On two Notions of Higher-Order Model-Checking

Geocal Meeting

N. Kobayashi E. Lozes F. Bruse

November 29th, 2016

Two Higher-Order Extensions of Model-Checking

H. O. Recursion Schemes

higher-order **models**

functional programs verification

model-checking is **complicated**

[Knapik& al, 2001] [Ong, 2006]

[Hague& al, 2008] [Kobayashi& Ong, 2009]

H. O. Fixpoint Logic

higher-order **properties**

rely-guarantee reasonning
non-regular properties

model-checking is **easy**

[Viswanathan& Viswanathan, 2004]

[Axelson,Lange,Somla, 2007] [Lange,Lozes, 2014]

How are they related?

Why the Question Matters

- we don't have a **simple proof** of HORS decidability

  but if we can reduce HORS model-checking to HFL model-checking, we may give a new, simpler proof of the decidability of HORS model-checking.

- we don't have an **efficient model-checker** for HFL

  but if we can reduce HFL model-checking to HORS model-checking, we can use existing HORS model-checkers.

A Simple Answer

Theorem [Ong, 2006] The HORS model-checking problem is k-EXPTIME complete at order k.

Theorem [Axelson,Lange,Somla, 2007] The HFL model-checking problem is k-EXPTIME complete at order k.

$\Rightarrow$ the two problems can be reduced one to each other.

But... encoding a k-EXPTIME Turing machine is not what we are looking for.

The Big Picture

recursion scheme		altern. parity tree autom.
$tree(\mathcal{G})$	is accepted by	$\mathcal{A}$

$\mathcal{S}$ $\models$ φ

lts | | HFL formula

The Big Picture

recursion scheme — altern. parity tree autom.

tree($\mathcal{G}$) is accepted by $\mathcal{A}$

$\mathcal{S}$ $\models$ φ

lts — HFL formula

Recursion Schemes

recursion scheme altern. parity tree autom.

tree($\mathcal{G}$) is accepted by $\mathcal{A}$

$\mathcal{S}$ $\models$ φ

lts HFL formula

Recursion Schemes

terminals (order≤ 1)

$a : \star \to \star \to \star$

$b : \star \to \star$

$c : \star$

non-terminals

$S : \star$

$F : (\star \to \star) \to \star$

$B : (\star \to \star) \to \star \to \star$

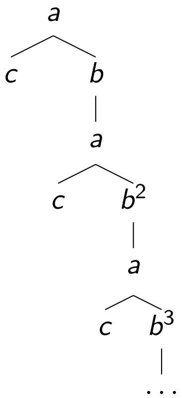
rules

$$\begin{array}{lcl} S & \to & F\ b \\ F\ x & \to & a\ c\ (x\ (F\ (B\ b))) \\ B\ x\ y & \to & b\ (x\ y) \end{array}$$

reductions

$$\begin{array}{ll} S & \to F\ b \\ & \to a\ c\ (b\ (F\ (B\ b))) \\ & \to a\,c\,(b\,(a\,c\,(B\,b\,(F\,(B\,(B\,b)))))) \\ & \to \ldots \end{array}$$

limit tree

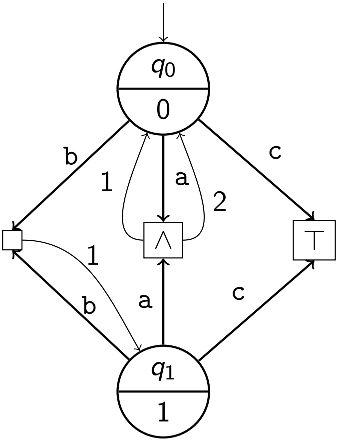


Alternating Parity Tree Automaton

recursion scheme		**altern. parity tree autom.**
$tree(\mathcal{G})$	is accepted by	$\mathcal{A}$

