was given by Marco Kuhlmann. Damian Niwinski's talk connected to the session of the first day on higher-order schemes: they are equivalent to panic automata, the invention and topic of Damian. An important practical consideration is incremental evaluation: it was discussed for XPath by Henrik Björklund and for succinct regular expressions by Wim Martens.

We thank Schloss Dagstuhl for the professional and inspiring atmosphere it provides. Such an intense research seminar is possible because Dagstuhl so perfectly meets all researchers' needs. For instance, elaborate research discussions in the evening were followed by musical intermezzi of playing piano trios by Beethoven and Mozart, or by table tennis matches and sauna sessions.

2   Table of Contents

3 Overview of Talks

3.1 Incremental XPath Evaluation

Henrik Björklund (University of Umeå, SE)

Incremental view maintenance for XPath queries asks to maintain a materialized XPath view over an XML database. It assumes an underlying XML database D and a query Q. One is given a sequence of updates U to D, and the problem is to compute the result of $Q(U(D))$: the result of evaluating query Q on database D after having applied updates U. This article initiates a systematic study of the Boolean version of this problem. In the Boolean version, one only wants to know whether $Q(U(D))$ is empty or not. In order to quickly answer this question, we are allowed to maintain an auxiliary data structure. The complexity of the maintenance algorithms is measured in, (1) the size of the auxiliary data structure, (2) the worst-case time per update needed to compute $Q(U(D))$, and (3) the worst-case time per update needed to bring the auxiliary data structure up to date. We allow three kinds of updates: node insertion, node deletion, and node relabeling. Our main results are that downward XPath queries can be incrementally maintained in time $O(\text{depth}(D)\cdot\text{poly}(|Q|))$ per update and conjunctive forward XPath queries in time $O(\text{depth}(D)\cdot\log(\text{width}(D))\cdot\text{poly}(|Q|))$ per update, where $|Q|$ is the size of the query, and $\text{depth}(D)$ and $\text{width}(D)$ are the nesting depth and maximum number of siblings in database D, respectively. The auxiliary data structures for maintenance are linear in $|D|$ and polynomial in $|Q|$ in all these cases.

References

1 Henrik Björklund, Wouter Gelade, and Wim Martens. Incremental XPath Evaluation. *ACM Transations on Database Systems*, Vol. 35, No. 4, Article 29, November 2010.

3.2 Learning Deterministic Top–Down Tree Transducers with Inspection

Adrien Boiret (Université Lille, FR)

We present a Myhill–Nerode result on deterministic top–down tree transducers, an extension of the preexisting result by Lemay, Maneth, Niehren in 2010.

This result gives us a minimal normal form, and then allows us to devise a learning algorithm on the Gold model (Gold 1978), using behavior examples as an input, and providing the minimal normal form as the output.

3.3   Learning Tree Series

Frank Drewes (University of Umeå, SE)

We give an introduction to the basic ideas of active learning, focusing on the learning of tree languages and tree series from a so-called minimal adequate teacher.

In active learning, the learning algorithm is allowed to request training data by need. The paradigm was designed to improve accuracy and reduce annotation effort, and is particularly appropriate when data is easy to come by, but labelling it is expensive. The labelling resource is typically modelled as an oracle capable of answering certain kinds of queries. The blueprint of such a resource is Angluin's minimal adequate teacher (MAT) which accepts membership queries and equivalence queries [1]. In a membership query, the oracle is asked to label an element as inside our outside the target language L. In an equivalence query, the oracle is given a language model M (e.g., a finite automaton or a grammar) and is expected to return a counter-example to the conjecture that $L(M) = L$, or to acknowledge that L has been correctly acquired.

In this talk, we discuss the inference of tree languages and, more generally, tree series within the MAT model. A tree series is a function from a domain of trees to some algebraic structure, often a semiring or semifield. We focus on the generalisation of Angluin's LSTAR algorithm to trees, and explain central tools and techniques such as observation tables, contradiction backtracking, and proof-of-life contexts by means of an extensive example that covers the main ideas of [2, 3, 4]. The talk concludes with a summary of related results, in which one or more of the components (i) language model, (ii) target class, and (iii) oracle definition has been altered. In doing so, we touch upon the learnability of tree transducers, residual and universal automata, and variations on the classical MAT queries such as correction queries and inclusion queries.

