# Algorithmic Reasoning about Böhm Trees

Luke Ong

University of Oxford

(Joint with Bahareh Afshari, Matthew Hague, Graham Leigh, Steven Ramsay, and Takeshi Tsukada)

IMS Workshop on Higher-Order Model Checking

Singapore, 20-23 September 2016

## What is higher order about higher-order model checking (HOMC)?

Higher-Order Model Checking Problem: Given HORS $\mathcal{G}$ and property $\varphi$, does the tree $[\![\mathcal{G}]\!]$ satisfy $\varphi$?

The generator $\mathcal{G}$ is higher order, but the tree $[\![\mathcal{G}]\!]$ is not.

**Desiderata:**

1. Model check (higher-type) Böhm trees.
   Why? Böhm trees are the computation trees of higher-type functional programs.
2. Compositionalise higher-order model checking.
   HOMC is (mostly) whole program analysis. This can seem surprising since higher order is supposed to aid modular structuring of programs.

Challenging, because the elegant theorems of “Rabin’s Heaven” fail in the world of Böhm trees.

## Overview of Böhm Trees

- Böhm trees are the term-trees of possibly infinite $\lambda$-terms in $\beta$-NF.
- Assume Böhm trees are well-sorted (i.e. simply-typed) and $\eta$-long, and use only finitely many free variables (names).

Böhm trees are term-trees defined coinductively $(n \geq 0)$:

$$t^o ::= \bot \mid x^A \, s_1^{A_1} \, \cdots \, s_n^{A_n}$$
$$s^A ::= \lambda x_1^{A_1} \cdots x_n^{A_n}.t^o \qquad \text{where } A = A_1 \to \cdots \to A_n \to o$$

- Böhm trees subsume ordinary (node-labelled, ranked) trees.
- Alternative presentation as $\Sigma$-binding trees (a version of data trees).

Böhm trees (of composable sorts) can compose. Thus Böhm trees are higher-order functions on trees.

- $\lambda\mathbf{Y}$-definable Böhm trees are a natural class of finitely-presentable Böhm trees.

## Example Böhm Tree ("Semi-infinite Grid"): $u_\infty$

$u_\infty$ uses infinitely many variable names, and each variable occurs infinitely often.

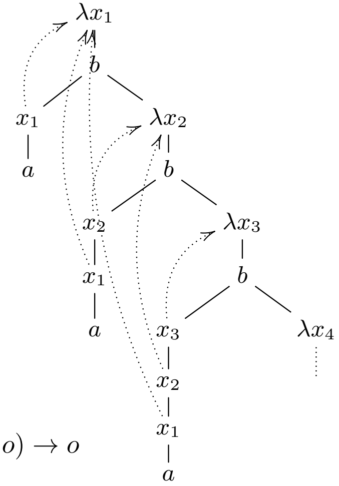
$u_\infty$ has an undecidable MSO theory (Salvati; Clairambault & Murawski FSTTCS13).

$u_\infty$ is $\lambda\mathbf{Y}$-definable of order 4:

$u_\infty = \mathsf{BT}(M)$ where

$$\Gamma \ \vdash \ \underbrace{\mathbf{Y}\,(\lambda f.\lambda y^o.\lambda x^{o\to o}.b\,(x\,y)\,(f\,(x\,y)))\,a}_{M} \ : \ (o \to o) \to o$$

with $\Gamma = a : o,\ b : o \to ((o \to o) \to o) \to o$.

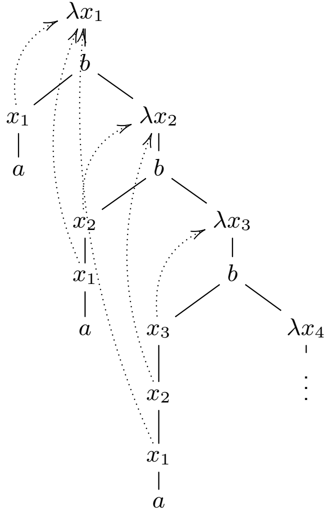


## Model Checking Higher-type Böhm Trees (Tsukada & O. LICS14)

Property $\Phi :=$
"There are only finitely many occurrences of bound variables, $x_i$s, in each branch."

