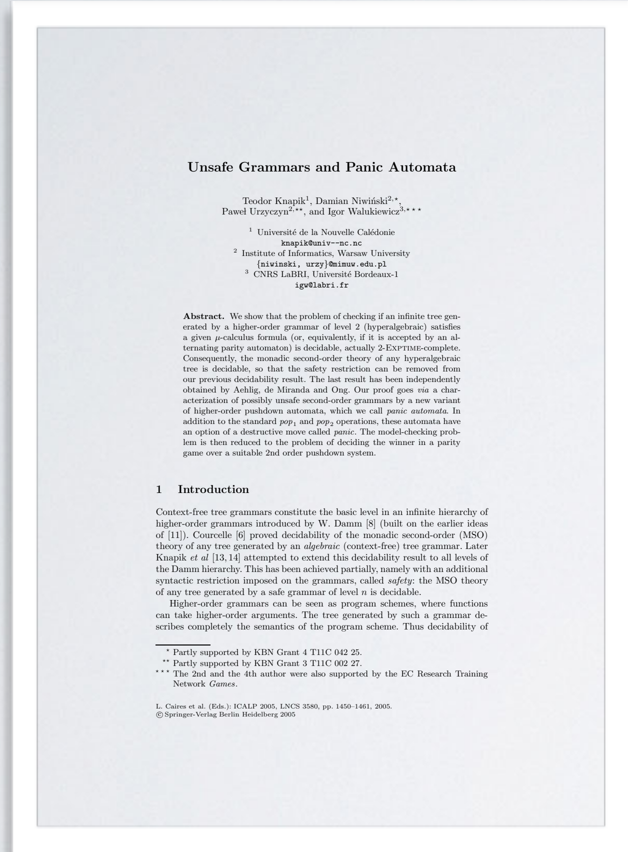
LINEARITY IN HIGHER-ORDER RECURSION SCHEMES

Pierre Clairambault
CNRS & ENS Lyon

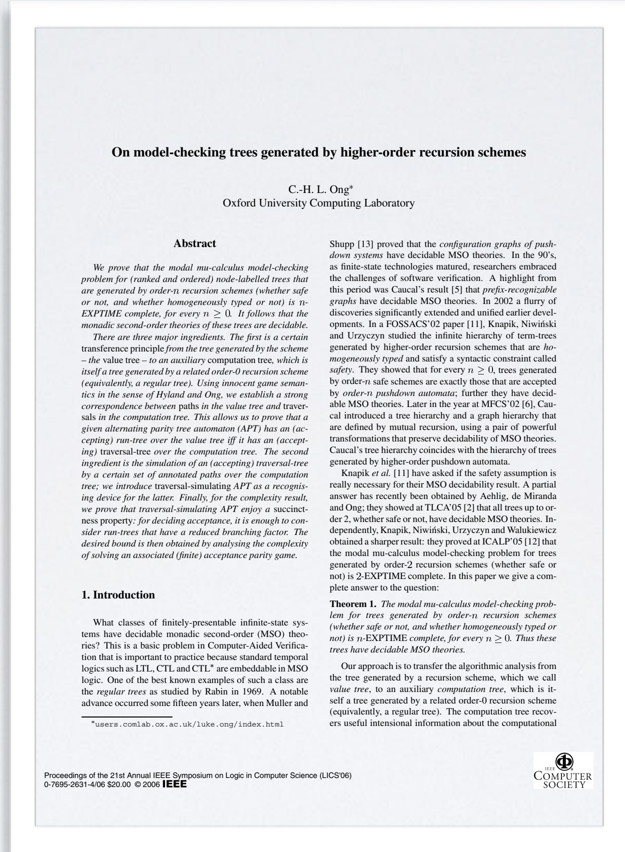
Charles Grellois
Aix Marseille Université

