On Higher-Order Program Verification and Two Notions of Higher-Order Model Checking

Naoki Kobayashi
University of Tokyo

Summaries of papers from POPL09, POPL17 (joint work with Etienne Lozes, Florian Bruse), and more recent work (joint work with Takeshi Tsukada, and Keiichi Watanabe)

Two Notions of Higher-Order Model Checking

	Models	Logic
finite state model checking	finite state systems	modal μ-calculus

Two Notions of Higher-Order Model Checking

	Models	Logic
finite state model checking	finite state systems	modal μ-calculus
HORS model checking [Knapik+ 01; Ong 06]	higher-order recursion schemes (HORS)	modal μ-calculus

Useful for modeling a certain class of
infinite state systems
(such as higher-order functional programs)

Two Notions of Higher-Order Model Checking

	Models	Logic
finite state model checking	finite state systems	modal μ-calculus
HORS model checking [Knapik+ 01; Ong 06]	higher-order recursion schemes (HORS)	modal μ-calculus
HFL model checking [Viswanathan& Viswanathan 04]	finite state systems	higher-order modal fixpoint logic (HFL)

Useful for describing non-regular properties

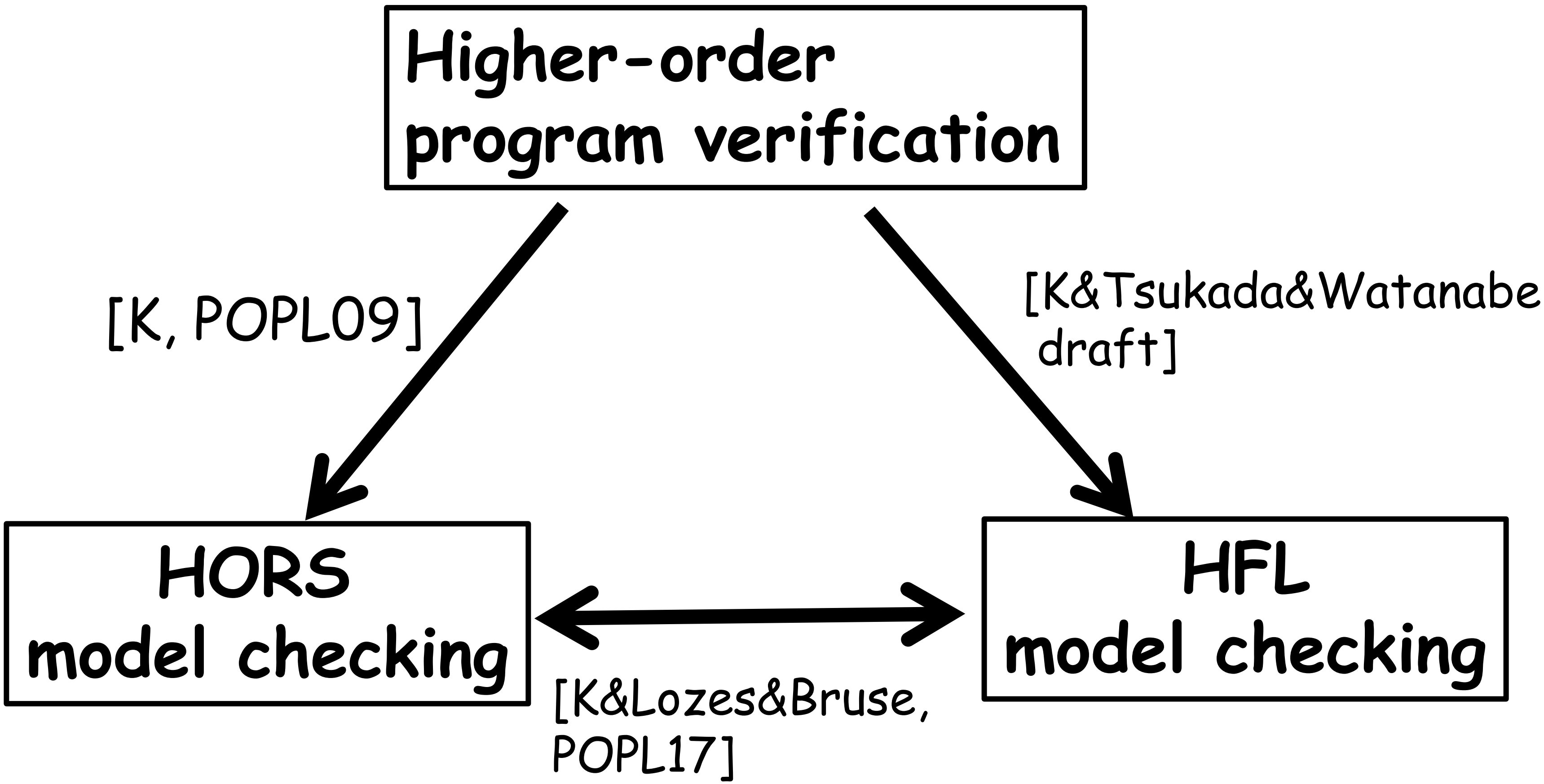
Two Notions of Higher-Order Model Checking

		Logic
finite state model checki		modal μ-calculus
HORS model checking [Knapik+ 01; Ong 06]	higher-order recursion schemes (HORS)	modal μ-calculus
HFL model checking [Viswanathan& Viswanathar	finite state systems	higher-order modal fixpoint logic (HFL)

Applied to verification of higher-order programs [K09][K+11]...

verification of concurrent systems [VV 04][Lange+ 14]

This Talk


Higher-order program verification
[K, POPL09]
[K&Tsukada&Watanabe, draft]
HORS model checking
HFL model checking
[K&Lozes&Bruse, POPL17]

Outline

- Reviews of HORS model checking and HFL model checking
  - HORS model checking
  - HFL model checking
- From program verification to HORS model checking
- Conversion between HORS/HFL model checking
- From program verification to HFL model checking
- Conclusion

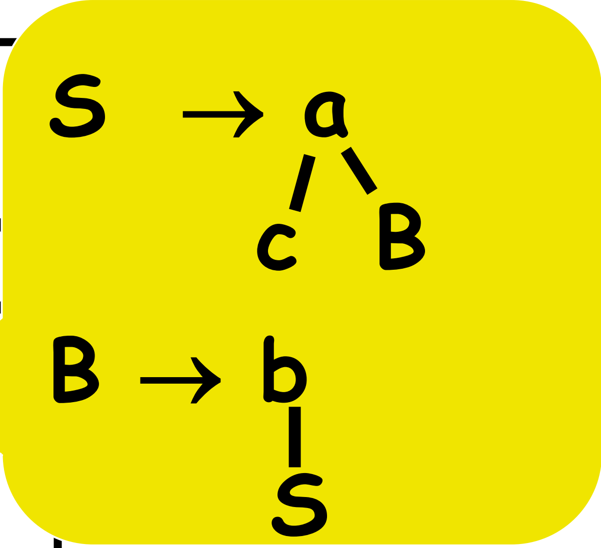
Higher-Order Recursion Scheme (HORS)

- Grammar for generating an infinite tree

Order-0 HORS
(regular tree grammar)