$\mathcal{S}$	$\models$	φ
lts		HFL formula

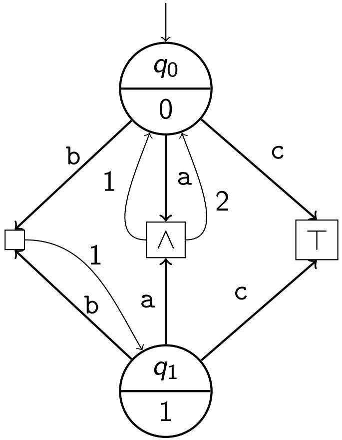
Alternating Parity Tree Automaton



$\mathcal{A} = (Q, \Sigma, \delta, q_0, \Omega)$ with

- $\delta(q, x) \in \text{Bool}^+(Dir(x) \times Q)$ where $Dir(x) = \{1, \dots, \text{arity}(x)\}$
- ex: $\delta(q_0, b) = (1, q_1)$: move to first child and state q_1
- ex: $\delta(q_0, a) = (1, q_0) \wedge (2, q_0)$
- $\Omega : Q \to \{0, \dots, p-1\}$: priority function
- ex: $\Omega(q_0) = 0$, $\Omega(q_1) = 1$

Alternating Parity Tree Automaton

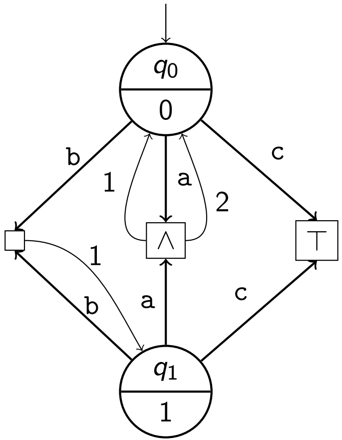


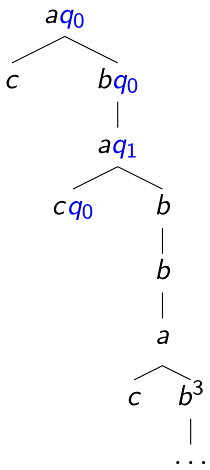
acceptance game on a given tree T

- a play π is a path of T labeled with states
- parity condition: prover wins if
  - either π is finite
  - or $\pi = s_0 s_1 \dots s_i \dots$ with $\limsup_{i\to\infty} \Omega(s_i)$ even

- $T \in L(\mathcal{A})$ if prover has a winning strategy

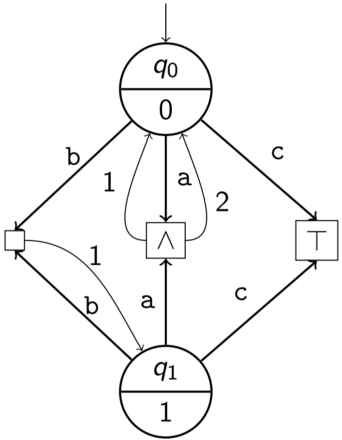
Alternating Parity Tree Automaton

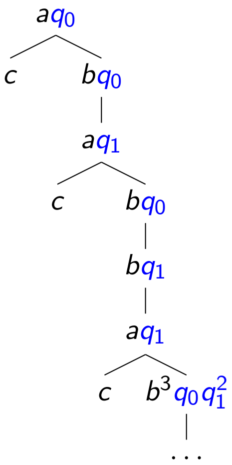




ex: accepting

Alternating Parity Tree Automaton





ex: non-accepting

Higher-Order Fixpoint Logic

recursion scheme		altern. parity tree autom.
$tree(\mathcal{G})$	is accepted by	$\mathcal{A}$

