

Infinite Trees, Recursion Schemes and Game Semantics

Luke Ong

Oxford University Computing Laboratory

Related papers available at

`www.comlab.ox.ac.uk/oucl/work/luke.ong/`

A Basic Problem in Verification: Find classes of finitely-presentable infinite structures with decidable monadic second-order (MSO) theories.

We study the infinite hierarchy of (possibly infinite) term-trees generated by **higher-order recursion schemes** (= simply-typed lambda calculus + 1st-order function symbols + fixpoints). **Why?**

- Rich and unifying class - subsumes (and extends) major results.
- Robust framework - admits different characterizations.

Main Results. For each $n \geq 0$:

- (i) The modal mu-calculus model checking problem for *RecSchTree* _{n} (trees generated by order- n recursion schemes) is n -EXPTIME complete. Hence trees in *RecSchTree* _{n} have decidable MSO theories.
- (ii) *RecSchTree* _{n} = *PanicAutoTree* _{n} (i.e. trees generated order- n panic automata).

An algorithmic application of game semantics. Many further directions.

Outline

1. Basic Definitions: Recursion Schemes, MSO Logic, Modal Mu-Calculus
2. Survey: Infinite Structures with Decidable MSO Theories
3. MSO Decidability of $\langle \mathbf{RecSchTree}_n \rangle_{n \in \omega}$
4. Automata-Theoretic Characterization of $\langle \mathbf{RecSchTree}_n \rangle_{n \in \omega}$.

Order- n (Deterministic) Recursion Scheme $G = (\mathcal{N}, \Sigma, \mathcal{R}, S)$

Fix a set Var of typed variables (written as φ, x, y etc).

- \mathcal{N} : Typed **non-terminals** of order at most n (written as upper-case letters), including a distinguished **start symbol** $S : o$.
- Σ : **Ranked alphabet of terminals**: $\underline{f} \in \Sigma$ has **arity** $ar(\underline{f}) \geq 0$ which determines a first-order type $\underline{f} : \underbrace{o \rightarrow \cdots \rightarrow o}_{ar(\underline{f})} \rightarrow o$
- \mathcal{R} : An **equation** for each $D : A_1 \rightarrow \cdots \rightarrow A_m \rightarrow o$ from \mathcal{N} of shape

$$D \varphi_1 \cdots \varphi_m = e$$

where the **applicative term** $e : o$ is constructed from

- terminals $\underline{f}, \underline{g}, \underline{a}$, etc. from Σ
- variables $\varphi_1 : A_1, \dots, \varphi_m : A_m$ from Var ,
- non-terminals D, F, G , etc. from \mathcal{N} .

An order-1 example. $\Sigma = \{ \underline{f}, \underline{g}, \underline{a} \}$. Take

$$G_1 : \begin{cases} S & = & F \underline{a} \\ F x & = & \underline{f} x (F (\underline{g} x)) \end{cases}$$

The **value tree** $\llbracket G \rrbracket$ of (or **tree generated by**) a recursion scheme G is a possibly infinite applicative term *constructed from the terminals*, which is obtained by unfolding the equations *ad infinitum*, replacing formal by actual parameters each time, starting from S .

E.g. We have $\llbracket G_1 \rrbracket = f a (f (g a) (f (g (g a))(\dots)))$.

View infinite term $\llbracket G \rrbracket$ as a **Σ -labelled (ranked and ordered) tree**.

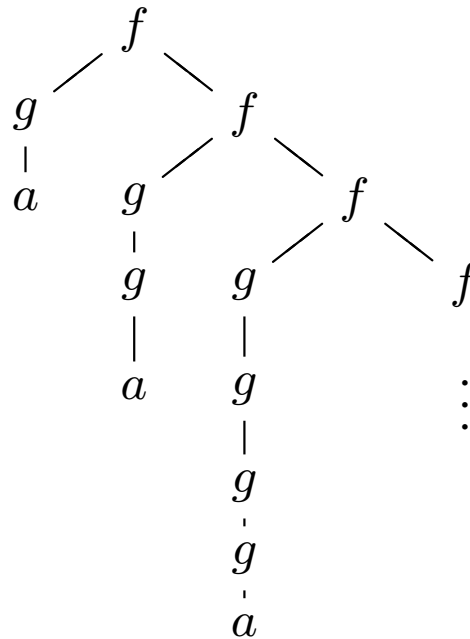
Formally a **Σ -labelled tree** is a function $t : \text{dom}(t) \longrightarrow \Sigma$ such that $\text{dom}(t) \subseteq \{ 1, \dots, m \}^*$ is prefix-closed, and for all **nodes** $\alpha \in T$, the Σ -symbol $t(\alpha) \in \Sigma$ has arity k iff α has k children, namely $\alpha 1, \dots, \alpha k \in T$.