Andrzej Murawski
University of Oxford

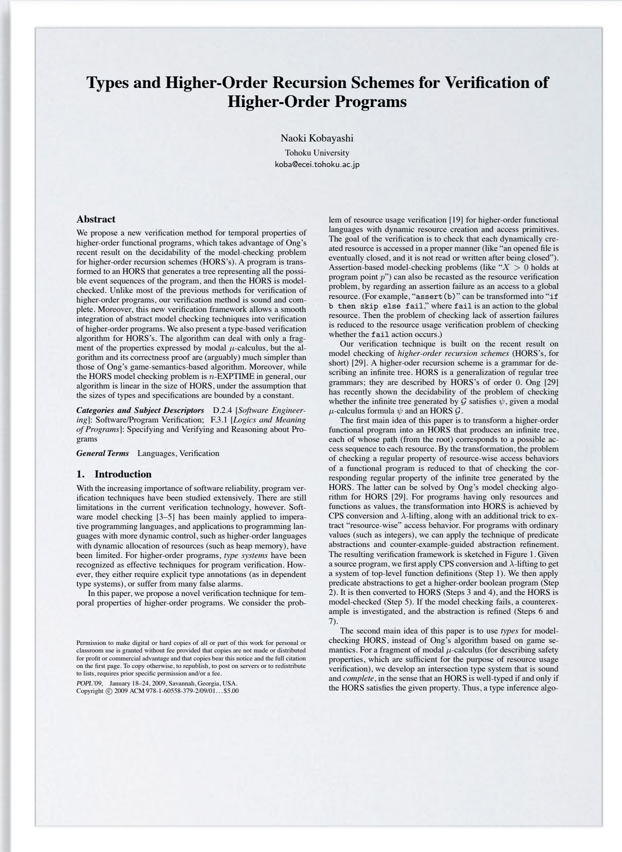
IFIP WG 2.2



Walukiewicz et al
(ICALP 2005)



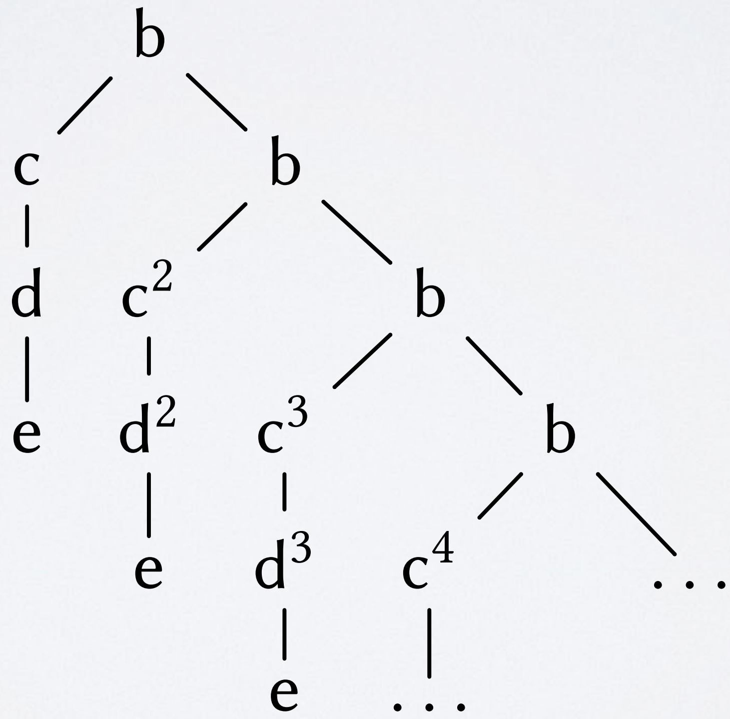
Ong
(LICS 2006)



Kobayashi
(POPL 2009)

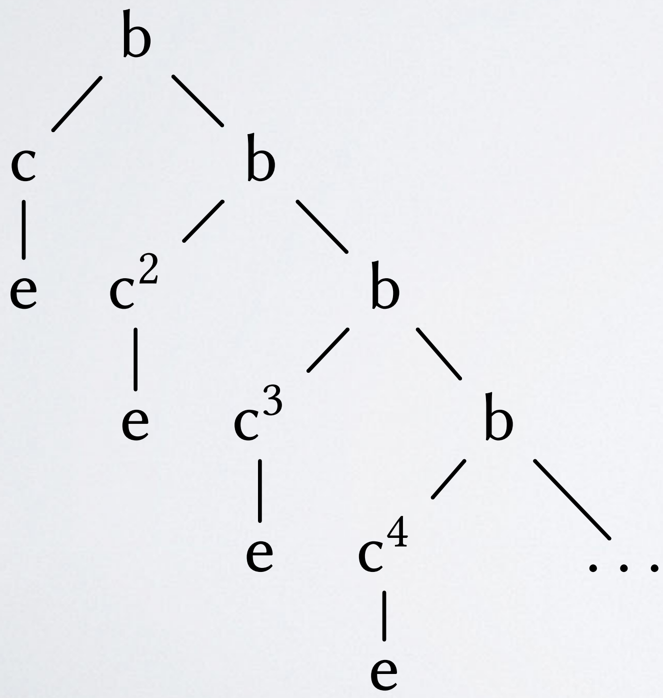
TREES OVER FIRST-ORDER ALPHABETS

$$\Sigma = \{b :: o \to o \to o, c :: o \to o, d :: o \to o, e :: o\}$$



RECURSIVE APPLICATIVE PROGRAM SCHEMES (NIVAT, 1970S)