$\mathcal{S}$	$\models$	φ
lts		**HFL formula**

Higher-Order Fixpoint Logic

η	$::=$	$\bullet \mid \eta_1 \to \eta_2$	(simple types)
φ, ψ	$::=$	$\top \mid \bot$	(true,false)
		$\mid \varphi \vee \psi$	(disjunction)
		$\mid \varphi \wedge \psi$	(conjunction)
		$\mid \langle a \rangle \varphi$	(may modality)
		$\mid [a]\varphi$	(must modality)
		$\mid X$	(variable)
		$\mid \mu X^{\eta}.\ \varphi$	(h.o least fixed point)
		$\mid \nu X^{\eta}.\ \varphi$	(h.o greatest fixed point)
		$\mid \lambda X^{\eta}.\ \varphi$	(abstraction)
		$\mid \varphi\ \psi$	(function application)

remark: negation is admissible [Lozes, FICS'2015]

Examples

- predicate transformers

$$\lambda X.\, p \vee \langle a \rangle X \qquad \lambda X.\, \lambda Y.\, X \vee \langle a \rangle Y$$

Examples

- predicate transformers

$$\lambda X.\ p \vee \langle a \rangle X \qquad \lambda X.\ \lambda Y.\ X \vee \langle a \rangle Y$$

- higher-order predicate transformers

$$\lambda F. \lambda X.\ F\ (F\ X)$$

Examples

- predicate transformers

$$\lambda X.\ p \vee \langle a \rangle X \qquad \lambda X.\ \lambda Y.\ X \vee \langle a \rangle Y$$

- higher-order predicate transformers

$$\lambda F.\lambda X.\ F\ (F\ X)$$

- recursive predicate transformers

$$\mu F.\lambda X.\ X \vee \bigvee_{a \in \Sigma} \langle a \rangle (F\ (\langle a \rangle X))$$

Non-Regular Properties

The semantics of

$$\mu F.\lambda X.\ X \vee \bigvee_{a \in \Sigma} \langle a \rangle (F\ (\langle a \rangle X))$$

can be computed by its approximants

$$\begin{aligned}
F^0\ X &= \bot \\
F^1\ X &= X \\
F^2\ X &= X \vee \bigvee_{a \in \Sigma} \langle a \rangle \langle a \rangle X \\
F^3\ X &= F^2\ X \vee \bigvee_{a,b \in \Sigma} \langle a \rangle \langle b \rangle \langle b \rangle \langle a \rangle X \\
\ldots \\
F^\omega\ X &= \bigvee_{\text{palindrome } w} \langle w \rangle X
\end{aligned}$$

Global Model-Checking

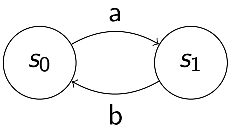
recursion scheme | | altern. parity tree autom.
$tree(\mathcal{G})$ is accepted by $\mathcal{A}$

$\mathcal{S}$ $\models$ φ

lts | | HFL formula

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

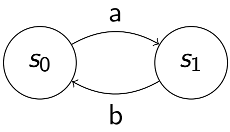


$$\mu F.\lambda X.\ X \wedge [a]F\ \langle b\rangle X$$

X	$F^0\ X$
$\emptyset$	
$\{s_0\}$	
$\{s_1\}$	
$\{s_0, s_1\}$	

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

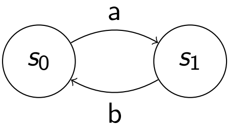


$\mu F.\lambda X.\ X \wedge [a]F\ \langle b \rangle X$

X	$F^0\ X$
$\emptyset$	$\emptyset$
$\{s_0\}$	
$\{s_1\}$	
$\{s_0, s_1\}$	

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

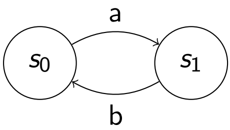


$$\mu F.\lambda X.\ X \wedge [a]F\ \langle b\rangle X$$

X	$F^0\ X$
$\emptyset$	$\emptyset$
$\{s_0\}$	$\emptyset$
$\{s_1\}$	
$\{s_0, s_1\}$	