An order-2 example.

$$\Sigma = \{ \underline{f}, \underline{g}, \underline{a} \}. \quad B : (o \rightarrow o) \rightarrow (o \rightarrow o) \rightarrow o \rightarrow o, \quad F : (o \rightarrow o) \rightarrow o$$

$$G_2 : \begin{cases} S & = & F \underline{g} \\ B \varphi \psi x & = & \varphi (\underline{\psi} x) \\ F \varphi & = & \underline{f} (\varphi \underline{a}) (F (B \varphi \varphi)) \end{cases}$$

$$\text{The value tree, } \llbracket G_2 \rrbracket : \{ 1, 2 \}^* \longrightarrow \Sigma, \text{ is: } \begin{cases} \epsilon & \mapsto & f & 11 & \mapsto & a \\ 1 & \mapsto & g & 21 & \mapsto & g \\ 2 & \mapsto & f & 22 & \mapsto & f \\ & & \dots & \dots & & \dots \end{cases}$$



For $n \geq 0$ write ***RecSchTree* _{n}** for the class of Σ -labelled trees generated by order- n recursion schemes. ***RecSchTree*₀** = { regular trees };
***RecSchTree*₁** = { algebraic trees }.

Here \mathcal{L} is MSO Logic or modal mu-calculus.

\mathcal{L} -MODEL-CHECKING PROBLEM FOR *RecSchTree* _{n}

{
 INSTANCE: An order- n recursion scheme G , and a formula $\varphi \in \mathcal{L}$
 QUESTION: Does the Σ -labelled tree $\llbracket G \rrbracket$ satisfy φ ?

Two problems about the tree hierarchy $\langle \text{RecSchTree}_n \rangle_{n \in \omega}$

1. **Decidability.** For which $n \geq 2$ is the problem decidable? If so, what is the complexity for the order- n problem?
2. Find **automata-theoretic characterization** of ***RecSchTree* _{n}** .

We use game semantics to solve the problems.

Monadic Second-Order Logic (for Σ -labelled trees $t : T \longrightarrow \Sigma$)

First-order variables: x, y, z , etc. (ranging over *nodes*, which are finite words over $\{1, \dots, m\}$, for a fixed m)

Second-order variables: X, Y, Z , etc. (ranging over *sets* of nodes i.e. *monadic* relations)

MSO formulas are built up from **atomic formulas**:

1. **Parent-child relationship between nodes**: $\mathbf{d}_i(x, y) \equiv$ “ y is i -child of x ”
2. **Node labelling**: $\mathbf{p}_f(x) \equiv$ “ x has label f ” where f is a Σ -symbol
3. **Set-membership**: $x \in X$

and closed under

- boolean connectives: \neg, \vee etc.
- first-order quantifications: $\forall x. -, \exists x. -$
- second-order quantifications: $\forall X. -, \exists X. -$.

Why MSO Logic?

It is a kind of **gold standard** in Verification!

- **MSO is very expressive.** All standard temporal logics (e.g. LTL, CTL, CTL*, etc.) embed into modal mu-calculus which embeds into MSO.
- **Any obvious extension of MSO would break decidability.**

Examples:

- **Reachability property:** “ X is a path”
- “ X is a **cut**” i.e. no two nodes in it are \leq -compatible, and it has a non-empty intersection with every maximal path; “ X is finite”.
- **Recurrence condition:** “There are **infinitely** many occurrences of the symbol $f : o \rightarrow o$.”

But “MSO cannot count”: E.g. “ X has twice as many elements as Y ”.

Outline

1. Basic Definitions: Recursion Schemes, MSO Logic, Modal Mu-Calculus
2. **Survey: Infinite Structures with Decidable MSO Theories**
3. MSO Decidability of $\langle \mathbf{RecSchTree}_n \rangle_{n \in \omega}$
4. Automata-Theoretic Characterization of $\langle \mathbf{RecSchTree}_n \rangle_{n \in \omega}$.

Structures with Decidable MSO Theories: *Some Milestones*

1. **Rabin 1969**: Regular trees. “Mother of all decidability results”
2. **Muller and Schupp 1985**: Configuration graphs of pushdown automata.
3. **Caucal (ICALP 1996)**: Prefix-recognizable graphs (= ϵ -closures of configuration graphs of pushdown automata, **Stirling 2000**).
4. **Knapik, Niwiński and Urzyczyn (TLCA 2001, FOSSACS 2002)**:
PushdownTree _{n} Σ = Trees generated by order- n pushdown automata.
SafeRecSchTree _{n} Σ = Trees generated by order- n **safe** recursion schemes.
5. **Caucal (MFCS 2002)**. *CaucalTree* _{n} Σ and *CaucalGraph* _{n} Σ .

Theorem (KNU-C). For every $n \geq 0$,

$$\mathit{PushdownTree}_n \Sigma = \mathit{SafeRecSchTree}_n \Sigma = \mathit{CaucalTree}_n \Sigma.$$

Question. Do Σ -labelled trees generated by **unsafe** recursion schemes have decidable MSO theories? If so, at which orders?

Hierarchies of Finitely-Presentable Infinite Structures

Safety seems a robust definition: several characterisations

<i>Equivalent</i> Higher-Order Generating Devices	Classes of Structures		
	Word Languages	Trees	Graphs
Pushdown Automata	Maslov 74, 76	KNU 02	Cachat, Caucal, etc.
Safe Recursion Schemes	Damm 82	KNU 02	?
Indexed Grammars	Maslov 76	?	?

	Word Languages	Trees
Order 0	Regular languages	Regular trees (Rabin, etc.)
Order 1	Context-free languages; e.g. $a^n b^n$	Algebraic trees (Bourcelle, etc.)
Order 2	Indexed languages; e.g. $a^n b^n c^n$	Hyperalgebraic trees (KNU 01)
...

What is the safety constraint?