References

1   Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.
2   Frank Drewes and Johanna Högberg. Query learning of regular tree languages: How to avoid dead states. *Theor. Comp. Sys.*, 40(2):163–185, 2007.
3   Frank Drewes and Heiko Vogler. Learning deterministically recognizable tree series. *Journal of Automata, Languages and Combinatorics*, 12:333–354, 2007.
4   Andreas Maletti. Learning deterministically recognizable tree series – revisited. In Symeon Bozapalidis and George Rahonis, editors, *Proceedings of the 2nd international conference on Algebraic informatics*, CAI'07, pages 218–235, Berlin, Heidelberg, 2007. Springer-Verlag.

3.4 From Two-Way to One-Way Finite State Transducers

Emmanuel Filiot (Université Paris 12, FR)

Any two-way finite state automaton is equivalent to some one-way finite state automaton. This well-known result, shown by Rabin and Scott and independently by Shepherdson, states that two-way finite state automata (even non-deterministic) characterize the class of regular languages. It is also known that this result does not extend to finite string transductions: (deterministic) two-way finite state transducers strictly extend the expressive power of (functional) one-way transducers. In particular deterministic two-way transducers capture exactly the class of MSO-transductions of finite strings. In this talk, we address the following definability problem: given a function defined by a two-way finite state transducer, is it definable by a one-way finite state transducer? By extending Rabin and Scott's proof to transductions, we show that this problem is decidable. Our procedure builds a one-way transducer, which is equivalent to the two-way transducer, whenever one exists.

3.5 Forward and Backward Application of Symbolic Tree Transducers

Zoltán Fülöp (University of Szeged, HU)

We characterized symbolically recognizable (s-recognizable) tree languages in terms of classical recognizable tree languages and relabelings of infinite range. Also we gave sufficient conditions for that the syntactic composition of two symbolic tree transducers (stt) computes the composition of the tree transformations computed by each stt. We considered forward and backward application of stt and proved that the backward application of an stt to any s-recognizable tree language yields and s-recognizable tree language. We gave a linear stt of which the range is not an s-recognizable tree language. We showed that the forward application of a simple and linear stt preserves s-recognizability.

References
1 Fülöp, Zoltán and Vogler, Heiko. Forward and Backward Application of Symbolic Tree Transducers, http://arxiv.org/abs/1208.5324, 2013

3.6   On the Translations Produced by Multi Bottom-Up Tree Transducers

Daniel Gildea (University of Rochester, US)

Multi Bottom-Up Tree Transducers have recently been proposed as a model for machine translation due to the attractive property that they are closed under composition. Tree transducers are defined as relations between trees, but in syntax-based machine translation, we are ultimately concerned with the relations between the strings at the yields of the input and output trees. We examine the formal power of Multi Bottom-Up Tree Transducers from this point of view.

3.7   Higher-Order Tree Transducers and Their Expressive Power

Kazuhiro Inaba (Google Japan, JP)

This talk reviews and discusses about the expressive power of higher-order tree grammars and transducers. In the literature, two major notions of "high-order" devices have been studied. One that well-known to the transducer community is the line of researches on IO-/OI- hierarchy [1] and high-level transducers [2]. There, trees are the order-0 entities, and order-$(n+1)$ entities are the functions from order-n entities to an order-n entity. These classes of higher-order hierarchy are known to have beautiful properties. Notably, their expressive power is characterized by n-iterated pushdown stack (i.e., stack of stack of ... of stack), or by n-fold composition of first order macro tree transducers. We show, by fully utilizing the decomposition result, that the languages in OI-hierarchy are context-sensitive [3].

On the other hand, pushed by the recent need for higher-order model checking, broader class of higher-order functions are now investigated [4, 5]. It is called an "unsafe" grammar/transducer (and as a contrast, the former definition is called "safety" restriction), and takes all terms of simply-typed lambda- calculus into account. In particular, it involves a higher-order term containing lower- order free variables, which can never occur under the "safe" construction. Contrary to the safe case, no result is known about the first-order decomposition for unsafe case, nor whether it is included in the class of context-sensitive languages. We discuss the ongoing approach to tackle those open problems.

References
1   W. Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20:95–207, 1982.
2   J. Engelfriet and H. Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Informatica*, 26:131–192, 1988.