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

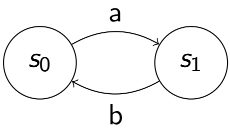


$\mu F.\lambda X.\ X \wedge [a]F\ \langle b \rangle X$

X	$F^0\ X$
$\emptyset$	$\emptyset$
$\{s_0\}$	$\emptyset$
$\{s_1\}$	$\{s_1\}$
$\{s_0, s_1\}$	

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

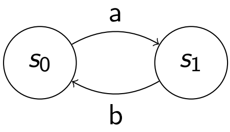


$$\mu F.\lambda X.\ X \wedge [a]F\ \langle b\rangle X$$

X	$F^0\ X$	$F^1\ X$
$\emptyset$	$\emptyset$	
$\{s_0\}$	$\emptyset$	
$\{s_1\}$	$\{s_1\}$	
$\{s_0, s_1\}$	$\{s_1\}$	

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

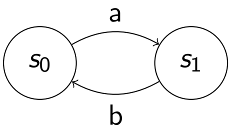


$\mu F.\lambda X.\ X \wedge [a]F\ \langle b\rangle X$

X	$F^0\ X$	$F^1\ X$
$\emptyset$	$\emptyset$	$\emptyset$
$\{s_0\}$	$\emptyset$	
$\{s_1\}$	$\{s_1\}$	
$\{s_0, s_1\}$	$\{s_1\}$	

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

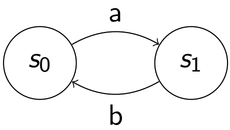


$$\mu F.\lambda X.\ X \wedge [a]F\ \langle b \rangle X$$

X	$F^0\ X$	$F^1\ X$
$\emptyset$	$\emptyset$	$\emptyset$
$\{s_0\}$	$\emptyset$	$\{s_0\}$
$\{s_1\}$	$\{s_1\}$	
$\{s_0, s_1\}$	$\{s_1\}$	

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

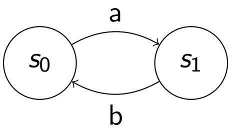


$\mu F.\lambda X.\ X \wedge [a]F\ \langle b \rangle X$

X	$F^0\ X$	$F^1\ X$
$\emptyset$	$\emptyset$	$\emptyset$
$\{s_0\}$	$\emptyset$	$\{s_0\}$
$\{s_1\}$	$\{s_1\}$	$\{s_1\}$
$\{s_0, s_1\}$	$\{s_1\}$	

Global Model-Checking

- represent functions in extension
- compute fixpoints by their approximants

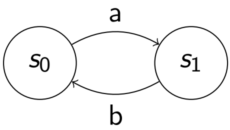


$$\mu F.\lambda X.\ X \wedge [a]F\ \langle b \rangle X$$

X	$F^0\ X$	$F^1\ X$
$\emptyset$	$\emptyset$	$\emptyset$
$\{s_0\}$	$\emptyset$	$\{s_0\}$
$\{s_1\}$	$\{s_1\}$	$\{s_1\}$
$\{s_0, s_1\}$	$\{s_1\}$	$\{s_0, s_1\}$

Global Model-Checking

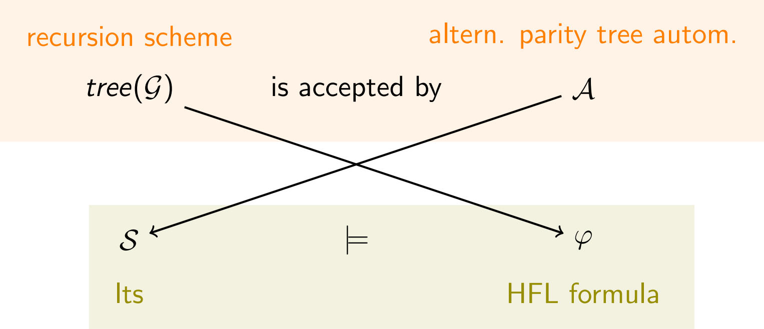
- represent functions in extension
- compute fixpoints by their approximants