$$F_i(x_1, \cdots, x_{n_i}) = e_i$$



$$\begin{array}{rcl} S & = & F(ce) \\ F(x) & = & b(x, F(cx)) \end{array}$$

HIGHER-ORDER TYPES

$$A ::= o \mid A \to A$$

$$\begin{aligned} ord(o) &= 0 \\ ord(A_1 \to A_2) &= \max(ord(A_1)+1, ord(A_2)) \end{aligned}$$

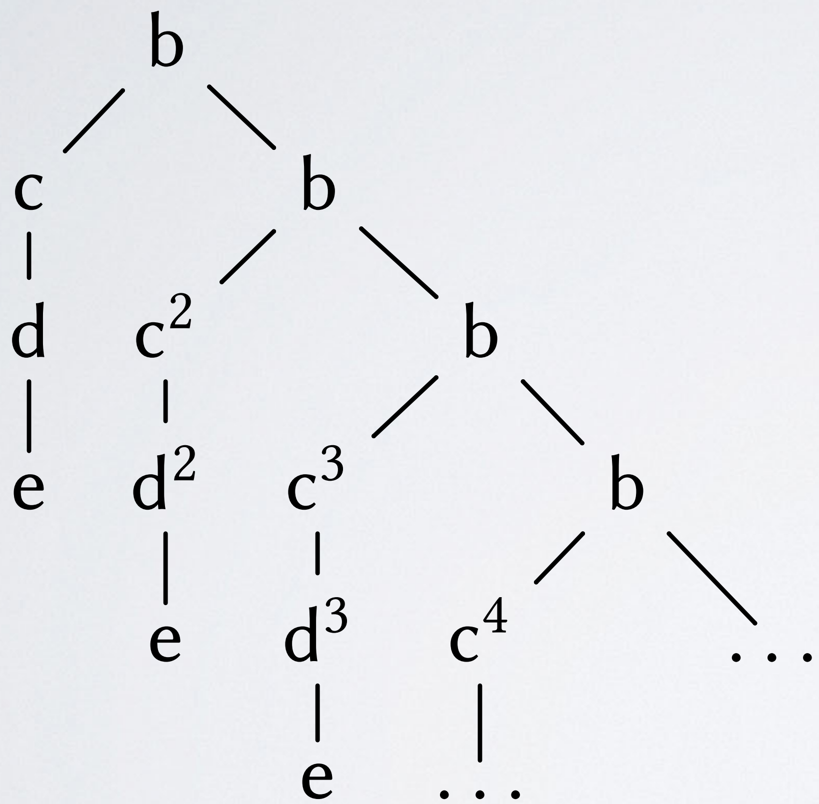
HIGHER-ORDER RECURSION SCHEMES

order-2 HORS $\mathcal{G} = \langle \Sigma, \mathcal{N}, \mathcal{R}, S \rangle$

$$\Sigma = \{\mathrm{b} :: o \to o \to o, \mathrm{c} :: o \to o, \mathrm{d} :: o \to o, \mathrm{e} :: o\}$$
$$\mathcal{N} = \{S :: o, F :: (o \to o) \to o, G :: o \to o, H :: (o \to o) \to o \to o\}$$

$$\begin{aligned} S &= F\,G \\ F f &= b\,(f\,\mathrm{e})\,(F\,(H\,f)) \\ G x &= \mathrm{c}\,(\mathrm{d}\,x) \\ H f x &= \mathrm{c}\,(f\,(\mathrm{d}\,x)) \end{aligned}$$

EXAMPLE



$$
\begin{aligned}
S &= F\,G \\
F\,f &= b\,(f\ \mathrm{e})\,(F\,(H\,f)) \\
G x &= \mathrm{c}\,(\mathrm{d}\,x) \\
H\,f\,x &= \mathrm{c}\,(f\,(\mathrm{d}\,x))
\end{aligned}
$$