$S \rightarrow a\ c\ B$

$B \rightarrow b\ S$

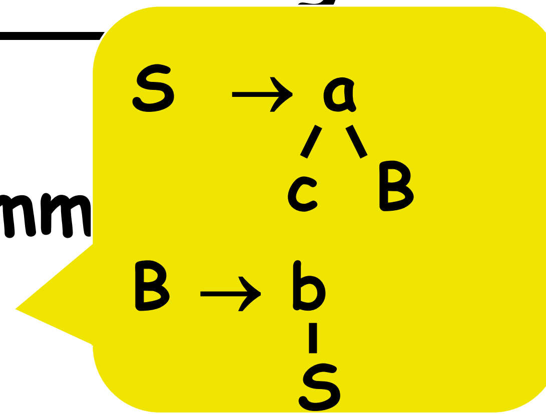


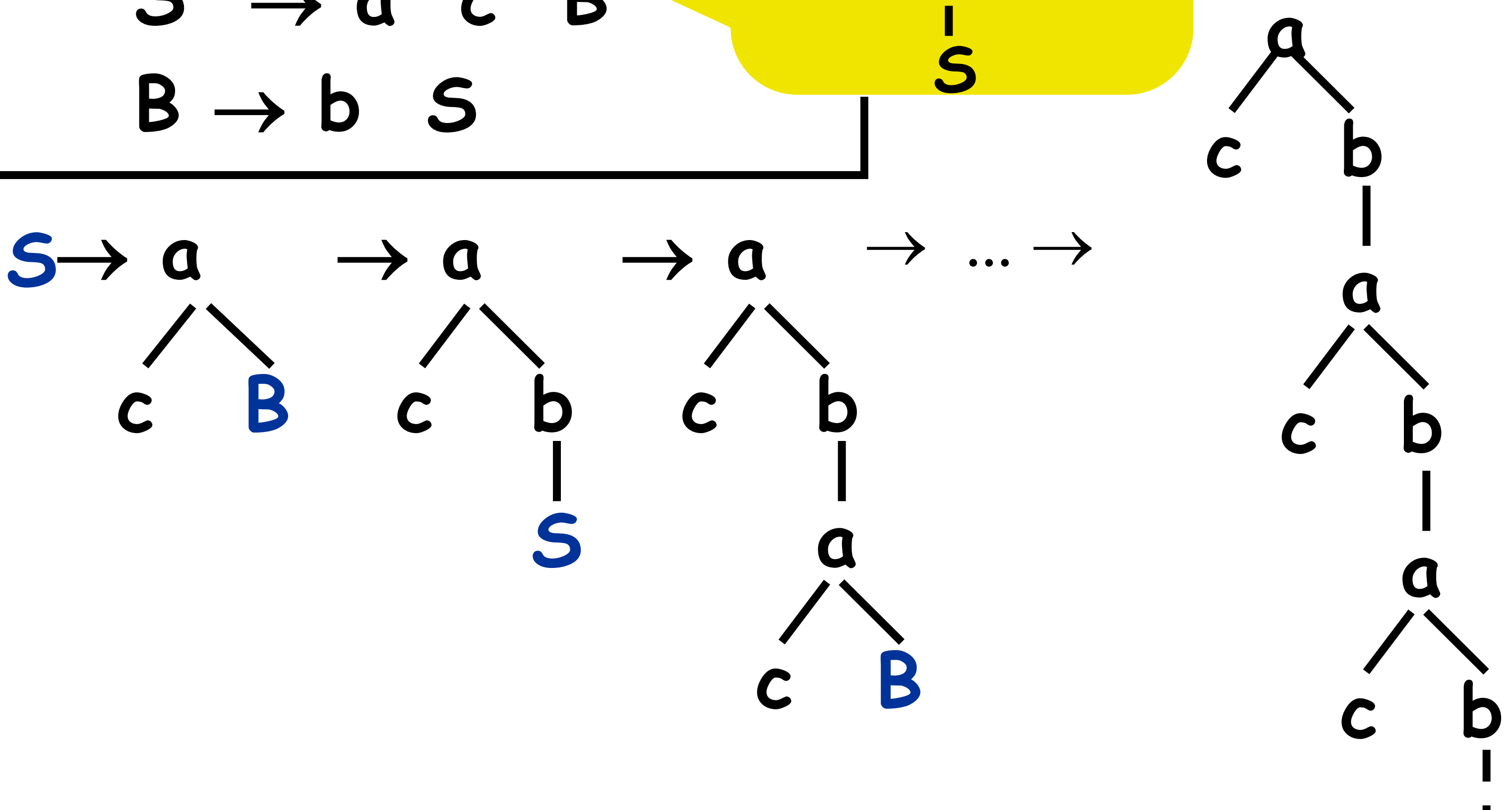
Higher-Order Recursion Scheme (HORS)

♦ Grammar for generating an infinite tree

Order-0 HORS
(regular tree gramm

S → a c B
B → b S





Higher-Order Recursion Scheme (HORS)

- Grammar for generating an infinite tree

Order-1 HORS

$S \rightarrow A\ c$

$A\ x \rightarrow a\ \ x\ \ (A\ (b\ x))$

$S: o,\ A: o \rightarrow o$

Higher-Order Recursion Scheme (HORS)

- Grammar for generating an infinite tree

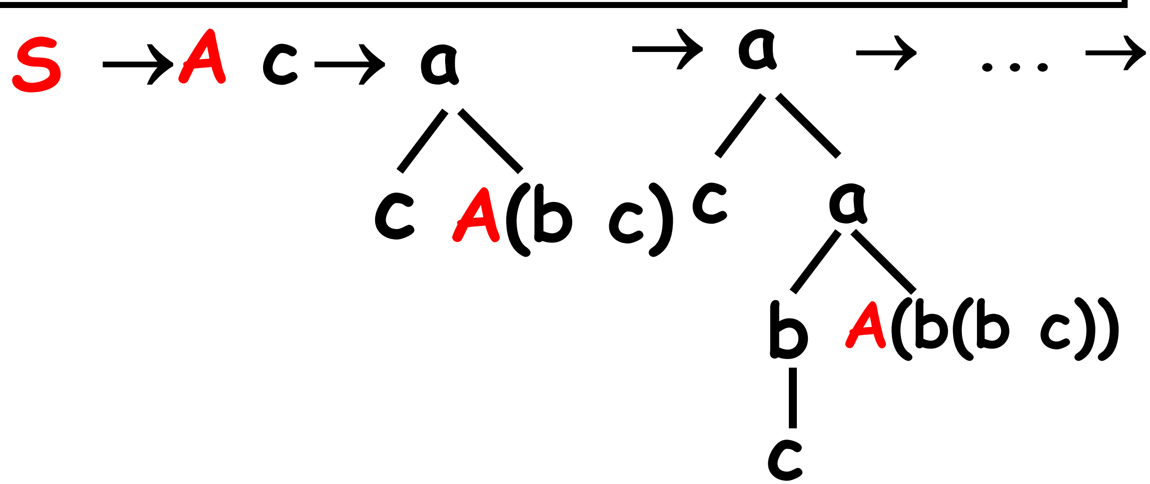
Order-1 HORS

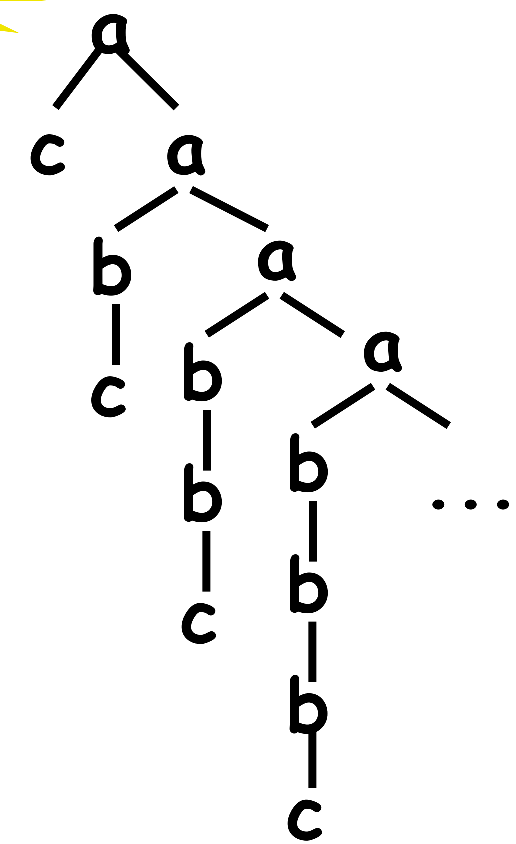
$S \to A\ c$

$A\ x \to a\ x\ (A\ (b\ x))$

$S: o,\ A: o \to o$

Tree whose paths are labeled by $a^{m+1}\ b^m\ c$





Higher-Order Recursion Scheme (HORS)

- Grammar for generating an infinite tree

Order-1 HORS

$S \rightarrow A\ c$

$A\ x \rightarrow a\ \ x\ \ (A\ (b\ x))$

S: o, A: o→ o

HORS

≈

Call-by-name simply-typed λ-calculus

+

recursion, tree constructors

HORS Model Checking

Given
  G: HORS
  A: alternating parity tree automaton (APT)
    (a formula of modal μ-calculus or MSO),
does A accept Tree(G)?

e.g.
- Does every finite path of Tree(G) end with "c"?
- Does "a" occur below "b" in Tree(G)?