$\mu F.\lambda X.\ X \wedge [a]F\ \langle b \rangle X$

X	$F^0\ X$	$F^1\ X$	$F^2\ X$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\{s_0\}$	$\emptyset$	$\{s_0\}$	$\{s_0\}$
$\{s_1\}$	$\{s_1\}$	$\{s_1\}$	$\{s_1\}$
$\{s_0, s_1\}$	$\{s_1\}$	$\{s_0, s_1\}$	$\{s_0, s_1\}$

From HORS Model-Checking to HFL Model-Checking



From HORS Model-Checking to HFL Model-Checking

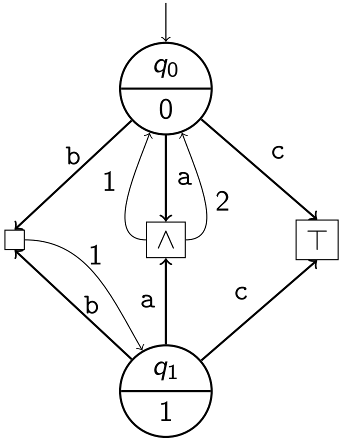
recursion scheme | altern. parity tree autom.

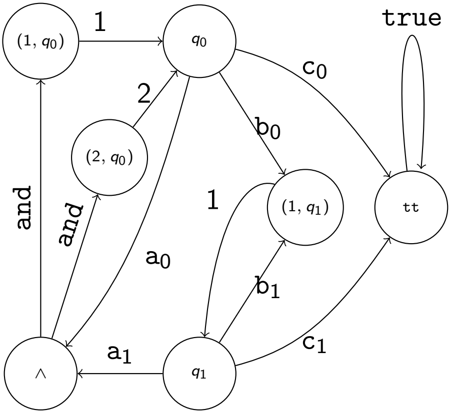
$tree(\mathcal{G})$ is accepted by $\mathcal{A}$

$\mathcal{S}$ $\models$ φ

lts | HFL formula

Encoding an automaton as a LTS





Recursion Schemes as HFL Formulas

recursion scheme — altern. parity tree autom.

tree($\mathcal{G}$) is accepted by $\mathcal{A}$

$\mathcal{S}$ $\models$ φ

lts — HFL formula

Notation

The sequence $\mathcal{E} := X_1^{\eta_1} =_{\alpha_1} \varphi_1; \ldots; X_n^{\eta_n} =_{\alpha_n} \varphi_n$ stands for the formula $\mathit{toHFL}(\mathcal{E})$ defined as

$$\begin{aligned} \mathit{toHFL}(X^\eta =_\alpha \varphi) &= \alpha X^\eta.\varphi \\ \mathit{toHFL}(\mathcal{E}; X^\eta =_\alpha \varphi) &= \mathit{toHFL}([\alpha X^\eta.\varphi/X]\mathcal{E}). \end{aligned}$$

example:

$$\begin{array}{ll} A & =_\mu \langle a\rangle(B\ A); \\ B & =_\nu \lambda X.\ A \vee \langle b\rangle X \end{array} \quad \text{stands for} \quad \mu A.\ \langle a\rangle\big((\nu B.\lambda X.\ A \vee \langle b\rangle X)\ A\big).$$

Note: in general, the order of the equations matters.

From HORS to HFL

naive idea

- for every rule

$$F\ x_1 \ldots x_n \to t$$

  introduce an equation

$$F =_\nu \lambda x_1 \ldots x_n.\ (t)^\dagger$$

  .

- the formula $(t)^\dagger$ mimicks the term t
  - a non-terminal F becomes a recursive variable
  - a parameter x becomes a λ-bound variable
  - a terminal a becomes a formula that forces to move along the transition of the LTS that encodes the transitions $\delta(-, a)$ of the automaton.