3 K. Inaba and S. Maneth. The complexity of tree transducer output languages. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 244–255, 2008.

4 T. Knapik, D. Niwiński, and P. Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *Typed Lambda Calculi and Applications (TLCA)*, pages 253–267, 2001.

5 C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Logic in Computer Science (LICS)*, pages 81–90, 2006.

3.8 Tree Automata with Constraints: A Brief Survey

Florent Jacquemard (IRCAM & INRIA – Paris, FR)

It is well-known that tree automata define exactly regular languages of trees. However for some problems one sometimes needs to test for equalities and disequalities of subtrees. For instance, ranges of non-linear tree transducers cannot be represented by tree automata. To overcome this problem some extensions of tree automata with tree (dis)equality constraints have been proposed. This talk surveys some of the most important models and their applications. Two families of automata are presented. First, we consider automata with local constraints. They have been used to solve problems related to pattern-matching, tree rewriting and more recently the tree homomorphism problem. Then, motivated by applications to XML processing and security protocols verification, we present a more recent model of tree automata with global constraints.

3.9 Tree Indexing by Deterministic Automata

Jan Janousek (Czech Technical University, CZ)

We present a basic survey and the main ideas of tree indexing implemented by deterministic automata. Given a tree of size n, we construct deterministic pushdown automata or deterministic finite tree automata which represent a full index of the tree for subtrees or tree patterns. Given an input tree pattern which matches the tree, its acceptance by the automaton is performed in $\mathcal{O}(m)$ time, where m is the size of the tree pattern, and does not depend on n. We present deterministic pushdown automata which read linear notations of trees and are analogous to deterministic finite automata representing full index of strings: a subtree pushdown automaton is analogous to a string factor automaton. A tree pattern pushdown automaton is then an extension of the subtree pushdown automaton for indexing the tree patterns. Moreover, also oracle versions of these automata can be constructed, as it is in the case for string indexing automata. All these pushdown automata are input-driven and with just one pushdown symbol in their basic versions. Therefore, they are convenient for implementation. Another possibility is to construct a deterministic finite tree automaton representing the index, or a deterministic pushdown automaton reading a tree in a linear

notations that corresponds to the finite tree automaton. These possibilities are also discussed
and shown.

References
1   Janousek, J., Melichar, B. On Regular Tree Languages and Deterministic Pushdown Auto-
        mata. In *Acta Informatica*, Vol. 46, No. 7, pp. 533-547, Springer, 2009.
2   Janousek, J. String Suffix Automata and Subtree Pushdown Automata. In: *Proceedings
        of the Prague Stringology Conference 2009*, pp. 160–172, Czech Technical University in
        Prague, Prague, 2009.
3   Melichar, B., Janousek, J., Flouri, T. Arbology: Trees and Pushdown Automata. In: *Ky-
        bernetika*, vol. 48, No.3, pp. 402-428, 2012.

3.10   Verifying Higher-Order Tree Transducers by Higher-Order Model Checking

Naoki Kobayashi (University of Tokyo, JP)

We discuss methods for type-checking (unsafe) higher-order tree transducers using forward
inference and higher-order model checking. The idea is to represent the forward image as a
higher-order recursion scheme, and use higher-order model checking to check the inclusion
by the output language. This approach is arguably more efficient in practice, thanks to the
recent advance of higher-order model checking algorithms. We present two variations of the
method for computing the forward image: one for the combination of regular input languages
and a higher-order multi-tree transducer, and the other for the combination of a higher-order
input language and a higher-order (single) tree transducer. The former is joint work with
Hiroshi Unno and Naoshi Tabuchi, presented at POPL 2010.

3.11   Natural Language Grammars: A Tree-Based Approach

Marco Kuhlmann (Uppsala University, SE)

In this talk I gave three examples of problems in theoretical computational linguistics whose
solution, in various ways, has involved the use of tree grammars and tree transducers: the
so-called lexicalization of tree adjoining grammars; the definition of a grammar formalism for
dependency grammar; and the question about the generative capacity of categorial grammars.
I also mentioned some open problems in computational linguistics that may be attacked
using tree automata theory.

References
1   Marco Kuhlmann and Giorgio Satta. Tree-Adjoining Grammars Are Not Closed Under
        Strong Lexicalization. *Computational Linguistics*, 38(3):617–629, 2012.