Questions:

1. Is there a "logic" that can describe properties such as $\Phi$?
2. Is the "logic" decidable for Böhm trees?



Questions:

1. Is there a "logic" that can describe properties such as $\Phi :=$ "There are only finitely many occurrences of bound variables, $x_i$s, in each branch"?

2. Is the "logic" decidable for Böhm trees?

Answers: YES, (intersection) types, & YES. (Tsukada & O. LICS14)

1. A semantics: type-checking game, $\Gamma \models t : \tau$, for Böhm trees $t$ and types $\tau$
2. A decidable, complete proof system, $\Gamma \vdash M : \tau$, for $\lambda\mathbf{Y}$-definable Böhm trees: $\Gamma \vdash M : \tau \iff \Gamma \models \mathrm{BT}(M) : \tau$

E.g. Böhm tree $t$ satisfies $\Phi$ just if $\Gamma \models t : (((q,1) \to q,1) \to q,0)$ where $\Gamma = a : (q,1), b : ((q,0) \to (((q,1) \to q,1) \to q,0) \to q,1)$.

This talk: What kind of automata and (modal) logic correspond to these intersection types for reasoning about Böhm trees?

These devices may aid understanding and facilitate applications.

## Recall: A “Classical” Automata-Logic-Games Correspondence

$$\begin{tikzcd} \text{APT} \arrow[rr, equal] \arrow[dr, equal] & & L_\mu \arrow[dl, equal] \\ & \text{Parity Games} & \end{tikzcd}$$

$L_\mu$ (Mu-Calculus) Model Checking Problem and PARITY are inter-reducible

$\Rightarrow$: Essentially: Fundamental Semantic Theorem [Streett and Emerson Info & Comp 1989]

$\Leftarrow$: E.g. [Walukiewicz Info & Comp 2001]

$L_\mu$ and APT are effectively equi-expressive for tree languages [Emerson & Jutla FoCS 91]

This talk: We give an Automata-Logic-Games Correspondence for higher-type Böhm trees.

Summary:

$$\begin{tikzcd}
\text{Böhm Trees:} & \text{ADTA} \arrow[rr, equal] \arrow[dr, equal] \arrow[dd, dotted, no head, "\cap"'] & & L_\mu^{\to} \arrow[dl, equal] \arrow[dd, dotted, no head, "\cap"] \\
& & \text{Type-Checking Games} \arrow[dd, dotted, no head, "\cap"] & \\
\text{(Ranked) Trees:} & \text{APT} \arrow[rr, equal] \arrow[dr, equal] & & L_\mu \arrow[dl, equal] \\
& & \text{Parity Games} &
\end{tikzcd}$$

## Outline: Higher type Automata-Logic-Games Correspondences

0. Motivations: Model Checking Böhm Trees / Compositional Higher-Order Model Checking

1. Alternating Dependency Tree Automata (ADTA)

2. Higher-Type Mu-Calculus, $L_{\mu}^{\rightarrow}$

3. Conclusions and Further Directions

## Alternating Dependency Tree Automata (ADTA)

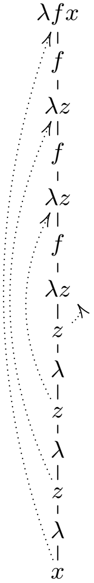
ADTA are automata over Böhm trees presented as $\Sigma$-binding trees.

Given a binder-bindee alphabet $\Sigma = \Sigma_{\mathrm{var}} \uplus \Sigma_{\lambda} \uplus \Sigma_{\mathrm{aux}}$, a $\Sigma$-binding tree is a $\Sigma$-labelled tree equipped with a binder function.
$\Sigma$-binding trees are a kind of ranked data trees.

Stirling (FoSSaCS 2009) introduced ADTA for finite Böhm trees to characterise solution sets of the Higher-Order Matching Problem.

We extend ADTA to infinite trees with $\omega$-regular conditions.

## Example: Böhm trees as $\Sigma$-binding trees



The order-4 sort

$$\mathfrak{M} = (((o \to o) \to o) \to o) \to o \to o$$

called monster, is inhabited by the terms

$$M_n := \lambda f\, x. f(\lambda z_1.(f(\lambda z_2.(\cdots f(\lambda z_n.z_n(\cdots z_2(z_1\, x)))))))$$

for all $n \geq 0$.