MSO (LICS 2006)

On model-checking trees generated by higher-order recursion schemes

C.-H. L. Ong*
Oxford University Computing Laboratory

Theorem 1. *The modal mu-calculus model-checking problem for trees generated by order-n recursion schemes (whether safe or not, and whether homogeneously typed or not) is n-*EXPTIME *complete, for every $n \geq 0$. Thus these trees have decidable MSO theories.*

MSO (FOSSSACS 2002)

Higher-Order Pushdown Trees Are Easy

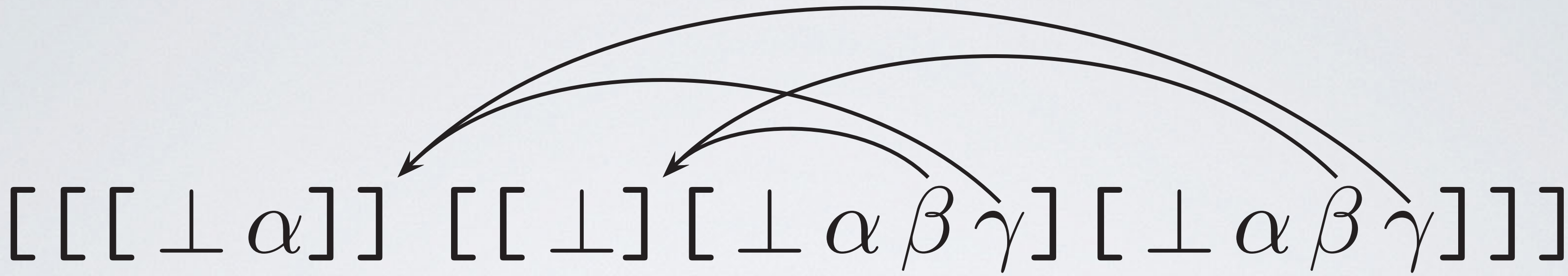
Teodor Knapik[1], Damian Niwiński[2⋆], and Paweł Urzyczyn[2⋆⋆]

[1] Université de la Réunion, BP 7151,
97715 Saint Denis Messageries Cedex 9, Réunion
knapik@univ--reunion.fr

[2] Institute of Informatics, Warsaw University
ul. Banacha 2, 02–097 Warszawa, Poland
{niwinski,urzy}@mimuw.edu.pl

Abstract. We show that the monadic second-order theory of an infinite tree recognized by a higher-order pushdown automaton of any level is decidable. We also show that trees recognized by pushdown automata of level n coincide with trees generated by safe higher-order grammars of level n. Our decidability result extends the result of Courcelle on algebraic (pushdown of level 1) trees and our own result on trees of level 2.

AUTOMATA



Unsafe Grammars and Panic Automata

Teodor Knapik[1], Damian Niwiński[2,⋆],
Paweł Urzyczyn[2,⋆⋆], and Igor Walukiewicz[3,⋆⋆⋆]

ICALP'05

Collapsible Pushdown Automata and Recursion Schemes

MATTHEW HAGUE, Royal Holloway, University of London
ANDRZEJ S. MURAWSKI, DIMAP and Department of Computer Science, University of Warwick
C.-H. LUKE ONG, Department of Computer Science, University of Oxford
OLIVIER SERRE, IRIF, CNRS & Université Paris Diderot – Paris 7

LICS'08

HORS (POPL 2009)

Types and Higher-Order Recursion Schemes for Verification of Higher-Order Programs

Naoki Kobayashi
Tohoku University
koba@ecei.tohoku.ac.jp

Abstract

We propose a new verification method for temporal properties of higher-order functional programs, which takes advantage of Ong's recent result on the decidability of the model-checking problem for higher-order recursion schemes (HORS's). A program is transformed to an HORS that generates a tree representing all the possible event sequences of the program, and then the HORS is model-checked. Unlike most of the previous methods for verification of higher-order programs, our verification method is sound and complete. Moreover, this new verification framework allows a smooth integration of abstract model checking techniques into verification of higher-order programs. We also present a type-based verification

lem of resource usage verification [19] for higher-order functional languages with dynamic resource creation and access primitives. The goal of the verification is to check that each dynamically created resource is accessed in a proper manner (like "an opened file is eventually closed, and it is not read or written after being closed"). Assertion-based model-checking problems (like "$X > 0$ holds at program point p") can also be recasted as the resource verification problem, by regarding an assertion failure as an access to a global resource. (For example, "`assert(b)`" can be transformed into "`if b then skip else fail`," where `fail` is an action to the global resource. Then the problem of checking lack of assertion failures is reduced to the resource usage verification problem of checking whether the `fail` action occurs.)