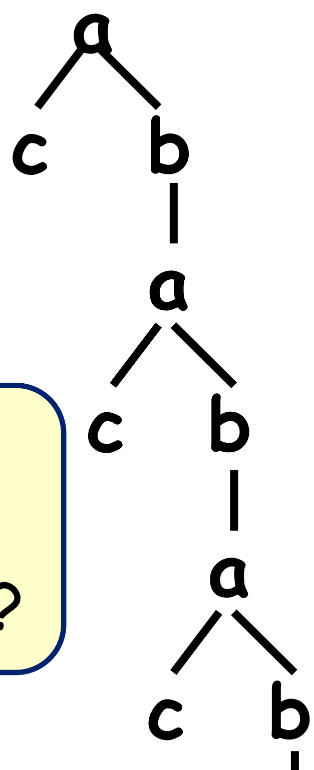
k-EXPTIME-complete [Ong, LICS06]
(for order-k HORS)
but practical algorithms exist

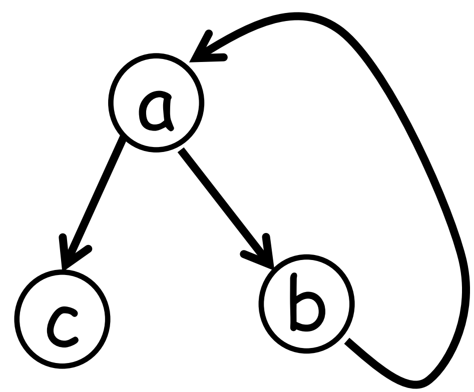
$$k\left\{ 2^{2^{\cdot^{\cdot^{2^{p(x)}}}}} \right.$$

HORS Model Checking as Generalization of Finite State/Pushdown Model Checking

- order-0 ≈ finite state model checking
- order-1 ≈ pushdown model checking

infinite tree ≈ transition system





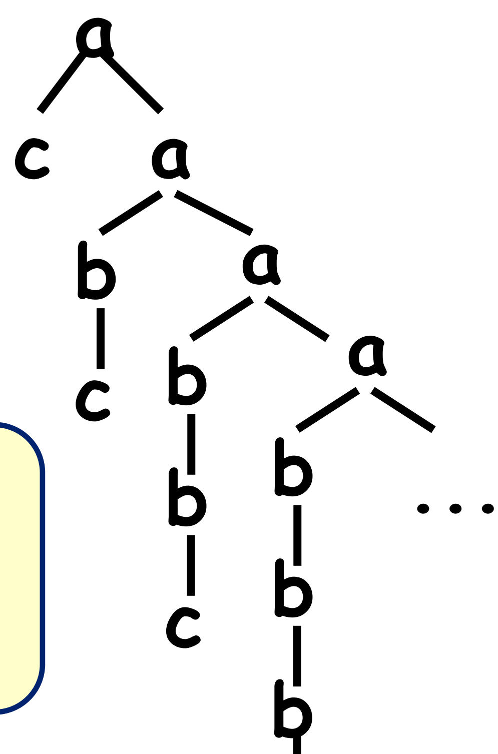
Does "a" occur below "b"?

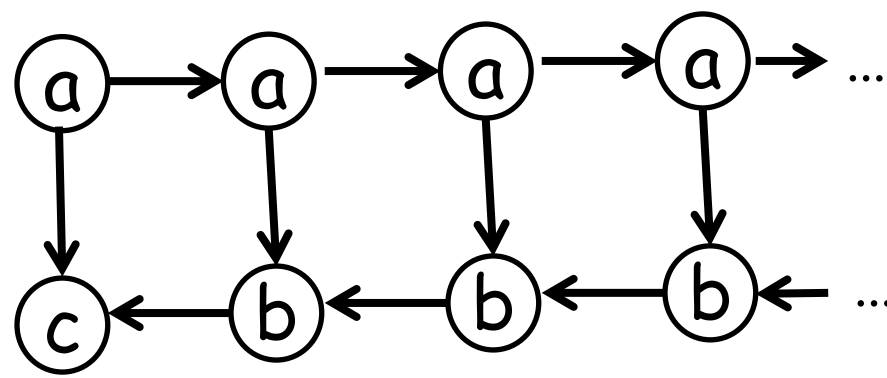
Is there a transition sequence in which "a" occurs after "b"?

HORS Model Checking as Generalization of Finite State/Pushdown Model Checking

- order-0 ≈ finite state model checking
- order-1 ≈ pushdown model checking

infinite tree ≈ (infinite-state) transition system





Does "a" occur below "b"?

Is there a transition sequence in which "a" occurs after "b"?

Outline

- Reviews of HORS model checking and HFL model checking
  - HORS model checking
  - HFL model checking
- From program verification to HORS model checking
- Conversion between HORS/HFL model checking
- From program verification to HFL model checking
- Conclusion

Higher-Order Modal Fixpoint Logic (HFL) [Viswanathan&Viswanathan 04]

- Higher-order extension of the modal μ-calculus

φ ::= true

$\varphi_1 \wedge \varphi_2$	
$\varphi_1 \vee \varphi_2$	
$[a]\varphi$	φ must hold after a
$\langle a\rangle\varphi$	φ may hold after a
X	propositional variable
$\mu X.\varphi$	least fixpoint
$\nu X.\varphi$	greatest fixpoint

Higher-Order Modal Fixpoint Logic (HFL) [Viswanathan&Viswanathan 04]

- Higher-order extension of the modal μ-calculus

φ ::= true	
$\varphi_1 \wedge \varphi_2$	
$\varphi_1 \vee \varphi_2$	
$[a]\varphi$	φ must hold after a
$<a>\varphi$	φ may hold after a
X	predicate variable
$\mu X^{\kappa}.\varphi$	least fixpoint
$\nu X^{\kappa}.\varphi$	greatest fixpoint
$\lambda X^{\kappa}.\,\varphi$	(higher-order) predicate
$\varphi_1\ \varphi_2$	application

$\kappa ::= \bullet \mid \kappa_1 \rightarrow \kappa_2$

Selected Typing Rules for HFL

$$\Gamma \vdash \mathrm{true} : \bullet$$

$$\frac{\Gamma \vdash \varphi : \bullet}{\Gamma \vdash [a]\varphi : \bullet}$$

$$\frac{\Gamma \vdash \varphi : \bullet \qquad \Gamma \vdash \psi : \bullet}{\Gamma \vdash \varphi \wedge \psi : \bullet}$$

$$\frac{\Gamma, X : \kappa_1 \vdash \varphi : \kappa_2}{\Gamma \vdash \lambda X.\varphi : \kappa_1 \to \kappa_2}$$

$$\Gamma, X : \kappa \vdash X : \kappa$$

$$\frac{\Gamma \vdash \varphi : \kappa_1 \to \kappa_2 \qquad \Gamma \vdash \psi : \kappa_1}{\Gamma \vdash \varphi\ \psi : \kappa_2}$$

$$\frac{\Gamma, X : \kappa \vdash \varphi : \kappa}{\Gamma \vdash \mu X.\varphi : \kappa}$$

Semantics

$[\varphi]_I$: the set of states that satisfy φ

$L \models \varphi \Leftrightarrow s_{init} \in [\varphi]_\emptyset$ (s_{init}: initial state of L)

$[\mathrm{true}]_I = \mathrm{States}$ $\quad$ $[\varphi \wedge \psi]_I = [\varphi]_I \cap [\psi]_I$

$[\varphi \vee \psi]_I = [\varphi]_I \cup [\psi]_I$

$[\ [\alpha]\varphi\ ]_I = \{s \mid \forall t.(s \rightarrow_\alpha t \text{ implies } t \in [\varphi]_I)\}$

$[\langle\alpha\rangle \varphi\ ]_I = \{s \mid \exists t.(s \rightarrow_\alpha t \text{ and } t \in [\varphi]_I)\}$

$[\mu X^\kappa.\varphi]_I = \mathrm{lfp}(\lambda x \in [\kappa].[\varphi]_{I\{X=x\}})$