2 Marco Kuhlmann. Mildly Non-Projective Dependency Grammar. *Computational Linguistics*, 39(2):355–387, 2013.

3 Marco Kuhlmann, Alexander Koller, and Giorgio Satta. The Importance of Rule Restrictions in CCG. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 534–543, Uppsala, Sweden, 2010.

3.12 A Functional Language for Hyperstreaming XSLT

Pavel Labath (University of Bratislava, SK)

The problem of how to transform large data trees received on streams with a much smaller memory is still an open challenge despite of a decade of research on XML. Therefore, the current approach of the XSLT working group of the W3C is to provide streaming support only for a small fragment of XSLT 3.0. This has the drawback that many existing XSLT programs need to be rewritten in order to become executable on XML streams, while many others cannot be rewritten at all, since they are defining nonstreamble transformations.

We propose a new hyperstreaming approach that does not require any a priori restrictions. The model of hyperstreaming generalizes on the model of streaming by adding shredding operations for the output stream, so that its parts may be plugged together later on. Many transformations such as flips of document pairs are hyperstreamable but not streamable. We then present the functional language X-Fun for defining transformations between XML data trees, while providing shredding instructions. X-Fun can be understood as an extension of Frisch's XStream language with output shredding, while pattern matching is replaced by tree navigation with XPath expressions.

We also provide a compiler from a fragment of XSLT into X-Fun, which can then be considered as the core of XSLT. We then present a hyperstreaming algorithm for evaluating X-Fun programs which combines a recent XPath evaluator with a traditional functional programming engine. We have implemented a hyperstreaming evaluator for X-Fun and thus for XSLT and compared it experimentally with Saxon's XSLT implementation. It turns out that many XSLT programs become hyperstreamable with good efficiency and without any manual rewriting.

References
1 A. Frisch and K. Nakano. Streaming XML transformation using term rewriting. In *Programming Language Technologies for XML (PLAN-X)*, 2007b.

3.13 Determinacy and Rewriting of Top-Down and MSO Tree Transformations

Sebastian Maneth (University of Oxford, GB)

A query is determined by a view, if the result to the query can be reconstructed from the result of the view. We consider the problem of deciding for two given tree transformations, whether one is determined by the other. If the view transformation is induced by a tree transducer that may copy, then determinacy is undecidable, even for identity queries. For a large class of non-copying views, namely compositions of functional extended linear top-down tree transducers with regular look-ahead, we show that determinacy is decidable, where queries are given by deterministic top-down tree transducers with regular look-ahead or by MSO tree transducers. We also show that if a query is determined, then it can be rewritten into a query that works directly over the view and is in the same class as the given query. The proof uses

- uniformizers
- composition results and
- decidability of equivalence.

A uniformizer of a binary relation is a function that for each input of the relation chooses an arbitrary output of the relation. The idea is to construct a representation of the inverse of a view, then to build a uniformizer U for it, and then to build a transducer for the composition of the view, followed by U, followed by the query Q, and to check equivalence of this transducer with they query. The transducers are equivalent if and only if the query is determined by the view. These result will be presented at MFCS'2013 in Vienna.

3.14 Efficient Incremental Evaluation of Succinct Regular Expressions

Wim Martens (Universität Bayreuth, DE)

We present a method for efficient incremental evaluation of regular expressions with counters. Such expressions are used in grep, Python, Perl, Ruby, XML Schema, and are being considered for property paths in SPARQL 1.1. Furthermore, they can be exponentially more succinct than "traditional" regular expressions or non-deterministic finite automata as usually studied in the literature. Our evaluation method exploits the counter values in the expressions to avoid an exponential blow-up that the current state-of-the-art algorithms have. In this study, we present a thorough investigation of the use and structure of regular expressions with counters in some practical applications and we evaluate our algorithm on in synthetic and real benchmark tests based on the expressions we found in practice. Our benchmarks indicate that the new algorithm never performs worse than the state-of-the art but is up to a million times faster when the data is large.