Let $\Sigma = \underbrace{\{f^{((o\to o)\to o)\to o},\ x^o,\ z^{o\to o}\}}_{\Sigma_{\text{var}}} \cup \underbrace{\{\lambda f\, x,\ \lambda z,\ \lambda\}}_{\Sigma_\lambda} \cup \underbrace{\{\perp\}}_{\Sigma_{\text{aux}}}$

N.B. $\Sigma$ is determined by the sort $\mathfrak{M}$.

We can represent Böhm trees of sort $\mathfrak{M}$ as a $\Sigma$-binding tree.

E.g. $M_3$ on LHS.

## Alternating Dependency Tree Automata (ADTA)

An ADTA of sort $A$ is a tuple $\mathcal{A} = (Q, \Sigma_A, Q_I, \Delta, \Omega)$

- $Q$ is a finite state-set; $Q_I \subseteq Q$ are initial states
- $\Sigma_A = \Sigma_\lambda \cup \Sigma_{\mathrm{var}} \cup \{\bot\}$ where

$$\Sigma_{\mathrm{var}} := \{x^B \mid B \text{ contravariant subsort of } A\}$$
$$\Sigma_\lambda := \{\lambda x_1^{B_1} \cdots x_m^{B_m} \mid (B_1 \to \cdots \to B_m \to o) \text{ covariant subsort of } A\}$$

- $\Delta$ is a set of transition rules: 3 kinds
  - Acceptor transitions
  - Rejecter transitions
  - $\bot$ transitions
- $\Omega$ assigns a priority to transitions

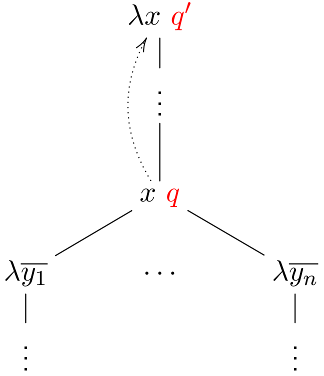
A Böhm tree $t$ is accepted by $\mathcal{A}$ if Acceptor has a winning strategy in the acceptance game $\mathcal{G}_{\mathrm{Acc}}(t, \mathcal{A})$.

## Acceptor transitions of a ADTA: $(q', q)\, x \Rightarrow (Q_1, \cdots, Q_n)$

Acceptor chooses transitions to read variables: $Q_i \subseteq Q$

$$(q', q)\, x \Rightarrow (Q_1, \cdots, Q_n)$$

Suppose we have a run to $x$:

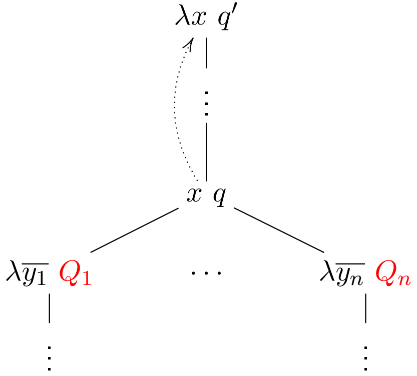


## Acceptor transitions of a ADTA: $(q', q)\, x \Rightarrow (Q_1, \cdots, Q_n)$

Acceptor chooses transitions to read variables:

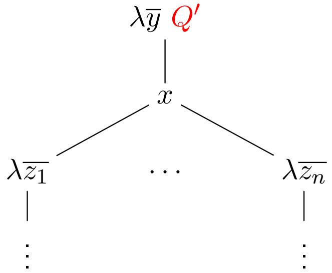
$$(q', q)\, x \Rightarrow (Q_1, \cdots, Q_n)$$

Suppose we have a run to $x$:



## Rejecter transitions of a ADTA: $q\,\lambda\overline{y} \overset{e}{\Rightarrow} q'$

Suppose Acceptor has just labelled the $\lambda\overline{y}$-node with $Q' \subseteq Q$:



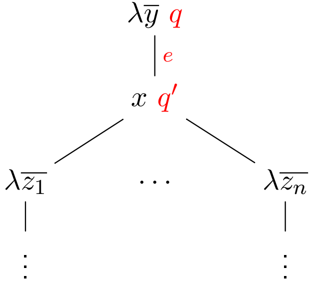
Rejecter chooses some $q \in Q'$ and a transition

$$q\,\lambda\overline{y} \overset{e}{\Rightarrow} q'$$

where $e$ is the priority assigned by $\Omega$

## Rejecter transitions of a ADTA: $q\,\lambda\overline{y} \overset{e}{\Rightarrow} q'$

Acceptor labels $\lambda$-nodes with a set of states $Q' \subseteq Q$:



Rejecter chooses some $q \in Q'$ and a transition

$$q\,\lambda\overline{y} \overset{e}{\Rightarrow} q'$$

where $e$ is the priority assigned by $\Omega$. Acceptor wins if the maximum infinitely-occurring priorities in the sequence labelling the transitions is even.

## Properties of ADTA

ADTA are closed under union and intersection (Stirling FoSSaCS09).

ADTA are closed under complementation.

- Proofs: 1. Automata-theoretic. 2. Higher-type Mu-calculus: negated formulas are equivalent to positive normal forms. 3. Types: see Tsukada's talk.

Emptiness of ADTA is undecidable.

- By reducing PCF Observational Equivalence to ADTA emptiness (O. & Tzvelekos LICS09).

Emptiness of nondeterministic DTA is decidable.

- Proofs: 1. Higher-type mu-calculus: Disjunctive formulas have Small Model Property via tableaux. 2. Type-checking games.

ADTA Acceptance of $\lambda\mathbf{Y}$-definable Böhm trees is decidable.

- Follows from decidability of type-checking games (Tsukada & O. LICS14), and equivalence between ADTA and types.

## Correspondence between ADTA and Types

Given type $\alpha :: A$, define $[\![\alpha]\!] := \{u \in \mathsf{BT}^A \mid \ \models u : \alpha\}$.

**Theorem (Effective Equi-expressivity)**

*Types and ADTA are equivalent for defining languages of Böhm trees. I.e. There are algorithms*

- *mapping ADTA $\mathcal{A}$ to types $\alpha_{\mathcal{A}}$ satisfying $L(\mathcal{A}) = [\![\alpha_{\mathcal{A}}]\!]$*
- *mapping types $\alpha$ to ADTA $\mathcal{A}_\alpha$ satisfying $[\![\alpha]\!] = L(\mathcal{A}_\alpha)$.*

A delicacy:

We suppress the distinction between types and a subsystem of parity permissive types. There is a similar distinction between ADTA and a subclass of parity permissive ADTA.

The equi-expressivity result holds at both levels (general and parity permissive).

## Higher-Type Mu-Calculus, $L_\mu^{\rightarrow}$

Formulas are sorted, $\varphi^A$.
Assume a set $\Sigma_{\mathrm{var}}$ of sorted abstract variables ranged over by $\mathsf{x}^B$, $\mathsf{y}^C$, etc.

### Syntax of $L_\mu^{\rightarrow}$

$$
\begin{aligned}
\varphi^o &::= \langle\bot\rangle \mid \mathbf{t} \mid \mathbf{f} \mid \varphi^o \wedge \varphi^o \mid \varphi^o \vee \varphi^o \mid \neg\varphi^o \mid \\
&\qquad \mathbb{v}_{\mathsf{x}^{\mathsf{B}}} \mid [i](\varphi^{B\to C}) \mid \alpha \mid \mu\alpha.\varphi^o \mid \nu\alpha.\varphi^o \\
\varphi^{B\to C} &::= \varphi^{B\to C} \wedge \varphi^{B\to C} \mid \varphi^{B\to C} \vee \varphi^{B\to C} \mid \\
&\qquad \neg\varphi^{B\to C} \mid \mathbb{\lambda}_{\mathsf{x}^{\mathsf{B}}}.\varphi^C .
\end{aligned}
$$

Two new constructs:

- abstract-variable predicate, $\mathbb{v}_{\mathsf{x}^{\mathsf{B}}} \ : \ o$
- abstract-lambda formula, $\mathbb{\lambda}_{\mathsf{x}^{\mathsf{B}}}.\varphi^C \ : \ B \to C$

which are detectors of variables and $\lambda$-abstractions respectively.