$[\nu X^\kappa.\varphi]_I = \mathrm{gfp}(\lambda x \in [\kappa].[\varphi]_{I\{X=x\}})$

$[\bullet] = 2^{\mathrm{States}}$

$[\kappa_1 \rightarrow \kappa_2] = \{f \in [\kappa_1] \rightarrow [\kappa_2] \mid f\text{: monotonic}\}$

$[\lambda X^\kappa.\varphi]_I = \lambda x \in [\kappa].[\varphi]_{I\{X=x\}}$

$[\varphi\ \psi]_I = [\varphi]_I\ [\psi]_I$ $\quad$ $[X]_I = I(X)$

Example

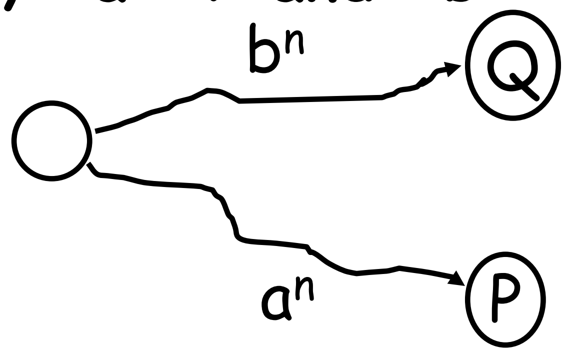
$$\left(\mu F^{\bullet\to\bullet\to\bullet}.\lambda X.\lambda Y.\ (X\wedge Y) \vee F\ (\langle a\rangle X)\ (\langle b\rangle Y)\right)\ P\ Q$$

$$= (P\wedge Q) \vee$$

$$\quad \left(\mu F^{\bullet\to\bullet\to\bullet}.\lambda X.\lambda Y.\ (X\wedge Y) \vee F(\langle a\rangle X)(\langle b\rangle Y)\right)\ (\langle a\rangle P)(\langle b\rangle Q)$$

$$= (P\wedge Q) \vee (\langle a\rangle P\wedge\langle b\rangle Q) \vee (\langle a\rangle\langle a\rangle P\wedge\langle b\rangle\langle b\rangle Q) \vee \ldots$$

For some n, $\langle a\rangle^n P$ and $\langle b\rangle^n Q$ hold

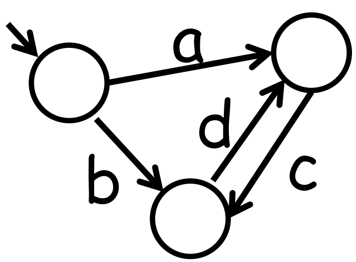


HFL Model Checking

Given
L: (finite-state) labeled transition system
φ: HFL formula,
does L satisfy φ?

e.g. L |= φ for:

L:



φ: $(\mu F.\lambda X.\lambda Y.\ (X \wedge Y)$
$\vee\ F\ (\langle a\rangle X)\ (\langle b\rangle Y))$
$(\langle c\rangle \mathrm{true})\ \ (\langle d\rangle \mathrm{true})$

HES (Hierarchical Equation Systems) Representation of HFL Formulas

$X_1 =_{\alpha 1} \varphi_1; \ \ldots; \ X_n =_{\alpha n} \varphi_n$
$(\alpha_i \in \{\mu, \nu\})$

Example:

HFL: $\nu X.\mu Y.(\langle a\rangle X \vee \langle b\rangle Y)$
(there exists a path $(b^*a)^\omega$)

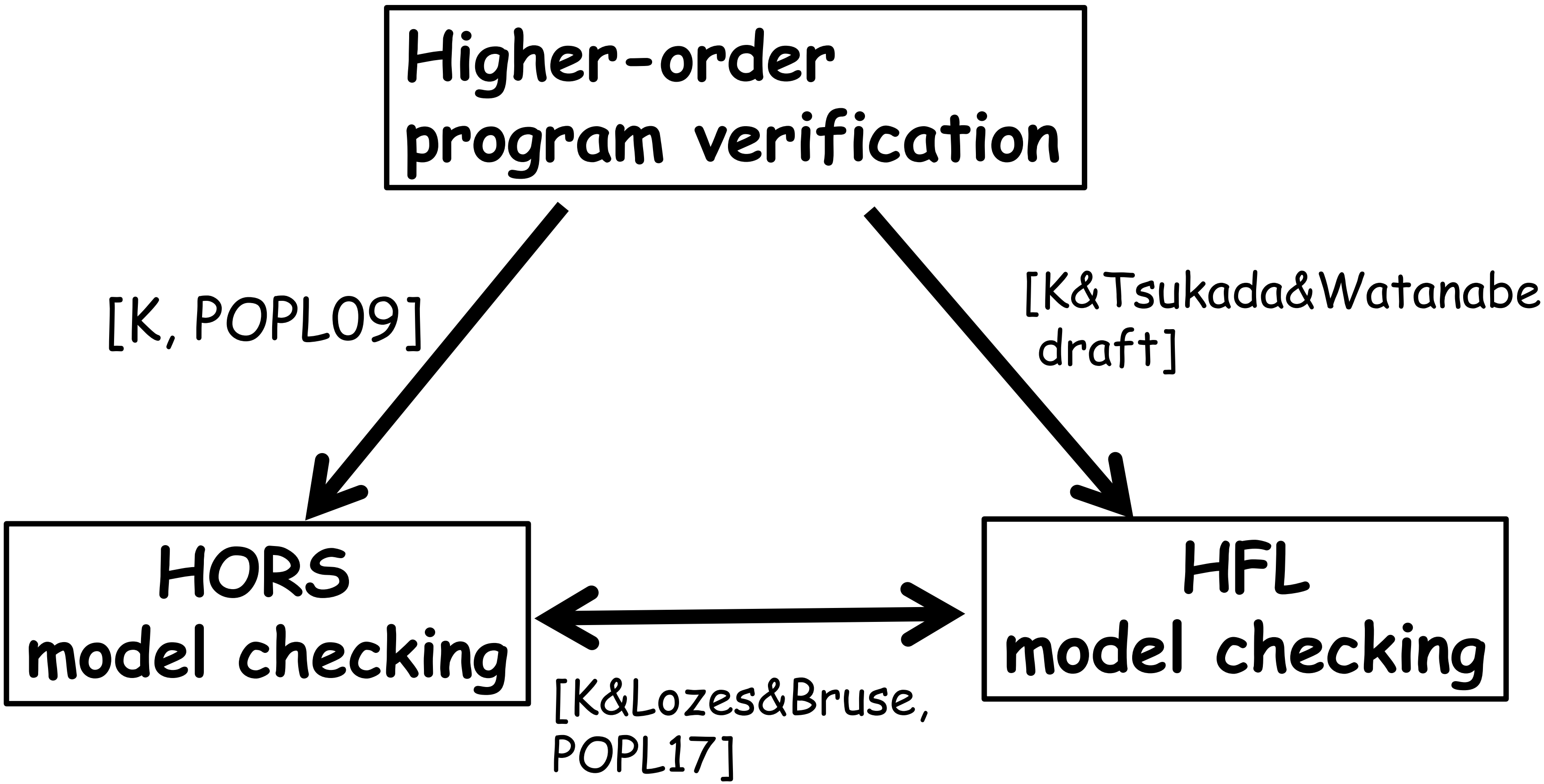
HES: $X =_\nu Y; \ Y =_\mu \langle a\rangle X \vee \langle b\rangle Y$

HORS vs HFL model checking

	Model	Spec.	complexity	Applications
HORS model checking	HORS	APT	k-EXPTIME complete (for order-k HORS)	Automated verification of functional programs [K 09][K+11]...
HFL model checking	LTS	HFL	k-EXPTIME complete (for order-k HFL)	Assume-guarantee reasoning [VV 04] Process equivalence checking [Lange+ 14]