W. Damm: [Derived types](#) in “IO and OI Hierarchies”, TCS 1982.

Definition [KNU02]. An order-2 equation is *unsafe* if the RHS has a sub-term P such that

- (i) P is order 1
- (ii) P occurs in an [operand](#) position (i.e. as 2nd argument of the application operator)
- (iii) P contains an order-0 parameter.

Examples of unsafe equations: $f : o^2 \rightarrow o$, $G, H : o$.

$$\begin{aligned} G x &= H (f x) \\ F \varphi x y &= f (F (F \varphi y) y (\varphi x)) a \end{aligned}$$

Safety (as presented above) seems syntactically awkward and semantically unnatural but (we shall see shortly) it has important algorithmic value.

In what sense is a safe λ -term *safe*?

A basic idea in lambda calculus / logic:

When performing β -reduction, one must use *capture-avoiding* substitution, which is standardly implemented by *renaming bound variables* afresh upon each substitution.

There is a price to pay for renaming:

Any machine that correctly computes:

$$\left\{ \begin{array}{l} \text{INPUT:} \quad \text{A simply-typed } \lambda\text{-term } M \\ \text{OUTPUT:} \quad \text{A } \beta\text{-reduction sequence from } M \end{array} \right.$$

needs an *unbounded* supply of names, and hence unbounded memory.

Safety lets us get away with no renaming of bound variables!

Safety reformulated as a simply-typed theory

We reexpress (and generalize) the safety constraint as a simply-typed theory. Sequents have the form

$$\underbrace{x_1 : A_1, \dots, x_i : A_i}_{\text{order } l_1} \mid \dots \mid \underbrace{x_l : A_l, \dots, x_n : A_n}_{\text{order } l_m} \vdash M : B$$

- Each A_i and B are **homogeneous**^a.
- Typing context **partitioned** according to orders with $l_1 \geq \dots \geq l_m$.

Formation rules must respect the partition:

- When forming abstraction, **all** variables of the lowest type-partition must be abstracted in an atomic step.
- When forming application, the operator-term must be applied to **all** operand-terms (one for each type) of the highest type-partition, in one atomic step.

^a o is **homogeneous**; and $(A_1 \rightarrow \dots \rightarrow A_n \rightarrow o)$ is **homogeneous** just if $order(A_1) \geq order(A_2) \geq \dots \geq order(A_n)$, and each A_i is homogeneous.

Safe λ -Calculus: System \mathcal{S} Typing Rules

$$\frac{(\overline{A_1} \mid \cdots \mid \overline{A_n} \mid o) \text{ homogeneous} \quad b \text{ is a type-}B \text{ constant}}{\overline{x_1} : \overline{A_1} \mid \cdots \mid \overline{x_n} : \overline{A_n} \vdash b : B}$$

$$\frac{(\overline{A_1} \mid \cdots \mid \overline{A_n} \mid o) \text{ homogeneous}}{\overline{x_1} : \overline{A_1} \mid \cdots \mid \overline{x_n} : \overline{A_n} \vdash x_{ij} : A_{ij}}$$

$$\frac{\overline{x_1} : \overline{A_1} \mid \cdots \mid \overline{x_{n+1}} : \overline{A_{n+1}} \vdash M : B \quad (\overline{A_{n+1}} \mid B) \text{ homogeneous}}{\overline{x_1} : \overline{A_1} \mid \cdots \mid \overline{x_n} : \overline{A_n} \vdash \lambda \overline{x_{n+1}}.M : (\overline{A_{n+1}} \mid B)}$$

$$\frac{\Gamma \vdash M : (\overline{B_1} \mid \cdots \mid \overline{B_m} \mid o) \quad \Gamma \vdash N_1 : B_{11} \cdots \Gamma \vdash N_{l_1} : B_{1l_1}}{\Gamma \vdash MN_1 \cdots N_{l_1} : (\overline{B_2} \mid \cdots \mid \overline{B_m} \mid o)}$$

When forming abstraction, **all variables** of the lowest-order type-partition must be abstracted. When forming application, the operator-term must be applied to **all operand-terms** (one for each type) of the highest-order type-partition.

Safe λ -calculus makes algorithmic sense

Example. Suppose $\underline{f} : o^2 \rightarrow o$. Variable capture can arise in

$$(\lambda\varphi^{(o,o)}.(\lambda x.\varphi x)) (\underline{f} x)$$

The term is *not* safe: \underline{f} should be applied to 2 arguments.

Theorem. “Safe λ -calculus = α -conversion-free λ -calculus”

In the safe lambda calculus, there is no need to rename bound variables when performing substitution $M[N_1/\varphi_1, \dots, N_n/\varphi_n]$ provided the substitution is performed **simultaneously** on **all** free variables of the same order in M .

Proof idea. Suppose φ free in M , and x free in N , and x captured in (capture permitting) $M[N/\varphi]$. Then M looks like $\dots (\lambda x.\dots \varphi \dots) \dots$.

Case analysis by comparing $order(x)$ with $order(\varphi)$.

Lemma. A free variable in a safe term has order as least that of the term. \square

Thus when reducing a safe λ -term, we do not need any supply of fresh name.

What is the right way to think of the Safe Lambda Calculus?

Safe λ -calculus seems of independent interest, and we don't understand it.

Design issues: Is the homogeneity assumption really necessary?

Proof theory: What kind of **reasoning principles** does it support (via Curry-Howard)? Is it useful to automated deduction / theorem proving?