LINEARITY (POPL'18)

Linearity in Higher-Order Recursion Schemes

PIERRE CLAIRAMBAULT, Univ Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, Lyon, France
CHARLES GRELLOIS, INRIA, Sophia Antipolis, France and Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, Marseille, France
ANDRZEJ S. MURAWSKI, Department of Computer Science, University of Oxford, United Kingdom

Higher-order recursion schemes (HORS) have recently emerged as a promising foundation for higher-order program verification. We examine the impact of enriching HORS with linear types. To that end, we introduce two frameworks that blend non-linear and linear types: a variant of the λY-calculus and an extension of HORS, called linear HORS (LHORS).

First we prove that the two formalisms are equivalent and there exist polynomial-time translations between them. Then, in order to support model-checking of (trees generated by) LHORS, we propose a refined version of alternating parity tree automata, called LNAPTA, whose behaviour depends on information about linearity. We show that the complexity of LNAPTA model-checking for LHORS depends on two type-theoretic parameters: linear order and linear depth. The former is in general smaller than the standard notion of order and ignores linear function spaces. In contrast, the latter measures the depth of linear clusters inside a type. Our main result states that LNAPTA model-checking of LHORS of linear order n is n-EXPTIME-complete, when linear depth is fixed. This generalizes and improves upon the classic result of Ong, which relies on the standard

- **Data (booleans)**

$$o \longrightarrow \cdots \longrightarrow o \longrightarrow o$$

- **CBV**

$$(A_1 \rightarrow_v A_2)^* = A_1^* \rightarrow (A_2^* \rightarrow o) \rightarrow o$$

THREE EXAMPLES


Higher-Order Multi-Parameter Tree Transducers and Recursion Schemes for Program Verification

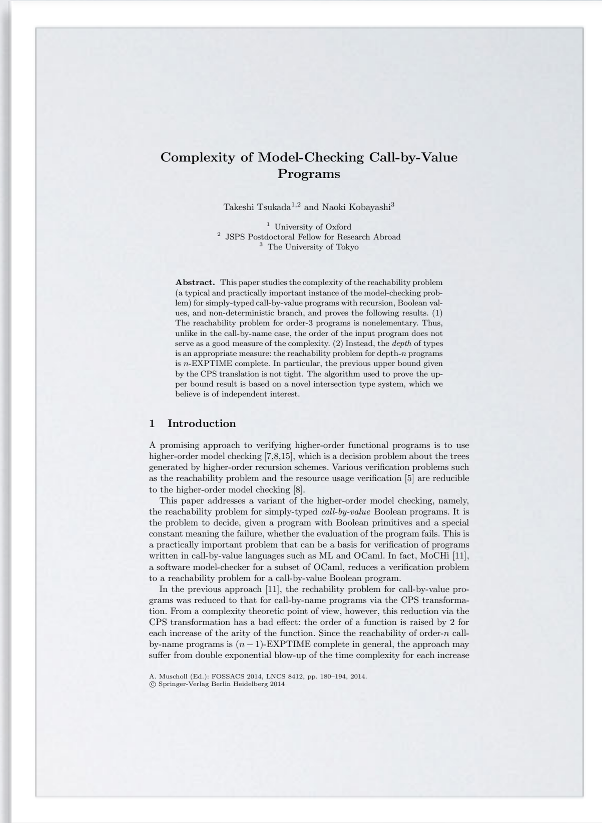
Naoki Kobayashi, Tohoku University, koba@ecei.tohoku.ac.jp
Naoshi Tabuchi, Tohoku University, tabee@kb.ecei.tohoku.ac.jp
Hiroshi Unno, Tohoku University, uhiro@kb.ecei.tohoku.ac.jp

RSFD
(POPL 2010)


A Traversal-based Algorithm for Higher-Order Model Checking

Robin P. Neatherway, University of Oxford, robin.neatherway@cs.ox.ac.uk
C.-H. Luke Ong, University of Oxford, luke.ong@cs.ox.ac.uk
Steven J. Ramsay, University of Oxford, steven.ramsay@cs.ox.ac.uk