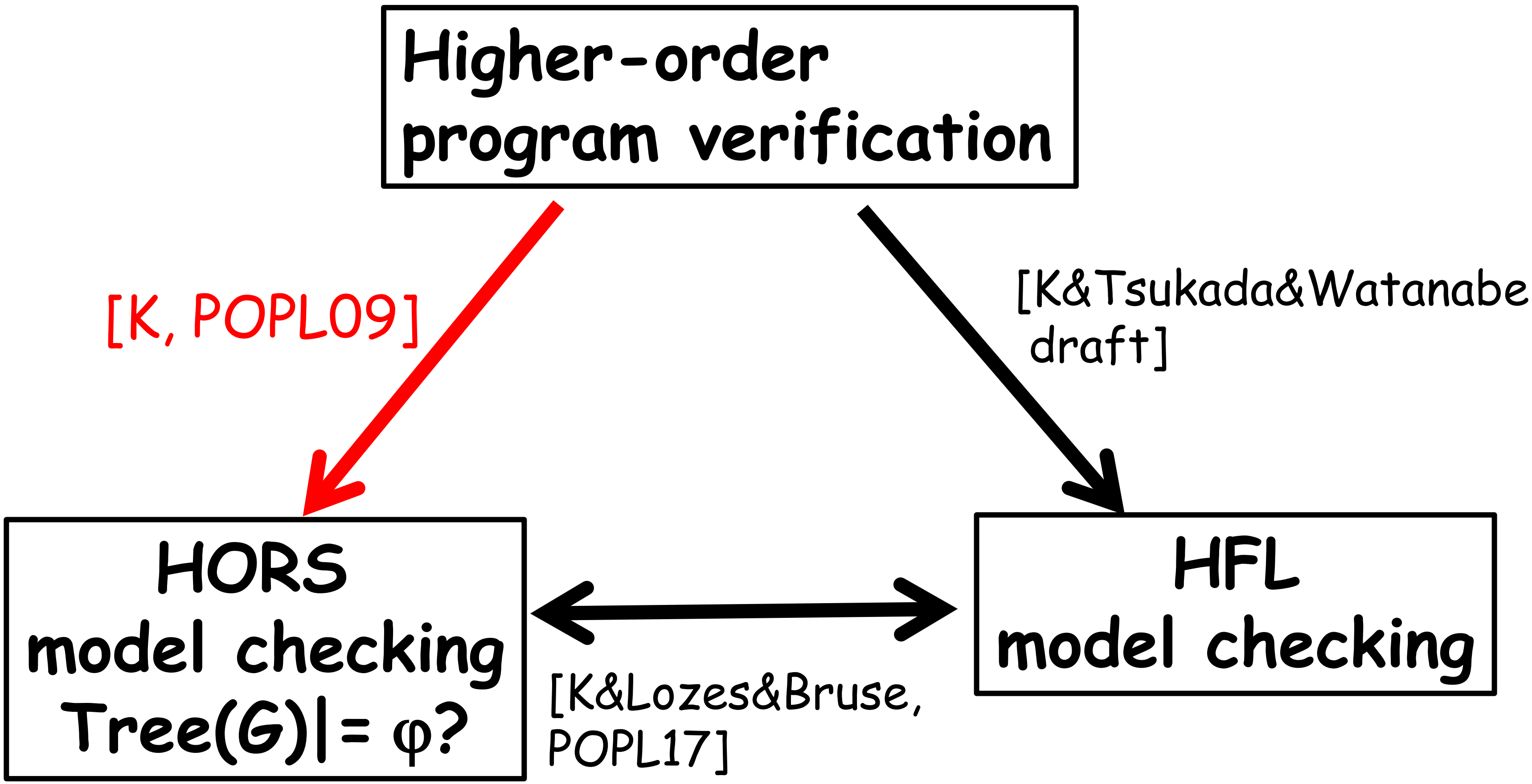
APT: alternating parity tree automaton
LTS: finite-state labeled transition system

This Talk



Higher-order program verification
[K, POPL09]
[K&Tsukada&Watanabe, draft]
HORS model checking
HFL model checking
[K&Lozes&Bruse, POPL17]

This Talk


Higher-order program verification
[K, POPL09]
[K&Tsukada&Watanabe, draft]
HORS model checking Tree(G)|= φ?
HFL model checking
[K&Lozes&Bruse, POPL17]

From Program Verification to HORS Model Checking
[K. POPL 2009]

Higher-order program
+
specification
(on events or output)

→ **Program Transformation** →

HORS G
(describing all event sequences or outputs)
+
Tree property φ,
describing valid event sequences or outputs

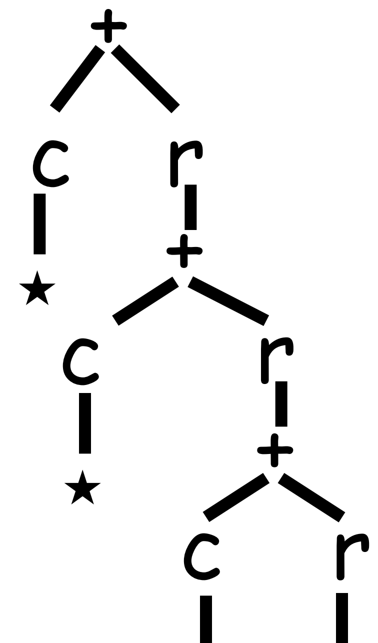
→ **HORS Model Checking**

From Program Verification to Model Checking: Example

```
let f x =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
      f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$



Is the file "foo" accessed according to read* close?

Is each path of the tree labeled by r*c?

From Program g:

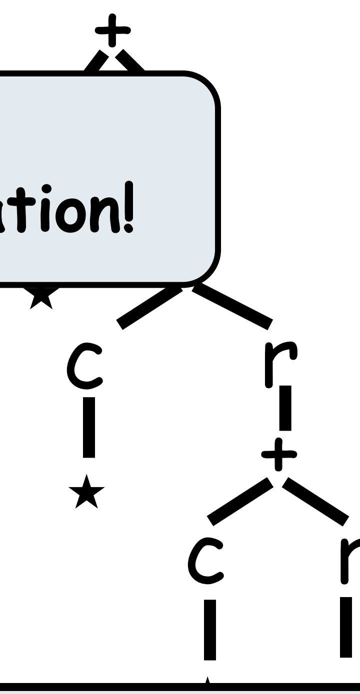
continuation parameter, expressing how "foo" is accessed after the call returns

```
let f x =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
      f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$

CPS Transformation!



Is the file "foo" accessed according to read* close?

Is each path of the tree labeled by r*c?

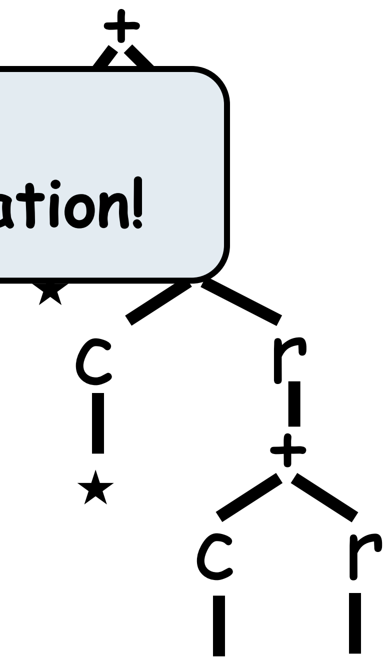
From Program Verification to Model Checking: Example

```
let f x =
  if * then close(x)
  else (read(x); f x)
in
let y = open "foo"
in
      f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$

CPS Transformation!



Is the file "foo" accessed according to read* close?

Is each path of the tree labeled by r*c?

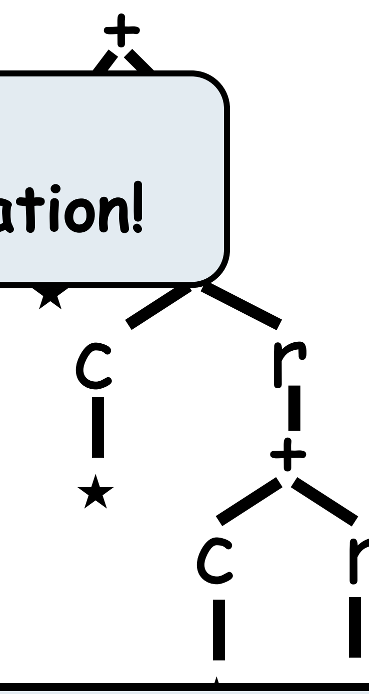
From Program Verification to Model Checking: Example

```
let f x =
  if * then close(x)
  else (read(x); f x)
in
let y = open "foo"
in
     f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$

CPS Transformation!