What is a **model** of safe λ -calculus? Does it have interesting models?

Game semantics: What kind of **pointer economy** does safety determine? Ans: Pointers are redundant in safe view-functions!

E.g. Kierstead terms: $\lambda f.f(\lambda x.f(\lambda y.y))$ is safe, but $\lambda f.f(\lambda x.f(\lambda y.x))$ is unsafe.

Implicit complexity. Simply-typed λ -calculus characterize polytime-computable numeric functions (Leivant-Marion 93). What about the safe terms?

Nevertheless, we shall prove that safety is *not* necessary for MSO decidability.

Two questions about *Safety*

Is safety a genuine or spurious constraint for:

1. **Expressiveness.** Are there *inherently* unsafe Σ -labelled trees?

I.e. Is there an unsafe recursion scheme whose value tree is not the value tree of any safe recursion scheme? If so, at what order?

Conjecture. Yes, at order 2. But note:

Theorem. (A+deM+O FOSSACS 2005) There is no inherently unsafe **word language** at order 2.

2. **Decidability.** Is safety necessary for decidability? Some partial results:

Theorem. (A+deM+O 05 / KNUW 05) Σ -labelled trees denoted by order-2 recursion schemes, **whether safe or not**, have decidable MSO theories.

Question. What about higher orders?

Yes: Decidability result extends to all orders.

Outline

1. Basic Definitions: Recursion Schemes, MSO Logic, Modal Mu-Calculus
2. Survey: Infinite Structures with Decidable MSO Theories
3. **MSO Decidability of $\langle \mathit{RecSchTree}_n \rangle_{n \in \omega}$**
4. Automata-Theoretic Characterization of $\langle \mathit{RecSchTree}_n \rangle_{n \in \omega}$.

Theorem. “Safety is not necessary for MSO decidability.”

The modal mu-calculus model checking problem for trees generated by arbitrary order- n recursion schemes is n -EXPTIME complete, for each $n \geq 0$.

Three major ingredients:

1. A **transference principle** from tree generated from the recursion scheme – **value tree** – to an auxiliary **computation tree**.
A **strong correspondence** between **paths** in the value tree and **traversals** in the computation tree – proof is by game semantics.
2. **Simulation** of (accepting) traversal-tree by a certain set of annotated paths over computation tree; the latter recognised by **traversal-simulating alternating parity tree automaton** (APT).
3. Complexity analysis of equivalent **acceptance parity game** to a **finite graph** (which generates the computation tree) and the traversal-simulating APT.

Transference Principle: from *value* tree to *computation* tree

Direct algorithmic analysis of **value tree** $\llbracket G \rrbracket$ (i.e. tree **generated** by the given recursion scheme) is futile:

- Value tree has no useful structure for our purpose.
- It is the “extensional” outcome of a (potentially infinite) computational process, the algorithmics of which are what we *should* analyse.

Our approach:

We consider an auxiliary **computation tree** $\lambda(G)$ which recovers useful intensional information about the computational process behind the construction of the value tree.

Indeed it is **static** view of the very computational process.

Roughly speaking, “**evaluating the computation tree (by contracting the β -redexes)** returns the value tree”.