HORSC
(POPL 2012)


Complexity of Model-Checking Call-by-Value Programs

Takeshi Tsukada[1,2] and Naoki Kobayashi[3]

[1] University of Oxford
[2] JSPS Postdoctoral Fellow for Research Abroad
[3] The University of Tokyo

CBV
(FOSSACS 2014)

LINEAR HORS

$$
\begin{array}{rcl}
\varphi, \psi, \ldots & ::= & o \mid \varpi \multimap \psi \mid \varphi \rightarrow \psi \\
\varpi, \kappa, \iota, \ldots & ::= & \varphi \mid \&_{i \in I} \varphi_i
\end{array}
$$

TYPING RULES

$$\frac{\Gamma \mid \Delta_1 \vdash_{\mathrm{ap}} t :: \varpi \multimap \varphi \qquad \Gamma \mid \Delta_2 \vdash_{\mathrm{ap}} u :: \varpi}{\Gamma \mid \Delta_1, \Delta_2 \vdash_{\mathrm{ap}} t\, u :: \varphi}$$

$$\frac{\Gamma \mid \Delta \vdash_{\mathrm{ap}} t :: \varphi_1 \to \varphi_2 \qquad \Gamma \mid \_ \vdash_{\mathrm{ap}} u :: \varphi_1}{\Gamma \mid \Delta \vdash_{\mathrm{ap}} t\, u :: \varphi_2}$$

$$\frac{\Gamma \mid \Delta \vdash_{\mathrm{ap}} t_i :: \varphi_i \qquad (i \in I)}{\Gamma \mid \Delta \vdash_{\mathrm{ap}} \langle t_i \mid i \in I \rangle :: \&_{i \in I} \varphi_i}$$

$\lambda\ell Y$

$$\frac{}{\Gamma, x :: \kappa \mid \Delta \vdash x :: \kappa}$$

$$\frac{j \in I}{\Gamma \mid \Delta, x :: \&_{i \in I} \varphi_i \vdash \pi_j\, x :: \varphi_j}$$

$$\frac{\Gamma \mid \Delta_1 \vdash t :: \kappa \multimap \varphi \qquad \Gamma \mid \Delta_2 \vdash u :: \kappa}{\Gamma \mid \Delta_1, \Delta_2 \vdash t\, u :: \varphi}$$

$$\frac{\Gamma \mid \Delta \vdash t :: \varphi \to \psi \qquad \Gamma \mid \_ \vdash u :: \varphi}{\Gamma \mid \Delta \vdash t\, u :: \psi}$$

$$\frac{\Gamma, x :: \varphi \mid \Delta \vdash t :: \psi}{\Gamma \mid \Delta \vdash \lambda x^{\varphi}.\, t :: \varphi \to \psi}$$

$$\frac{\Gamma \mid \Delta, x :: \kappa \vdash t :: \varphi}{\Gamma \mid \Delta \vdash \ell x^{\kappa}.\, t :: \kappa \multimap \varphi}$$

$$\frac{\Gamma \mid \Delta \vdash t_i :: \varphi_i \qquad (1 \leq i \leq n)}{\Gamma \mid \Delta \vdash \langle t_i \mid 1 \leq i \leq n \rangle :: \&_{1 \leq i \leq n} \varphi_i}$$

$$\frac{\Gamma \mid \Delta \vdash t :: \&_{1 \leq i \leq n} \varphi_i}{\Gamma \mid \Delta \vdash \pi_i\, t :: \varphi_i}$$

$$\frac{\Gamma, x : \kappa \mid \_ \vdash t :: \kappa}{\Gamma \mid \_ \vdash \mathbb{Y} x^{\kappa}.\, t :: \kappa}$$

LINEAR ORDER

$$
\begin{array}{rcl}
\varphi, \psi, \ldots & ::= & o \mid \varpi \multimap \psi \mid \varphi \to \psi \\
\varpi, \kappa, \iota, \ldots & ::= & \varphi \mid \&_{i \in I} \varphi_i
\end{array}
$$

$$
\begin{array}{rcl}
\ell o(o) & = & 0 \\
\ell o(\varpi \multimap \varphi) & = & \max(\ell o(\varpi), \ell o(\varphi)) \\
\ell o(\varphi \to \psi) & = & \max(\ell o(\varphi) + 1, \ell o(\psi)) \\
\ell o(\&_{i \in I} \varphi_i) & = & \max_{i \in I} \ell o(\varphi_i)
\end{array}
$$