Is the file "foo" accessed according to read* close?

Is each path of the tree labeled by r*c?

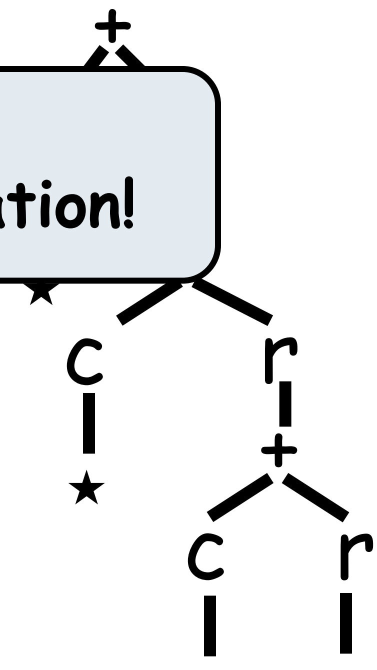
From Program Verification to Model Checking: Example

```
let f x =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
     f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$

CPS Transformation!



Is the file "foo" accessed according to read* close?

Is each path of the tree labeled by r*c?

From Program Verification to Model Checking: Example

```
let f(x) =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
     f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$

S

Is the file "foo" accessed according to read* close?

Is each path of the tree labeled by r*c?

From Program Verification to Model Checking: Example

```
let f(x) =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
     f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$

$F\ d\ \star$

Is the file "foo" accessed according to read* close?

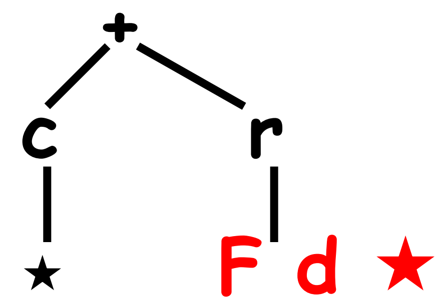
Is each path of the tree labeled by r*c?

From Program Verification to Model Checking: Example

```
let f(x) =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
      f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$



Is the file "foo" accessed according to read* close?

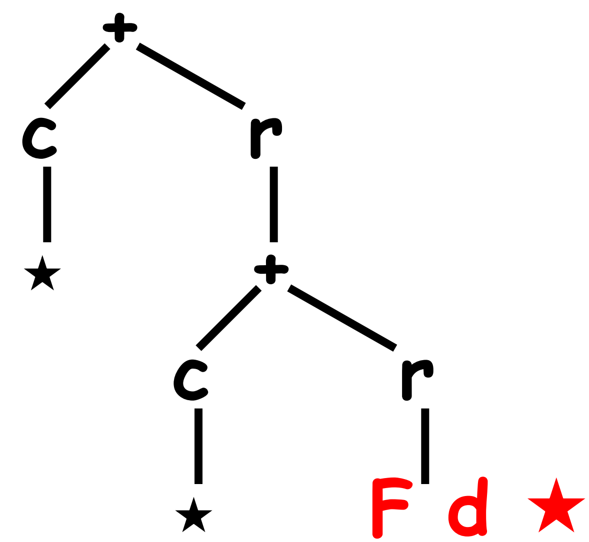
Is each path of the tree labeled by r*c?

From Program Verification to Model Checking: Example

```
let f(x) =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
      f (y)
```

$F\ x\ k \to +\ (c\ k)\ (r(F\ x\ k))$

$S \to F\ d\ \star$



Is the file "foo" accessed according to read* close?

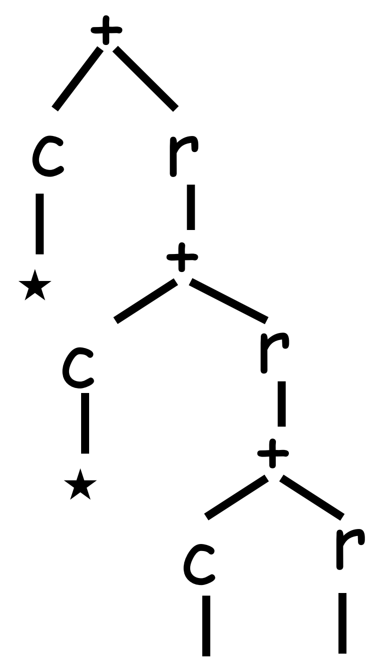
Is each path of the tree labeled by r*c?