N.B. $\mathbb{\lambda}_{\mathsf{x}^{\mathsf{B}}}.$- is not a binding construct.

## Semantics of Higher-Type Mu-Calculus

Assume a set $\mathcal{V}$ of concrete variables.
A concretisation function, $\zeta : \Sigma_{\mathrm{var}} \to \mathcal{P}_{\mathrm{fin}}(\mathcal{V})$, maps an abstract variable to a finite set of concrete variables.

Idea. $[\![\varphi^B]\!]_\rho(\zeta)$ is a set of Böhm trees $t$ of sort $B$ where $\mathrm{FV}(t) \subseteq \mathrm{Im}(\zeta)$.

- $\mathbb{\lambda}_{\mathsf{x}^{\mathsf{B}}}.-$ matches an abstraction $\lambda y^B.s$ and records $y$ as a possible concrete instantiation of $\mathsf{x}^B$ in the concretisation function $\zeta$. Thus

$$[\![\mathbb{\lambda}_{\mathsf{x}^{\mathsf{B}}}.\varphi^C]\!]_\rho(\zeta) = \{\lambda y^B.s \mid s \in [\![\varphi^C]\!]_\rho(\zeta[\mathsf{x}^B \mapsto \zeta(\mathsf{x}) \cup \{y^B\}])\}.$$

- $\mathbb{v}_{\mathsf{x}^{\mathsf{B}}}$ matches a concrete instantiation $y^B$ of $\mathsf{x}$, according to the concretisation function $\zeta$. Thus

$$[\![\mathbb{v}_{\mathsf{x}^{\mathsf{B}}}]\!]_\rho(\zeta) = \{y\, t_1 \cdots t_n \mid y \in \zeta(\mathsf{x}^B)\}.$$

Hence, for any abstract $\mathsf{x}$, $\zeta(\mathsf{x})$ is a set of concrete variable names that are indistinguishable w.r.t. the property.

## Example: Semi-Infinite Grid $u_\infty$ Revisited

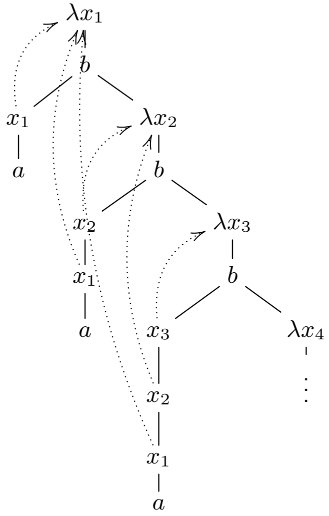
Property $\Phi :=$
"There are only finitely many occurrences of bound variables $x_i$s in each branch."

Let
$\chi :=$
$\mathbb{\lambda}_{\mathsf{x}}.\nu\alpha.\mu\beta.\Big(\big(\mathbb{v}_{\mathsf{b}} \wedge ([1]\beta \vee [2]\mathbb{\lambda}_{\mathsf{x}}.\beta)\big) \vee \big(\mathbb{v}_{\mathsf{x}} \wedge [1]\alpha\big)\Big).$

$\chi$ says "there is a branch with infinitely many occurrences of variables $x_i$s".

Thus $\Phi$ is $\neg\chi$;
and $u_\infty \in [\![\neg\chi]\!]_\emptyset(\zeta)$ where $\zeta$ : a, b $\mapsto \{a\}, \{b\}$.

There exists a $L_\mu^\to$-formula $\psi$ that classifies $u_\infty$, i.e., $[\![\psi]\!]_\emptyset(\zeta) = \{u_\infty\}$



## Set-theoretic Semantics of Higher-Type Mu-Calculus

$\mathsf{BT}^{\Delta \vdash B}$ is the set of Böhm trees of sort $B$ with free variables in $\Delta$.
Define $[\![\varphi^B]\!]_\rho \in \mathcal{P}(\mathsf{BT}^{\Sigma_{\mathrm{var}} \vdash B})$.