The Long Transform: from G to \overline{G}

\overline{G} -rules are obtained by: For each G -rule

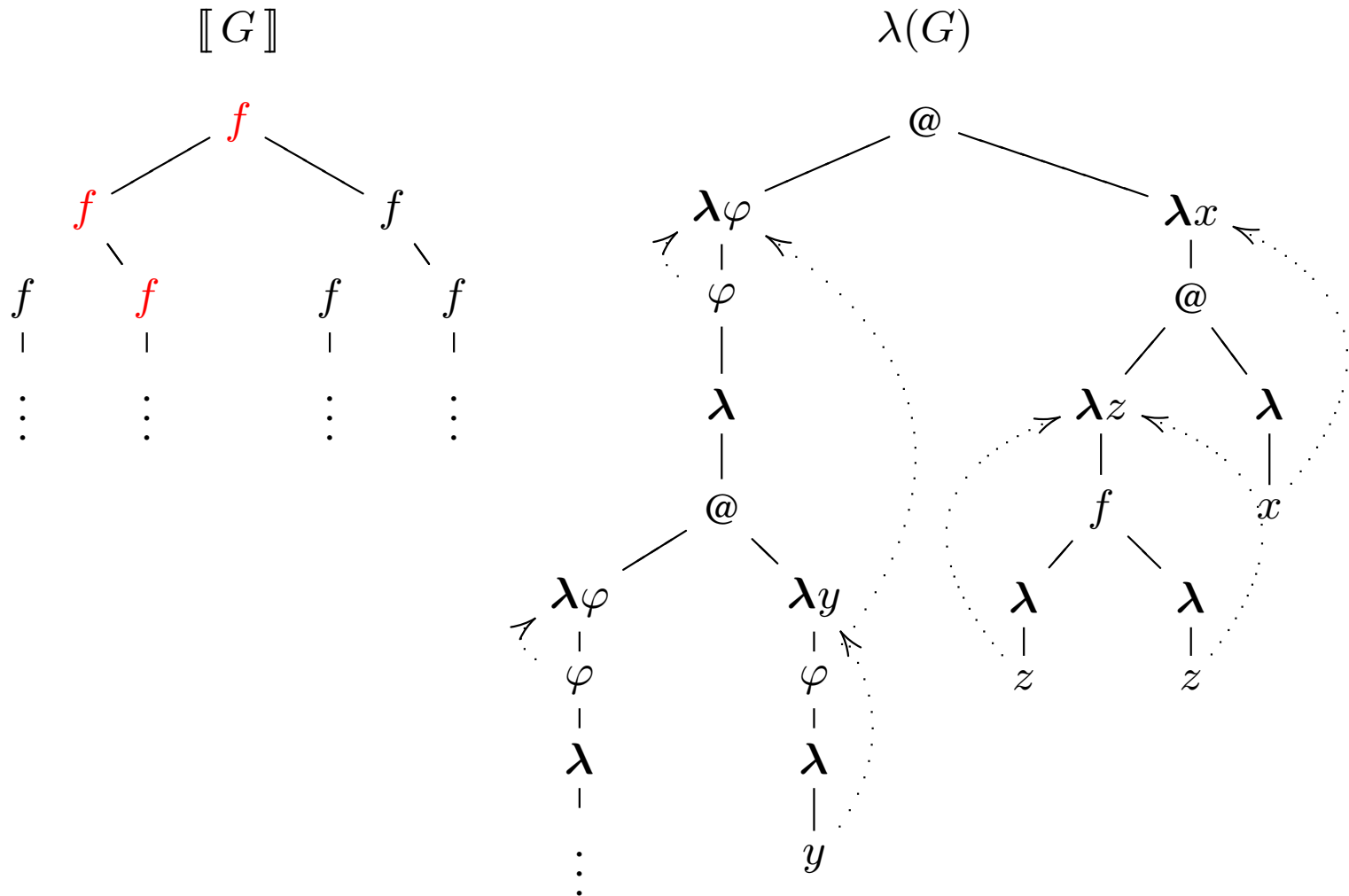
1. **Expand RHS to its η -long form**, including ground-type subterm in *operand* position. Thus $e : o$ η -expands to $\lambda.e$ (“dummy lambdas”).
2. **Insert long-apply symbol @**: Replace every ground-type subterm $D e_1 \cdots e_n$ by $@ D e_1 \cdots e_n$, where D ranges over non-terminals.
3. **Curry each equation.**
4. **Rename (bound) variables afresh.** Only finitely many new names.

Example.

$$G : \begin{cases} S = F H \\ F \varphi = \varphi (F \varphi) \\ H z = f z z \end{cases} \mapsto \overline{G} : \begin{cases} S = @ F (\lambda x. @ H \lambda. x) \\ F = \lambda \varphi. \varphi (\lambda. @ F (\lambda y. \varphi (\lambda. y))) \\ H = \lambda z. f (\lambda. z) (\lambda. z) \end{cases}$$

$$G : \begin{cases} S & = & F H \\ F \varphi & = & \varphi (F \varphi) \\ H z & = & f z z \end{cases} \quad \mapsto \quad \bar{G} : \begin{cases} S & = & @ F (\lambda x. @ H \lambda x) \\ F & = & \lambda \varphi. \varphi (\lambda. @ F (\lambda y. \varphi (\lambda. y))) \\ H & = & \lambda z. f (\lambda z) (\lambda z) \end{cases}$$

The **computation tree** $\lambda(G)$ is the term-tree obtained by infinitely unfolding \bar{G} :



Definition. *Traversals* over $\lambda(G)$ are justified sequences defined by induction:

(Root) The singleton sequence (comprising ϵ) is a traversal.

(App) If $t @$ is a traversal, so is $t @ \overset{\swarrow 0}{\lambda \bar{\xi}}$.

(Sig) If $t f$ is a traversal, so is $t f \overset{\swarrow i}{\lambda}$ where $1 \leq i \leq \text{arity}(f)$.

(Var) If $t n \overset{\swarrow i}{\lambda \bar{\xi}} \dots \xi$ is a traversal, so is $t n \overset{\swarrow i}{\lambda \bar{\xi}} \dots \xi \overset{\swarrow i}{\lambda \bar{\eta}}$.

(Lam) If $t \lambda \bar{\xi}$ is a traversal, so is $t \lambda \bar{\xi} n$, such that $\lceil t \lambda \bar{\xi} n \rceil$ is a path in $\lambda(G)$.

Key Lemma:

- (i) Traversals are justified sequences that satisfy Visibility.
- (ii) **P-views of traversals are paths in the computation tree.**

Theorem. (Correspondence) Let G be an order- n recursion scheme.

- (i) There is a 1-1 correspondence between **maximal paths** p in (Σ -labelled) value tree $\llbracket G \rrbracket$ and **maximal traversals** t_p over computation tree $\lambda(G)$.
- (ii) Further for each p , we have $p \upharpoonright \Sigma = t_p \upharpoonright \Sigma$.

Proof is by game semantics.

Idea:

“Local computation of β -reduction”

GoI:	Lamping graph	regular (consistent, legal) paths
Game semantics:	computation trees	traversals

Value tree $\llbracket G \rrbracket$ is a concrete representation of the strategy-denotation of G (in game semantics).

Paths in $\llbracket G \rrbracket$ correspond to **plays** in the strategy-denotation.

Traversals t_p over computation tree $\lambda(G)$ are just (representations of) the **uncoverings** of the plays (= path) p in the strategy-denotation of G .