3.15 Logical Definitions of Mildly Context-Sensitive Grammar Formalisms

Uwe Mönnich (Universität Tübingen, DE)

Joint work of Mönnich, Uwe
Main reference U. Mönnich, "A Logical Characterization of Extended TAGs," in Proc. of the 11th Int'l Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11), pp. 37–45, Paris, 2012.
URL http://alpage.inria.fr/tagplus11/lib/exe/fetch.php?media=tag_1105.pdf

The development of model-theoretic syntax in terms of a logical specification language that is both expressive and manageable presents problems for mono-level approaches. All these approaches suffer from a lack of expressive power in that the family of regular tree languages properly includes all other language families that are captured by the logical formalisms that have been considered in model-theoretic syntax. It is due to this lack of expressive power that grammatical phenomena like cross-serial dependencies in languages like Swiss German or Bambara are beyond the reach of the kind of logical apparatus currently applied to natural language syntax. The talk offers a solution to these problems by integrating a formally unified notion of grammar morphism into the framework of model-theoretic syntax. The approach we present follows Courcelle's extension of Rabin's method of model-theoretic interpretation. This extended version of the classical method of semantic interpretation serves to carve out the exact position of recent linguistic theories like minimalist syntax and (multicomponent) tree adjoining grammar within the family of mildly context-sensitive languages. In the process of determining this position we rely on the linguistically significant affinities between multiple regular tree grammars and simple context-free tree grammars on the one hand and minimalist syntax and tree adjoining grammars on the other. It turns out that the tree languages which are the output of finite-copying top-down tree transducers applied to regular tree languages are exactly the output tree languages of logical tree transducers which are direction preserving in the sense that edges in the output trees correspond to directed paths in the input trees. A similar result holds of (monadic) simple tree transducers which correspond to logical tree transducers which are either direction preserving or inverse direction preserving. Analyzing these results from the perspective of grammar theory leads to the overall conclusion that the formal counterparts of contemporary models of natural language syntax can be characterized in terms of grammar morphisms in the sense of this talk.

3.16 XML Stream Processing Based on Tree Transducer Composition

Keisuke Nakano (The University of Electro-Communications – Tokyo, JP)

Joint work of Nakano, Keisuke; Shin-Cheng Mu
Main reference K. Nakano, S.-C. Mu, "A Pushdown Machine for Recursive XML Processing," in Proc. of the 4th ASIAN Symp. on Programming Languages and Systems (APLAS'06), LNCS, Vol. 4279, pp. 340–356, Springer, 2006.
URL http://dx.doi.org/10.1007/11924661_21

I gave a talk about an application of tree transducer composition, that is, derivation of XML stream processors from XML tree manipulation programs. There are two styles of XML transformation programs: one is tree manipulation (like DOM programming) and another

is stream processing (like SAX programming). Tree manipulation style is much easier to write a program than stream processing style. However, a tree manipulation program is less efficient in both time and space than stream processing because it cannot start to output the result before building the whole tree structure in memory. In this talk, I presented a solution to obtain both advantages of two programming styles by deriving XML stream processors from tree manipulation programs. The derivation is based on the theory of tree transducer composition. I developed new composition laws because known composition laws do not work for this particular problem. This work has been presented in APLAS 2004, APLAS 2006, and a technical part of these results is based on my result published in Journal of Theory of Computing Systems in 2009.

3.17 Panic Automata. Yet Another Automata in Quest of Decidability of Mathematical Theories

Damian Niwinski (University of Warsaw, PL)

What features make the theory of a tree – say, the theory in monadic second-order logic (MSO) – decidable? The Rabin 1969 breakthrough result, establishing decidability of the MSO theory of the full binary tree, gave rise to analogous results for various shapes of trees. One way to classify (in general, labeled) trees is to view them as terms over a first-order signature, generated by recursive schemes (sometimes called tree grammars), using typed non-terminals of any order. In this setting, Rabin established the result for level 0, i.e., regular trees, and Courcelle extended it to level 1, i.e., algebraic trees. Knapik, Niwinski, and Urzyczyn [4] extended it further to trees generated by grammars of any level n, but imposing a technical restriction on the use of parameters, called safety. They a posteriori justified the relevant hierarchy of trees, by characterizing it in terms of higher-order pushdown automata. Recall that these automata, introduced by Maslov in 1974, use stacks of stacks of stacks...; the higher-level push operation duplicates the topmost stack of the appropriate level, and the pop operation pops such a stack. An elegant machine/grammar independent characterization of the hierarchy was later given Caucal [2]. In 2005, the KNU together with I.Walukiewicz [5] and independently Aehlig, de Miranda and Ong [1], showed decidability of the MSO theory of trees generated by grammars of level 2 without any restriction. The proof of the former group was based on an enhancement of the second-order pushdown automaton by a cascade popping operation (invented by Pawel Urzyczyn), called panic. Roughly speaking, this operation, guided by the topmost symbol on the stack, reconstructs the stack on which this symbol was put originally, disregarding its future duplications. Further, Luke Ong [6] established (2006) decidability of the MSO theories of trees generated by unrestricted grammars of any level, and extended (in 2008, together with Hague, Murawski, and Serre [3]) the automata-theoretic characterization, introducing collapsible pushdown automata, which generalize panic automata to all levels. It was only in 2011–2012, when Pawel Parys [7, 8] eventually separated the two concepts, by showing that the panic/collapse operation is indeed needed to capture the expressive power of higher-order grammars, or in other words, that the safety is indeed a restriction.