$$\begin{aligned}
[\![\mathbb{v}_{\mathsf{x}}]\!]_\rho &:= \{t \in \mathsf{BT}^{\Sigma_{\mathrm{var}} \vdash o} \mid t(\varepsilon) = \mathsf{x}\} \\
[\![\alpha]\!]_\rho &:= \rho(\alpha) \\
[\![\mathbb{\lambda}_{\mathsf{x}}.\varphi]\!]_\rho &:= \{\lambda y.t \mid t \in [\![\varphi]\!]_\rho \cdot (\mathsf{x} \uparrow y)\} \\
[\![[i]\varphi]\!]_\rho &:= \{\mathsf{x}\, t_1 \cdots t_n \in \mathsf{BT}^{\Sigma_{\mathrm{var}} \vdash o} \mid t_i \in [\![\varphi]\!]_\rho\} \\
[\![\nu\alpha.\varphi]\!]_\rho &:= \bigcup \{S \in \mathcal{P}(\mathsf{BT}^{\Sigma_{\mathrm{var}} \vdash o}) \mid S \subseteq [\![\varphi]\!]_{\rho[\alpha \mapsto S]}\} \\
[\![\mu\alpha.\varphi]\!]_\rho &:= \bigcap \{S \in \mathcal{P}(\mathsf{BT}^{\Sigma_{\mathrm{var}} \vdash o}) \mid [\![\varphi]\!]_{\rho[\alpha \mapsto S]} \subseteq S\} \\
[\![\neg\varphi^B]\!]_\rho &:= \mathsf{BT}^{\Sigma_{\mathrm{var}} \vdash B} \setminus [\![\varphi^B]\!]_\rho
\end{aligned}$$

where $(\mathsf{x} \uparrow y)\, \zeta := \zeta[\mathsf{x} \mapsto \zeta(\mathsf{x}) \cup \{y\}]$.

We view $X \subseteq \mathsf{BT}^{\Sigma_{\mathrm{var}} \vdash B}$ as a map $\zeta \mapsto X(\zeta) \subseteq \mathsf{BT}^{\mathrm{Im}(\zeta) \vdash B}$. Hence,
$[\![\varphi^B]\!]_\rho(\zeta) \in \mathcal{P}(\mathsf{BT}^{\mathrm{Im}(\zeta) \vdash B})$.

## Properties of Higher-Type Mu-Calculus

Characterisation by an intuitive model checking game $\mathcal{G}_{\mathrm{MC}}(t, \varphi, \zeta)$:
- $t \in [\![\varphi]\!]_{\emptyset}(\zeta)$ if and only if Verifier has a winning strategy for $\mathcal{G}_{\mathrm{MC}}(t, \varphi, \zeta)$.

**Theorem (Effective Equi-expressivity)**

*ADTA and $L_{\mu}^{\rightarrow}$ are equivalent for defining languages of Böhm trees.*
*I.e. There are algorithms*

1. *mapping ADTA $\mathcal{A}$ to $L_{\mu}^{\rightarrow}$-formula $\varphi_{\mathcal{A}}$ satisfying $L(\mathcal{A}) = [\![\varphi_{\mathcal{A}}]\!]_{\emptyset}(\emptyset)$*
2. *mapping $L_{\mu}^{\rightarrow}$-formula $\varphi$ to ADTA $\mathcal{A}_{\varphi}$ satisfying $[\![\varphi]\!]_{\emptyset}(\emptyset) = L(\mathcal{A}_{\varphi})$.*

Proof.

1. Translate ADTA $\mathcal{A}$ to a $L_{\mu}^{\rightarrow}$-formula in vectorial syntax, with length of vectors equals the number of states of $\mathcal{A}$.
2. Take a $L_{\mu}^{\rightarrow}$-formula $\varphi$ in a strongly-guarded canonical form. Define a ADTA whose states are elements of the closure of $\varphi$.

## Conclusions and Further Directions

ADTA, Higher-type Mu-calculus, and Type-checking Games are equi-expressive for high-type Böhm trees.

**Further Directions**

What is finite model property for $L_\mu^\rightarrow$? Our proposal:

Finite Model Property (FMP): For all $\varphi \in L_\mu^\rightarrow$, if $\varphi$ is satisfiable then it is satisfiable by a $\lambda\mathbf{Y}$-definable Böhm tree.

Disjunctive $L_\mu^\rightarrow$ (equivalent to NDTP) enjoys FMP.
Question: Does $L_\mu^\rightarrow$ enjoy FMP?