From run-tree over $\llbracket G \rrbracket$ to traversal-tree over $\lambda(G)$

Fact. (i) Over trees, MSO and mu-calculus are equi-expressive.
(ii) Mu-calculus and **alternating parity tree automata (APT)** are equivalent.

Thus: Property APT \mathcal{B} has an **accepting run-tree** over $\llbracket G \rrbracket$

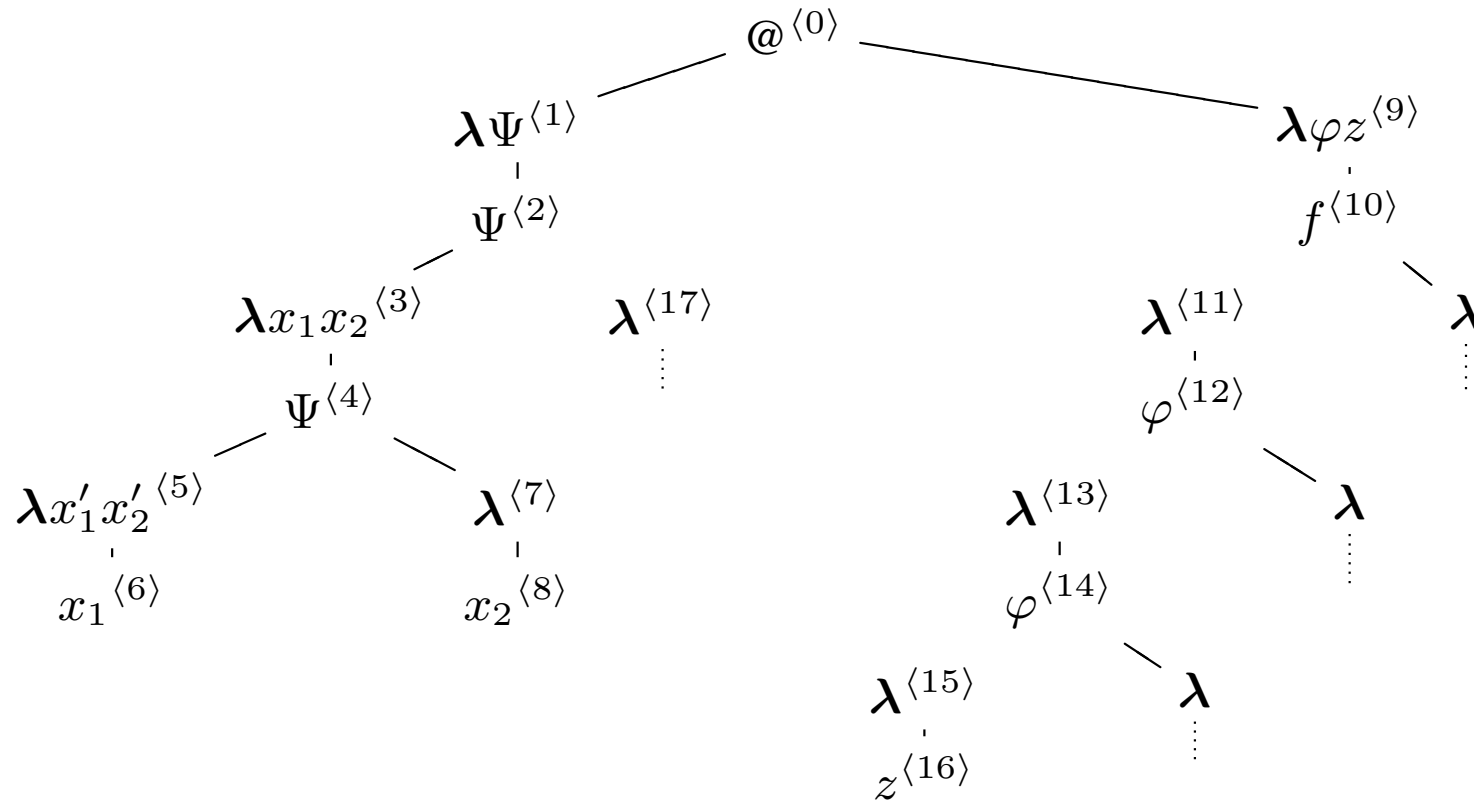
by def. $\left\{ \begin{array}{l} \exists \text{ certain set of } \delta_{\mathcal{B}}\text{-respecting, state-annotated maximal} \\ \text{paths in } \llbracket G \rrbracket \text{ satisfying parity condition} \end{array} \right.$

Thm (Corr) $\left\{ \begin{array}{l} \exists \text{ certain set of } \delta_{\mathcal{B}}\text{-respecting, state-annotated maximal} \\ \text{traversals over } \lambda(G) \text{ satisfying parity condition} \end{array} \right.$