From Program Verification to Model Checking:
Example

```
let f(x) =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
     f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$



Is the file "foo" accessed according to read* close?

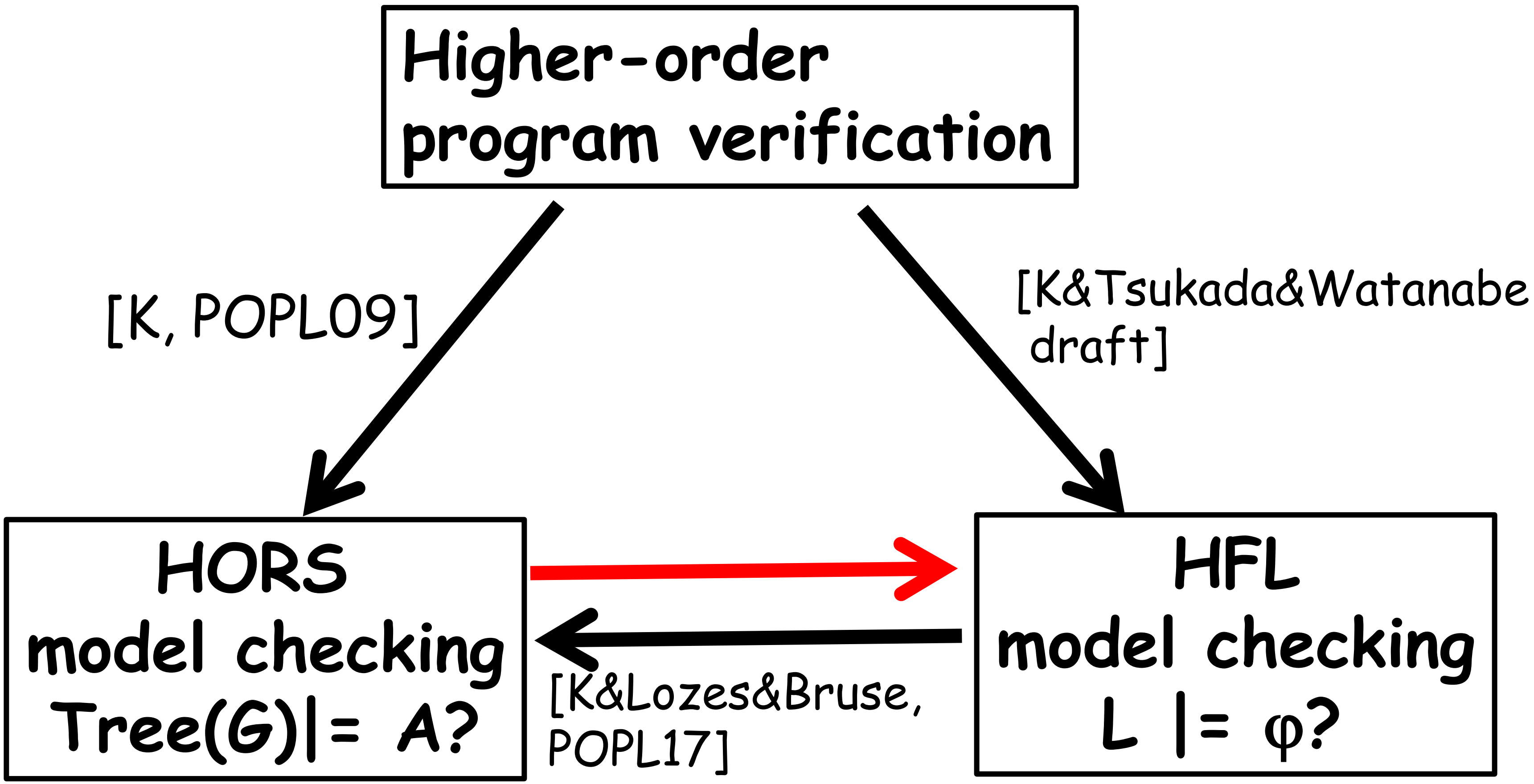
Is each path of the tree labeled by r*c?

Tool demonstration: MoCHi

[K&Sato&Unno, 2011]

(a software model checker for a subset of functional programming language OCaml)

HORS vs HFL model checking



From HORS to HFL model checking

- **Input:**
  - HORS G
  - Parity tree automaton A (with largest priority p)
- **Output:**
  - LTS L_A
  - HFL formula $\varphi_{G,p}$

  such that $\mathrm{Tree}(G) \models A$ iff $L_A \models \varphi_{G,p}$

Intuition:

- L_A simulates the transitions of A
- $\varphi_{G,p}$ describes "L_A has transitions corresponding to an accepting run of A over Tree(G)"

From HORS to HFL model checking

- ♦ Input:
  - HORS G
  - Parity tree automaton A (with largest priority p)
- ♦ Output:
  - LTS L_A
  - HFL formula $\varphi_{G,p}$

  such that $\mathrm{Tree}(G) \models A$ iff $L_A \models \varphi_{G,p}$

Intuition:

- L_A simulates the transitions of A
- $\varphi_{G,p}$ describes "L_A has transitions corresponding to an accepting run of A over Tree(G)"

Construction of L_A

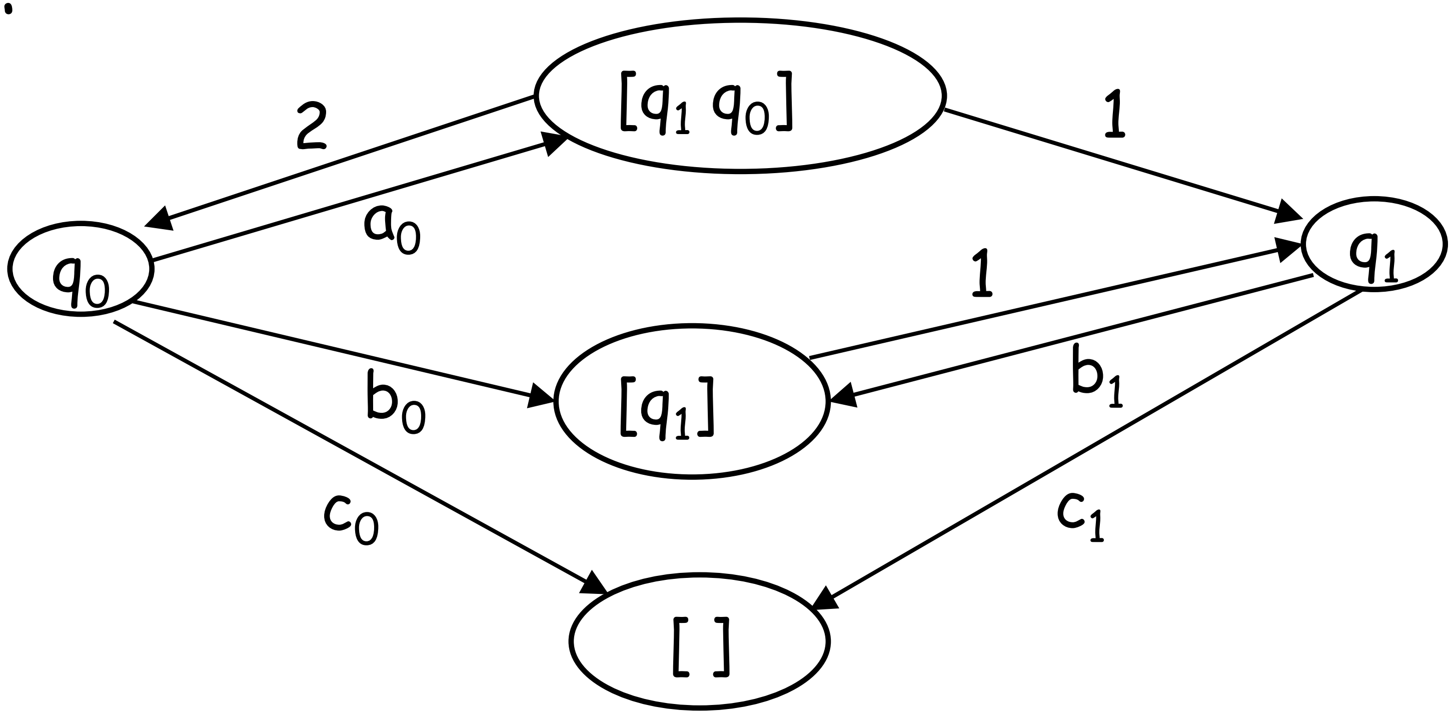