References

1 Aehlig, K., de Miranda J.G., and Ong, L., The monadic second order theory of trees given by arbitrary level-two recursion schemes is decidable. In: Proc. TLCA '05, Springer LNCS 3461 (2005), 39-54.

2 Caucal, D., On infinite terms having a decidable monadic second-order theory. MFCS 2002, 65–176.

3 Hague, M., Murawski, A.S., Ong, C.-H. L., Serre O., Collapsible Pushdown Automata and Recursion Schemes. LICS 2008, 452-461.

4 Knapik, T., Niwiński, D., Urzyczyn, P., Higher-order pushdown trees are easy. FoSSaCS 2002, 205–222.

5 Knapik, T., Niwiński, D., Urzyczyn, P., Walukiewicz, I., Unsafe Grammars and Panic Automata. ICALP 2005, 1450-1461.

6 Ong, C.-H. L., On Model-Checking Trees Generated by Higher-Order Recursion Schemes. LICS 2006, 81-90.

7 Parys, P., Collapse Operation Increases Expressive Power of Deterministic Higher Order Pushdown Automata. STACS 2011, 603-614.

8 Parys, P., On the Significance of the Collapse Operation. LICS 2012, 521-530.

3.18   Recursion Schemes and Pattern Matching

Chih-Hao Luke Ong (University of Oxford, GB)

Higher-order model checking is the model checking of infinite trees generated by higher-order recursion schemes (HORS) and related models of computation. We introduced the problem of deciding monadic second-order theories of trees generated by HORS, and discussed recent decidability proofs by Ong and others. Motivated by the standard functional programming idiom of definition by pattern matching, we introduced pattern matching recursion schemes (PMRS) and their safety verification problem. We sketched a semi-complete method that first builds an abstraction of the (undecidable) verification problem in the form of weak PMRS; a solution is then obtained by successive refinement using a CEGAR loop. This is based on a POPL 2011 paper by Luke Ong and Steven Ramsay.

3.19   Descriptional Complexity of Input-Driven Pushdown Automata

Kai T. Salomaa (Queen's University – Kingston, CA)

Every nondeterministic input-driven pushdown automaton (IDPDA) has an equivalent deterministic IDPDA and the IDPDA languages retain many of the desirable closure properties of regular languages. The IDPDA model is known in the literature also as a visibly pushdown automaton and is mathematically equivalent to the nested word automaton.

It is known that a deterministic automaton equivalent to a nondeterministic IDPDA of size n may need size $2^{\Omega(n^2)}$ (Alur, Madhusudan, J.ACM 2009). This talk surveys recent work on the descriptional complexity of converting a nondeterministic IDPDA to an unambiguous one and of determinizing an unambiguous IDPDA. Also we survey the descriptional complexity of the Boolean operations, concatenation and Kleene star on IDPDA languages.