Example

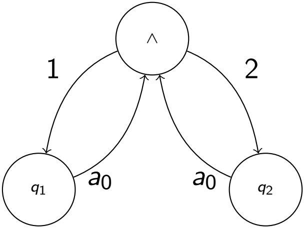
assume a is of arity 2

$$\begin{aligned} S &\to F\ a \\ F\ x &\to x\ (F\ x)\ (F\ x) \end{aligned}$$

with

$$\begin{aligned} \delta(q_i, a) &= (1, q_1) \wedge (2, q_2) \\ \Omega(q_i) &= 0 \end{aligned}$$

becomes



with

$$\begin{aligned} S &=_\nu F\ (\lambda x.\lambda y.\ \langle a_0\rangle(\langle 1\rangle x \wedge \langle 2\rangle y)) \\ F &=_\nu \lambda x.\ x\ (F\ x)\ (F\ x) \end{aligned}$$

Trivial Automata

An alternating parity tree automaton is *trivial* if $\Omega(q) = 0$ for all states q.

Theorem

Let $\mathcal{E}(\mathcal{G})$ be the HES obtained by the naive translation of the HORS $\mathcal{G}$.
Let $\mathcal{A}$ be a trivial APTA and let $\mathcal{S}(\mathcal{A})$ be its associated LTS.
Then

$$tree(\mathcal{G}) \in L(\mathcal{A}) \quad \text{iff} \quad \mathcal{S}(\mathcal{A}) \models \mathcal{E}(\mathcal{G})$$

Issues:

- how to deal with non-trivial automata?
- how to prove this theorem *simply*?

Main Technical Tool: HFL Typing Games

similar to Kobayashi-Ong typing games [Kobayashi,Ong, 2009] but a bit simpler

- no priorities in the intersection types
- simpler parity condition: the outermost recursive variable that is unfolded infinitely often determines the winner

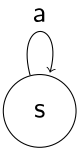
$$\tau ::= s \mid \tau_1 \wedge \cdots \wedge \tau_n \to \tau'$$

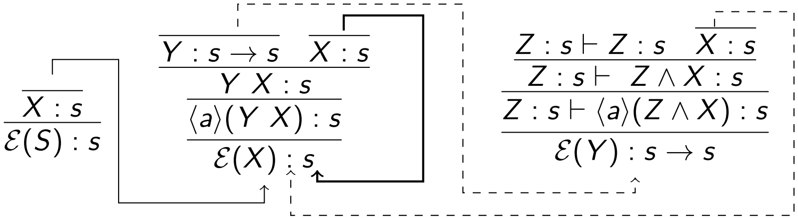
- the type s refines the type $\bullet$ of formulas that denote predicates $\vdash \varphi : s$ if $\varphi : \bullet$ and $s \models \varphi$
- $\vdash \varphi : \tau_1 \wedge \cdots \wedge \tau_n \to \tau'$ if for all ψ such that $\vdash \psi : \tau_i$ for all $i = 1, \ldots, n$, it holds that $\vdash \varphi\ \psi : \tau'$

Example

$$S =_\mu X;$$
$$Y =_\nu \lambda Z.\langle a\rangle(Z \wedge X);$$
$$X =_\mu \langle a\rangle(Y\ X).$$

with





Ingredients of the Proof

Theorem

HFL typing games capture HFL semantics: $\vdash \varphi : s$ is derivable (i.e. Prover has a winning strategy in the typing game) if and only if $s \models \varphi$.

Theorem

The translation $(.)^\dagger$ preserves typability: for trivial automata $\mathcal{A}$, $\vdash (t)^\dagger : q$ in the HFL typing game iff $\vdash t : q$ in the KO typing game for trivial automata.