LINEAR ORDER-0 LHORS

$$\mathcal{N} = \{ \quad S :: o, \quad F :: (o \multimap o) \multimap o, \quad G :: o \multimap o, \quad H :: (o \multimap o) \multimap o \multimap o \quad \}$$

$$\begin{aligned} S &= F\,G \\ F f &= b\,\langle f\,\mathrm{e}, F\,(H\,f)\rangle \\ G x &= \mathrm{c}\,(\mathrm{d}\,x) \\ H f x &= \mathrm{c}\,(f\,(\mathrm{d}\,x)) \end{aligned}$$

MAIN RESULT (POPL'18)

THEOREM 2. *Assume $n \geq 1$. The time complexity of checking whether a LNAPTA $\mathcal{A} = \langle \Sigma, Q, \delta, q_0 \rangle$ accepts the value tree of a LHORS $\mathcal{G}$ of linear order n (and bounded linear depth) is* $\exp_n(O(poly(|Q||\mathcal{G}|)))$.

APPLICATIONS

- RSFD

$$o \multimap \ldots \multimap o \multimap o$$

- HORSC

$$o \,\&\, \cdots \,\&\, o \multimap o$$

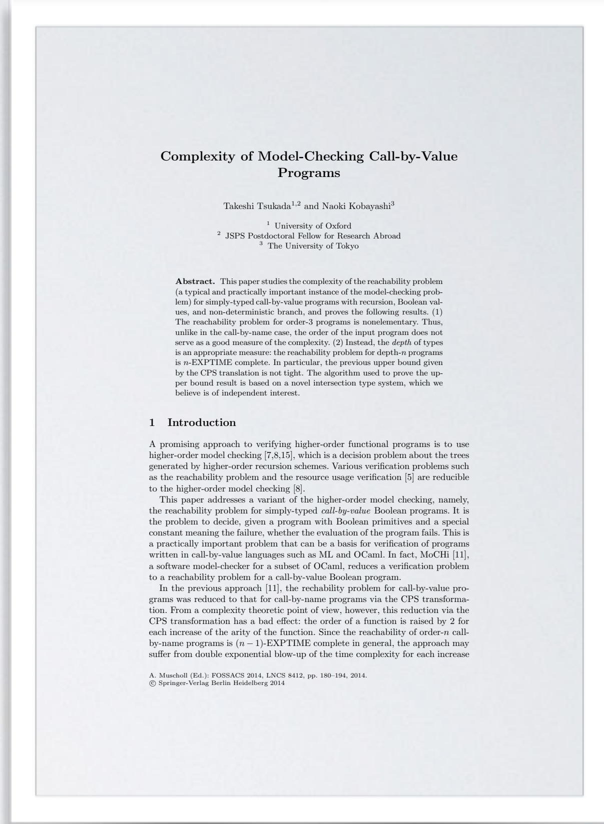
- CBV

$$(A_1 \to_v A_2)^* = A_1^* \to (A_2^* \to o) \multimap o$$

THREE PAPERS

RSFD
(POPL 2010)

HORSC
(POPL 2012)



CBV
(FOSSACS 2014)

LINEAR DEPTH

Definition 2. The **local linear depth** $\mathrm{lld}(\kappa)$ of a kind κ is defined inductively as follows

$$\begin{aligned} \mathrm{lld}(o) &= 0 & \mathrm{lld}(\varpi \multimap \varphi) &= \max(\mathrm{lld}(\varpi), \mathrm{lld}(\varphi)) + 1 \\ \mathrm{lld}(\&_{i \in I} \varphi_i) &= \max_{i \in I} \mathrm{lld}(\varphi_i) + 1 & \mathrm{lld}(\varphi \to \psi) &= \mathrm{lld}(\psi) \end{aligned}$$

The **linear depth** of κ, written $\ell\mathrm{d}(\kappa)$, is the maximum of $\mathrm{lld}(\iota)$, taken over all subkinds ι of κ.