References
1 Alexander Okhotin, Xiaoxue Piao, Kai Salomaa: Descriptional Complexity of Input-Driven Pushdown Automata. In: Languages Alive (H. Bordihn, M. Kutrib, B. Truthe (Eds.)), Springer Lecture Notes in Computer Science 2012, pp. 186-206 http://dx.doi.org/10.1007/978-3-642-31644-9_13

3.20   Visibly Pushdown Transducers

Frederic Servais (Hasselt University – Diepenbeek, BE)

The present work proposes visibly pushdown transducers (VPTs) for defining transformations of documents with a nesting structure. We show that this subclass of pushdown transducers enjoy good properties. Notably, we show that functionality is decidable in PTime and k-valuedness in co-NPTime. While this class is not closed under composition and its type checking problem against visibly pushdown automata is undecidable, we identify a subclass, the well-nested VPTs, closed under composition and with a decidable type checking problem. Furthermore, we show that the class of VPTs is closed under look-ahead, and that the deterministic VPTs with look-ahead characterize the functional VPTs transductions. Finally, we investigate the resources necessary to perform transformations defined by VPTs. We devise a memory efficient algorithm. Then we show that it is decidable whether a VPT transduction can be performed with a memory that depends on the level of nesting of the input document but not on its length.

3.21 FAST: Functional Abstraction of Symbolic Transductions

Margus Veanes (Microsoft – Redmond, US)

We introduce a tree manipulation language and tool, called FAST, which supports trees over infinite alphabets. The core of FAST is based on a combination of state-of-the-art satisfiability modulo theories solving techniques and tree automata and tree transducer algorithms, enabling it to model programs whose input and output can range over any decidable theory. Overall, we strike a balance between expressiveness and precise analysis that works for a large class of tree-manipulating programs.

3.22 Streaming Tree Transducers

Loris d'Antoni (University of Pennsylvania, US)

We introduce Streaming Tree Transducers as an analyzable, executable, and expressive model for transforming strings, unranked and ranked ordered trees, and forests. Given a linear encoding of the input tree, the transducer makes a single left-to-right pass through the input, and computes the output using a finite-state control, a visibly pushdown stack, and a finite number of variables that can store output chunks that can be combined using the operations of string-concatenation and tree-insertion. We prove that the expressiveness of the model coincides with transductions definable using monadic second-order logic (MSO).We establish complexity upper bounds of ExpTime for type-checking and NExpTime for checking functional equivalence for our model. We consider variations of the basic model when inputs/outputs are restricted tostrings and ranked trees, and in particular, present the model of bottom–up ranked-tree transducers, which is the first known MSO-equivalent transducer model that processes trees in a bottom–up manner.

References
1 Rajeev Alur and Loris D'Antoni. *Streaming Tree Transducers.* ICALP (2), 2012, 42-53.

Participants

- Henrik Björklund
  University of Umeå, SE
- Johanna Björklund
  University of Umeå, SE
- Adrien Boiret
  ENS – Paris, FR
- Bruno Courcelle
  University of Bordeaux, FR
- Loris d'Antoni
  University of Pennsylvania, US
- Frank Drewes
  University of Umeå, SE
- Emmanuel Filiot
  Université Paris 12, FR
- Zoltan Fülöp
  University of Szeged, HU
- Olivier Gauwin
  University of Bordeaux, FR
- Daniel Gildea
  University of Rochester, US
- Kazuhiro Inaba
  Google Japan, JP
- Florent Jacquemard
  IRCAM & INRIA – Paris, FR

- Jan Janousek
  Czech Technical University, CZ
- Naoki Kobayashi
  University of Tokyo, JP
- Marco Kuhlmann
  Uppsala University, SE
- Pavel Labath
  University of Bratislava, SK
- Aurélien Lemay
  University of Lille III, FR
- Sebastian Maneth
  NICTA & University of New
  South Wales, Sydney, AU
- Wim Martens
  Universität Bayreuth, DE
- Uwe Mönnich
  Universität Tübingen, DE
- Keisuke Nakano
  The University of
  Electro-Communications –
  Tokyo, JP
- Joachim Niehren
  INRIA Nord Europe – Lille, FR

- Damian Niwinski
  University of Warsaw, PL
- Chih-Hao Luke Ong
  University of Oxford, GB
- Pierre-Alain Reynier
  Université de Provence, FR
- Kai T. Salomaa
  Queen's Univ. – Kingston, CA
- Helmut Seidl
  TU München, DE
- Frédéric Servais
  Hasselt Univ. – Diepenbeek, BE
- Jean-Marc Talbot
  Aix-Marseille University, FR
- Sophie Tison
  Université de Lille I, FR
- Jan Van den Bussche
  Hasselt University, BE
- Margus Veanes
  Microsoft – Redmond, US
- Heiko Vogler
  TU Dresden, DE