A Taste of the Case of Non-Trivial Automata

Same idea, but in order to account for priorities

- non-terminals get duplicated
- arguments get duplicated

example

$$S \to F\ b$$
$$F\ x \to x\ (F\ x)$$

becomes

$$S^{\sharp 1} =_\mu F^{\sharp 1}\ b^{\sharp 1}\ b^{\sharp 1};$$

$$F^{\sharp 1} =_\mu \lambda X^{\sharp 1}.\lambda X^{\sharp 0}.\quad X^{\sharp 1}\quad (F^{\sharp 1}\ X^{\sharp 1}\ X^{\sharp 1})\quad (F^{\sharp 0}\ X^{\sharp 1}\ X^{\sharp 0});$$

$$S^{\sharp 0} =_\nu F^{\sharp 0}\ b^{\sharp 1}\ b^{\sharp 0};$$

$$F^{\sharp 0} =_\nu \lambda X^{\sharp 1}.\lambda X^{\sharp 0}.\quad X^{\sharp 0}\quad (F^{\sharp 1}\ X^{\sharp 1}\ X^{\sharp 1})\quad (F^{\sharp 0}\ X^{\sharp 1}\ X^{\sharp 0})$$

A Taste of the Case of Non-Trivial Automata (2)

Why argument duplication is needed can be illustrated at the level of types. Remember KO types [Kobayashi,Ong,2009] are

$$\theta ::= q \mid (\theta_1, m_1) \wedge \cdots \wedge (\theta_n, m_n) \to \theta$$

where m_i are priorities.

The translation relies on

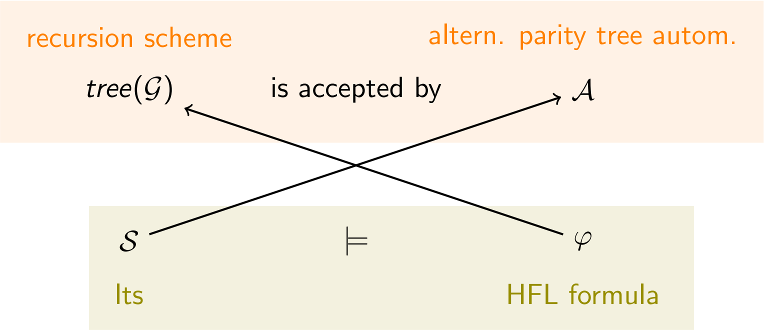
- KO type q being mapped to HFL type q
- KO type

  $$\bigwedge_{j \in J_0} (\theta_j, 0) \wedge \cdots \wedge \bigwedge_{j \in J_p} (\theta_j, p) \to \theta$$

  being mapped to

  $$\bigwedge_{j \in J_0} \theta_j \to \cdots \to \bigwedge_{j \in J_p} \theta_j \to \theta$$

From HFL Model-Checking to HORS Model-Checking



Main Ideas

- on LTS with n states, a HFL formula φ of order k is equivalent to a non-recursive formula $\varphi^{(\alpha)}$ obtained by $\alpha = 2^n_k$ unfoldings
- we create a HORS that generates the syntax tree of $\varphi^{(\alpha)}$
- the APTA evaluates the syntax tree of the formula over the LTS.

- challenge: generate $\varphi^{(\alpha)}$ at order k:
- we used Jones encoding of large numbers [Jones, JFP 2001]

Conclusion

no free lunch today

- new proof of HORS MC decidability, but not really simpler (unless perhaps for trivial automata)
- not clear that HORS model-checkers can be used for HFL model-checking, because of our use of large numbers encoding
- not clear that we cannot do better for HFL→HORS

but

- interesting type system for HFL
- answers the question of *local* model-checking in HFL
- possibly more intuitive than original KO types

Related Work?

- Florian Bruse. Alternating Parity Krivine Automata. MFCS 2014.
- Sylvain Salvati, Igor Walukiewicz. A model for behavioural properties of higher-order programs. CSL 2015.
- Charles Grellois, Paul-André Melliès. An Infinitary Model of Linear Logic. FOSSACS 2015.