Proposition 33. *Given a LHORS $\mathcal{G}$ with N non-terminals, kinds of maximal size S, linear depth D and linear order n; and a linear-nonlinear APT $\mathcal{A}$ with colours bounded by p and states bounded by $Q \geq p$. For $n \geq 1$, the time complexity for solving Typ$(\mathcal{G}, \mathcal{A})$ is* $O(N^{\lceil p/2 \rceil + 2} \exp_n(O(2^D)(QS)^{O(2^D)}))$.

LICS'09

A Type System Equivalent to the Modal Mu-Calculus Model Checking of Higher-Order Recursion Schemes

N. Kobayashi
Tokohu University

C.-H. L. Ong
University of Oxford

Abstract

The model checking of higher-order recursion schemes has important applications in the verification of higher-order programs. Ong has previously shown that the modal mu-calculus model checking of trees generated by order-n recursion scheme is n-EXPTIME complete, but his algorithm and its correctness proof were rather complex. We give an alternative, type-based verification method: Given a modal mu-calculus formula, we can construct a type system in which a recursion scheme is typable if, and only if, the (possibly infinite, ranked) tree generated by the scheme satisfies the formula. The model checking problem is thus reduced to a type checking problem. Our type-based approach yields a simple verification algorithm, and its correctness proof (constructed without recourse to game semantics) is comparatively easy to understand. Furthermore, the algorithm is polynomial-time in the size of the recursion scheme, assuming that the sizes of types and the formula are bounded above by a constant.

1 Introduction

The model checking of infinite structures generated by higher-order recursion schemes has drawn growing attention from both theoretical and practical communities. From

checking problem for trees generated by arbitrary order-n recursion schemes is n-EXPTIME complete (and hence these trees have decidable MSO theories); further [5] these schemes are equi-expressive with a new class of automata, called *collapsible* pushdown automata. On the practical side, Kobayashi [11] has recently shown that the verification of higher-order programs can be reduced to that of higher-order recursion schemes. He constructed a transformation of a higher-order program into a recursion scheme that generates a (possibly infinite) tree representing all the possible event sequences of the program; thus, temporal properties of the program can be verified by model-checking the recursion scheme.

Ong's algorithm for verifying higher-order recursion schemes is rather complex and probably hard to understand: The algorithm reduces the model-checking problem to a parity game over *variable profiles*, and its correctness proof relies on game semantics [7]. Hague et al. [5] gave an alternative proof via a reduction of the model checking of recursion schemes to that of collapsible pushdown automata; their reduction is also based on game semantics. Kobayashi [11] showed that given a Büchi tree automaton with a trivial acceptance condition (a class which Aehlig [1] has called *trivial automata*), one can construct an intersection type system in which a recursion scheme is typable if, and only if, the tree generated by the scheme is accepted by the automaton. (Prior to Kobayashi's work [11], Aehlig [1]

REFINEMENT

- Intersection types

$$\frac{q \in Q}{q :: o} \qquad \frac{\sigma :: \varpi \qquad \tau :: \varphi}{\sigma \multimap \tau :: \varpi \multimap \varphi} \qquad \frac{(\sigma_i :: \varphi)_{i \in I} \qquad \sigma :: \psi}{(\bigwedge_{i \in I} \Box_{c_i} \sigma_i) \to \sigma :: \varphi \to \psi}$$

- Parity game

$$Typ(\mathcal{G}, \mathcal{A}) \qquad \exp_n(O(poly(|Q||\mathcal{G}|)))$$

- Number of refinements

$$\sharp(\varphi \to \psi) \leq 2^{C\sharp\varphi}\sharp\psi \qquad \sharp(\varpi \multimap \varphi) = (\sharp\varpi + 1)(\sharp\varphi)$$

LNAPTA

Definition 4. **Tree kinds** are the kinds θ generated by

$$\theta \quad ::= \quad o \mid \nu \multimap \theta \mid o \rightarrow \theta \qquad \text{and} \qquad \nu \quad ::= \quad \&_{1 \leq i \leq n}\, o$$

A **tree signature** is a finite list $\Sigma = \mathrm{b}_1 :: \theta_1, \ldots, \mathrm{b}_n :: \theta_n$ where θ_i is a tree kind for all $1 \leq i \leq n$.

APTA = **LNAPTA** for standard tree kinds

CONCLUSIONS

- LHORS: more expressivity for the same asymptotic model-checking complexity

- Unification and extension of existing results, where reduction to HORS would give inaccurate bounds

- A tool to understand and tame complexity of higher-order model checking