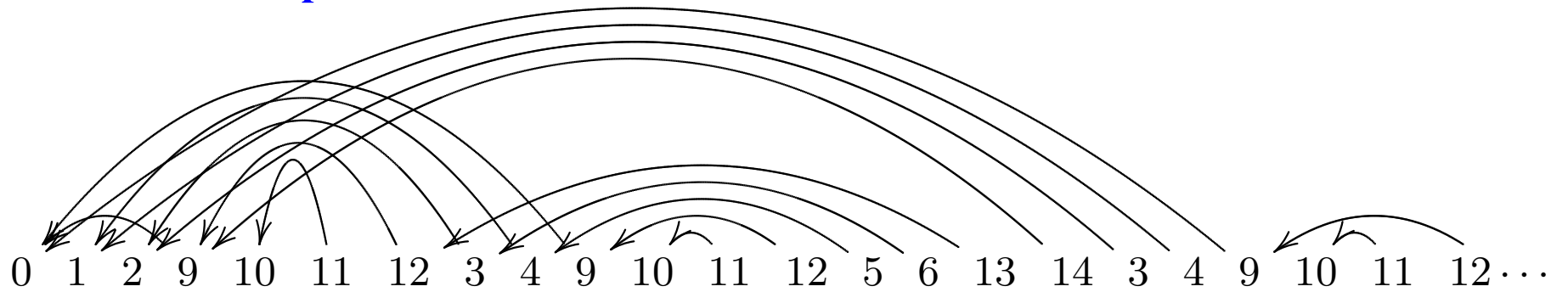
new def. $\left\{ \begin{array}{l} \text{Property APT } \mathcal{B} \text{ has an } \text{accepting traversal-tree} \text{ over } \lambda(G). \end{array} \right.$

Higher-order traversals can be very complex - they jump all over the tree, and can visit certain nodes infinitely often. (See order-3 example!)

Problem: Find a device to recognise an accepting traversal-tree.

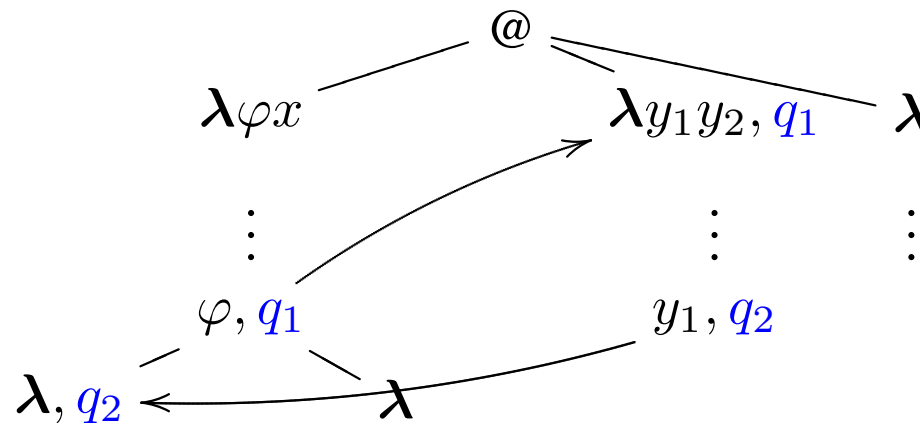


An order-3 example:



Simulate traversals by *paths* – an order-2 illustration

Idea. Simulate an annotated traversal by the respective **P-views** of all its prefixes, which are a set of annotated paths in the computation tree.



Suppose a traversal jumps from φ at simulating state q_1 to a sibling subtree rooted at $\lambda y_1 y_2$, subsequently exits it at y_1 and rejoins the original subtree at first λ -child of φ with state q_2 .

Simulate the traversal by **paths**:

- At φ with q_1 , **guess** that the detour will return at first λ -child with state q_2
- **Spawn** an automaton at $\lambda y_1 y_2$ to **verify the guess**.

Formalising the guesses as Variable Profiles $\mathbf{VP}_G^{\mathcal{B}}(A)$

Fix a recursion scheme G , and a property APT $\mathcal{B} = \langle \Sigma, Q, \delta, q_0, \Omega \rangle$ with p priorities. Write $[p] = \{1, \dots, p\}$.

$$\begin{aligned}\mathbf{VP}_G^{\mathcal{B}}(o) &= \text{Var}_G^o \times Q \times [p] \times 2^{\emptyset} \\ \mathbf{VP}_G^{\mathcal{B}}(A_1 \rightarrow \dots \rightarrow A_n \rightarrow o) &= \text{Var}_G^A \times Q \times [p] \times 2^{(\bigcup_{i=1}^n \mathbf{VP}_G^{\mathcal{B}}(A_i))}\end{aligned}$$

Asserting $(\varphi, q, m, c) \in \mathbf{VP}_G^{\mathcal{B}}(A)$ at node α of computation tree means: the traversal being simulated will reach some descendant-node that is labelled φ

- (i) with state q , such that
- (ii) m is the highest priority that will have been encountered up to that point
- (iii) further, the traversal (which will then jump to the root of a subtree that denotes the *actual* argument of φ) will eventually return to the children of the node labelled φ “in accord with c ”.

Theorem (Simulation). The following are equivalent:

- (i) Property APT \mathcal{B} has an accepting traversal-tree over the computation tree $\lambda(G)$.
- (ii) Traversal-simulating APT \mathcal{C} has an accepting run-tree over the computation tree $\lambda(G)$.

“(i) \Rightarrow (ii)” : From the traversal-tree annotated only by \mathcal{B} -states, we perform a succession of annotation operations, transforming it to a traversal-tree annotated by \mathcal{C} -states.

The set of P-views of all such \mathcal{C} -state-annotated traversals is precisely an accepting run-tree of \mathcal{C} .

“(ii) \Rightarrow (i)” : Reconstruct each traversal (of the putative traversal-tree) as a sequence of segments of paths (=P-views) in the accepting run-tree, thus inheriting an accepting state-annotation.

Satisfication of parity condition tricky!

Key Steps of the Decidability Proof: order- n G

Value tree $\llbracket G \rrbracket$ satisfies modal mu-calculus formula φ

\iff { Emerson + Jutla 1991 }

Property APT \mathcal{B} has accepting run-tree over $\llbracket G \rrbracket$

\iff { Correspondence Theorem }

\mathcal{B} has an accepting **traversal-tree** over **computation tree** $\lambda(G)$

\iff { Simulation Theorem }

Traversal-simulating APT \mathcal{C} has an accepting run-tree over $\lambda(G)$

\iff { Stirling, Vardi et al. etc. }

Eloise has winning strategy in acceptance parity game $\mathbf{G}(\text{Gr}(G), \mathcal{C})$

for finite graph $\text{Gr}(G)$ with $\text{unfold}(\text{Gr}(G)) = \lambda(G)$

Solving $\mathbf{G}(\text{Gr}(G), \mathcal{C})$ is $\exp_n O(|G| \cdot |Q| \cdot p)$ where APT \mathcal{C} has state-set Q and p priorities.

Understanding traversals

Some questions

1. For each $n \geq 0$, what kind of computing power do we need to compute traversals?
2. Can we characterize them automata-theoretically?
3. Find equivalent definitions (characterizations) of $\langle \mathbf{RecSchTree}_n \rangle_{n \in \omega}$.

Outline

1. Basic Definitions: Recursion Schemes, MSO Logic, Modal Mu-Calculus
2. Survey: Infinite Structures with Decidable MSO Theories
3. MSO Decidability of $\langle \mathbf{RecSchTree}_n \rangle_{n \in \omega}$
4. **Automata-Theoretic Characterization of $\langle \mathbf{RecSchTree}_n \rangle_{n \in \omega}$.**

Order-2 Pushdown Automata $\langle \Sigma, Q, Z, \delta, q_0 \rangle$

An **2-stack** (resp. $n + 1$ -stack) is a stack of 1-stacks (resp. n -stack).

2-stack operations $\theta \in Inst_2$:

$$\text{push}_2 : s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_n]}_{s_i} \mapsto s_1 \cdots s_{i-1} s_i s_i$$

$$\text{pop}_2 : s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_n]}_{s_i} \mapsto s_1 \cdots s_{i-1}$$

$$\text{push}_1(a) : s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_n]}_{s_i} \mapsto s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_n a]}$$

$$\text{pop}_1 : s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_n]}_{s_i} \mapsto s_1 \cdots s_{i-1} \underbrace{[a_1 \cdots a_{n-1}]}$$

Transition map: $\delta : Q \times Z \longrightarrow Q \times Inst_2$ induces small-step relation over configurations:

$$q s \rightarrow q' s' \quad \text{iff} \quad \delta(q, \text{top}_1(s)) = (q', \theta) \wedge s' = \theta(s).$$

Order- n Panic Automata

(Order-2 case by [AdMO05] + [KNUW05]; we generalize it to all orders)

Order-2 case for illustration:

Each stack symbol has a **pointer** to an earlier 1-stack (= a record of the stack content at the point of “creation”).

Each $j_1, \dots, j_n \leq i$.

$$\begin{aligned} \text{push}_1(a) &: s_1 \cdots s_i \underbrace{[a_1^{j_1} \cdots a_n^{j_n}]}_{s_{i+1}} \mapsto s_1 \cdots s_i [a_1^{j_1} \cdots a_n^{j_n} a^i] \\ \text{panic} &: s_1 \cdots s_i \underbrace{[a_1^{j_1} \cdots a_n^{j_n}]}_{s_{i+1}} \mapsto s_1 \cdots s_k \quad \text{where } j_n = k \end{aligned}$$

Panic “resets” automaton to the stack content at which current top element was first created.

Extension to order- n : $\text{push}_1^i(a)$ where $2 \leq i \leq n$

Order- n Recursion Schemes = Order- n Panic Automata

Theorem. For each $n \geq 0$, order- n panic automata and order- n recursion schemes are equi-expressive for Σ -labelled trees. I.e.

$$\mathit{RecSchTree}_n \Sigma = \mathit{PanicAutoTree}_n \Sigma$$

Proof idea

- From recursion scheme G to panic automaton \mathcal{A}_G :

Use game semantics.

Code traversals as n -stacks.

Invariant: The top 1-stack is the P-view of the encoded traversal.

- From panic automaton \mathcal{A} to recursion scheme $G_{\mathcal{A}}$:

Code configurations c as Σ -term M_c , so that $c \rightarrow c'$ implies M_c rewrites (in 1-step) to $M_{c'}$.

Many Further Directions

1. Is safety a genuine constraint on expressiveness? Equivalently, are order- n panic automata” more expressive order- n pushdown automata?

Conjecture. $\text{SafeRecSch}_2\Sigma \subset \text{RecSchTree}_2\Sigma$ I.e. There are *inherently* unsafe trees (at order 2).

Candidate: Urzyczyn’s tree.

2. Define **graphs generated by order- n recursion schemes** to be ϵ -closures of configuration graphs of order- n panic automata? Are their MSO theories decidable?

3. **“Mixing semantic and verification games”**: Denotational semantics of λ -calculus “relative to an alternating parity tree automaton (APT)”.

Problem. Construct a cartesian closed category (= model of the lambda calculus), parameterized by an APT, whose maps are witnessed by profiles (“guesses”).