(non-deterministic case; see the paper for the case of APT)

A:

$q_0 \rightarrow_a q_1\, q_0 \quad q_0 \rightarrow_b q_1 \quad q_1 \rightarrow_b q_1 \quad q_0 \rightarrow_c \quad q_1 \rightarrow_c$

$\Omega(q_0)=0 \quad \Omega(q_1)=1$

L_A:



Construction of L_A

(non-deterministic case; see the paper for the case of APT)

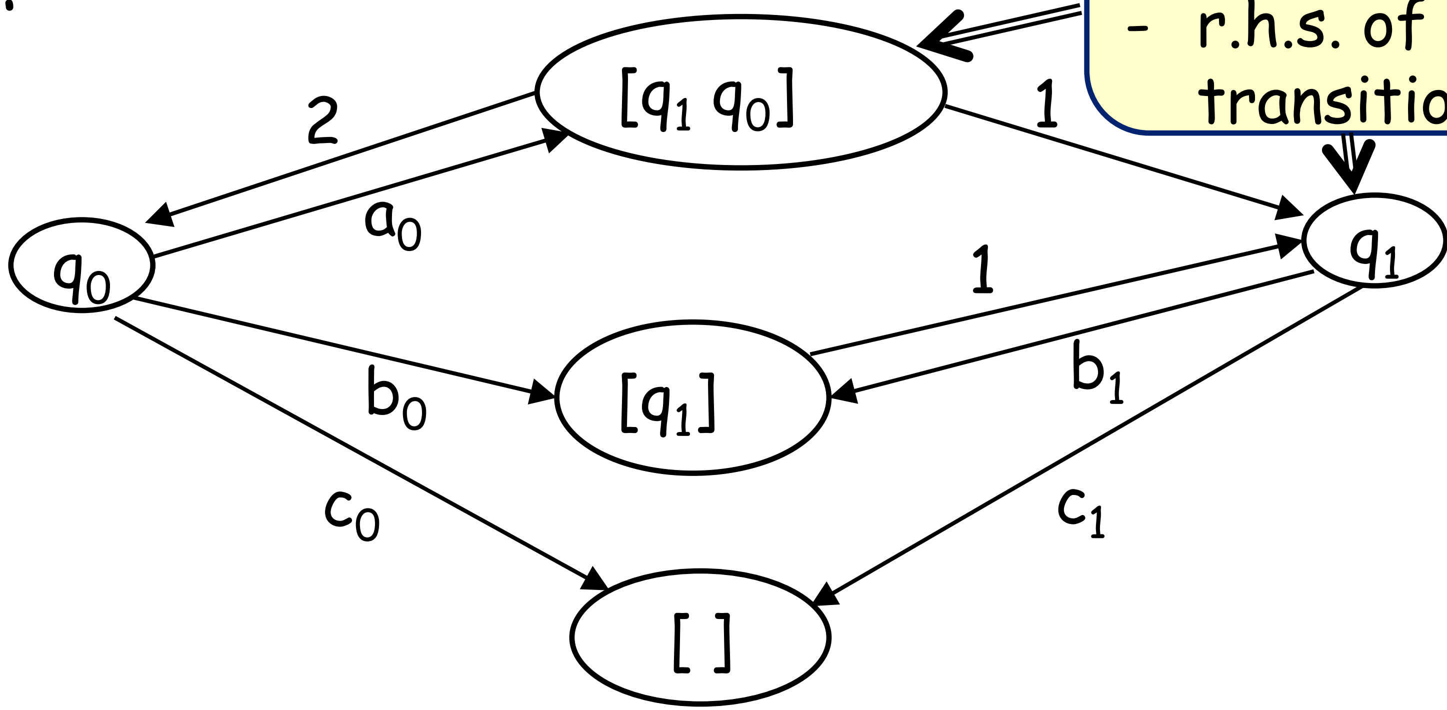
A:

$q_0 \rightarrow_a q_1\, q_0 \quad q_0 \rightarrow_b q_1 \quad q_1 \rightarrow_b q_1 \quad q_0 \rightarrow_c \quad q_1 \rightarrow_c$

$\Omega(q_0)=0 \quad \Omega(q_1)=1$

The states of L_A consist of:
- states of A and
- r.h.s. of transition rules

L_A:



Construction of L_A

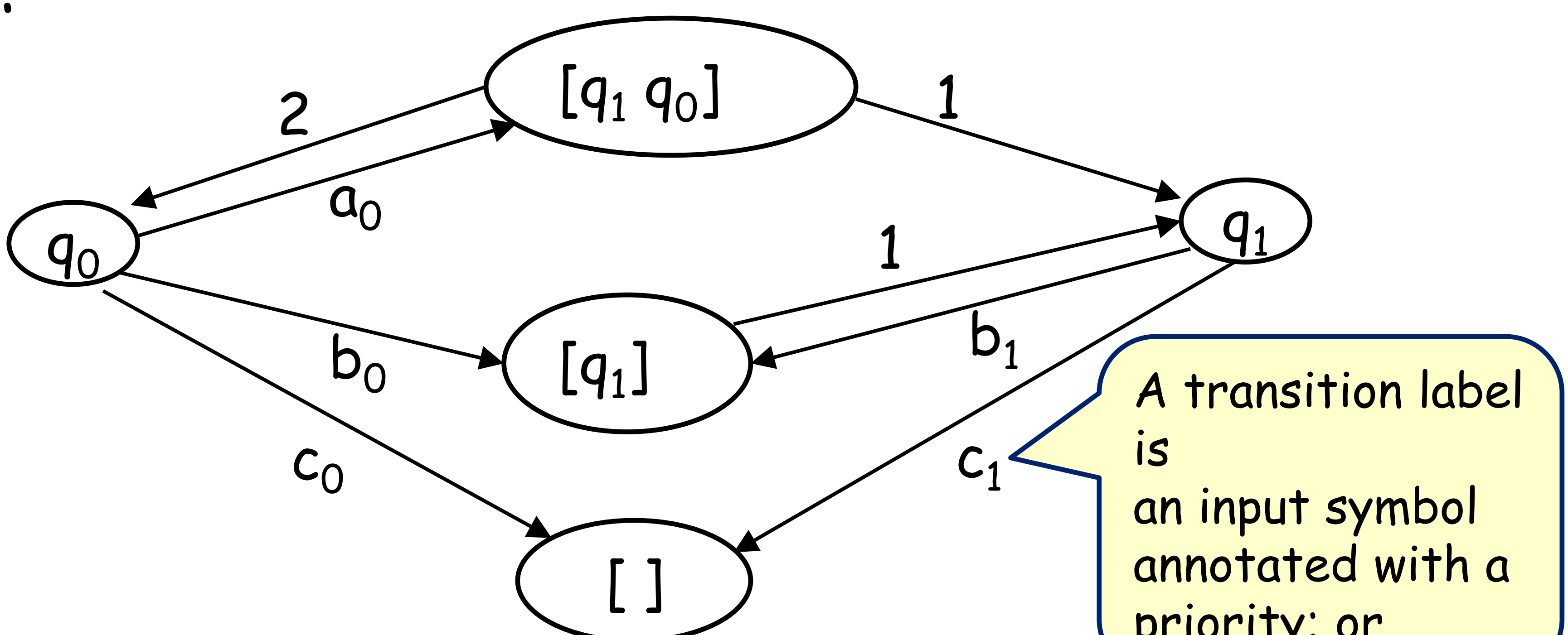
(non-deterministic case; see the paper for the case of APT)

A:

$q_0 \rightarrow_a q_1\, q_0 \qquad q_0 \rightarrow_b q_1 \qquad q_1 \rightarrow_b q_1 \qquad q_0 \rightarrow_c \qquad q_1 \rightarrow_c$

$\Omega(q_0)=0 \qquad \Omega(q_1)=1$

L_A:



Construction of L_A

(non-deterministic case; see the paper for the case of APT)

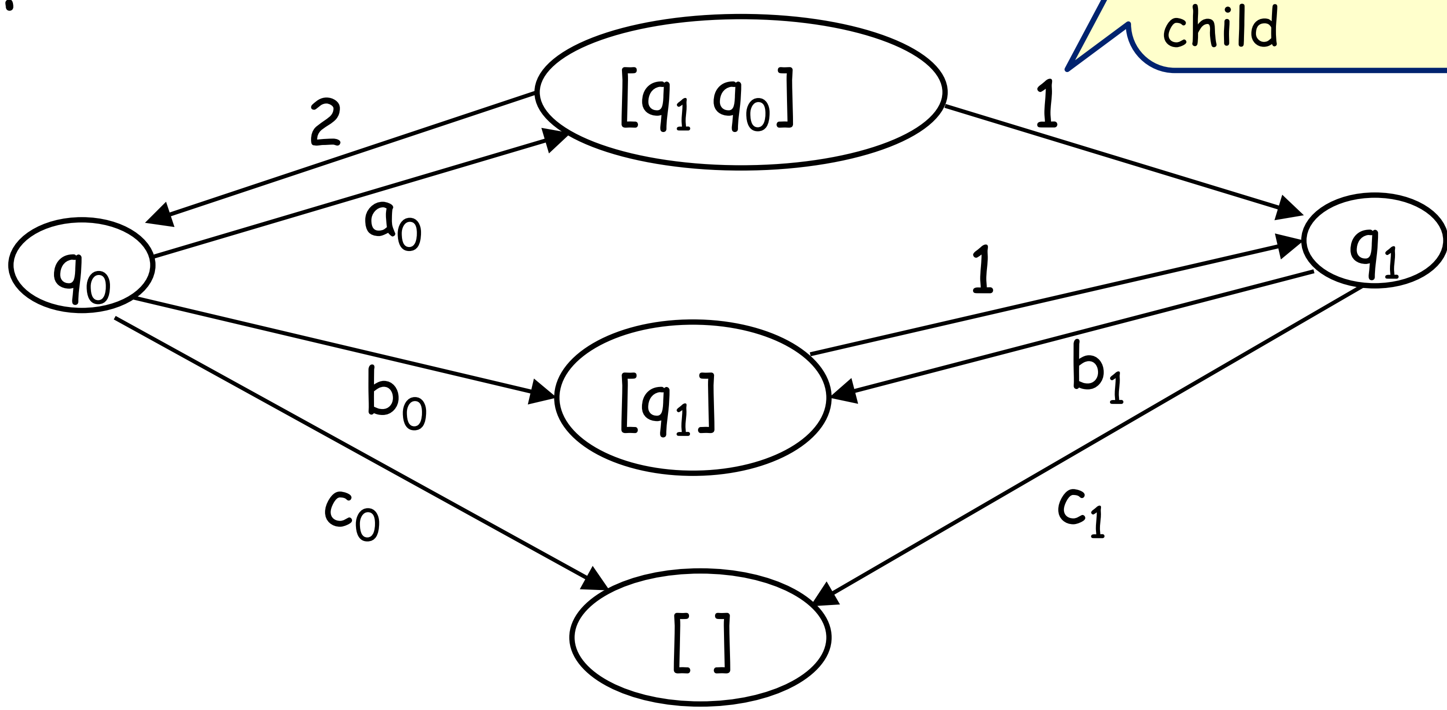
A:

$q_0 \rightarrow_a q_1\, q_0 \quad q_0 \rightarrow_b q_1 \quad q_1 \rightarrow_b q_1 \quad q_0 \rightarrow_c \quad q_1 \rightarrow_c$

$\Omega(q_0)=0 \quad \Omega(q_1)=1$

A transition label is ...; or a number to identify the visited child

L_A:



Outline

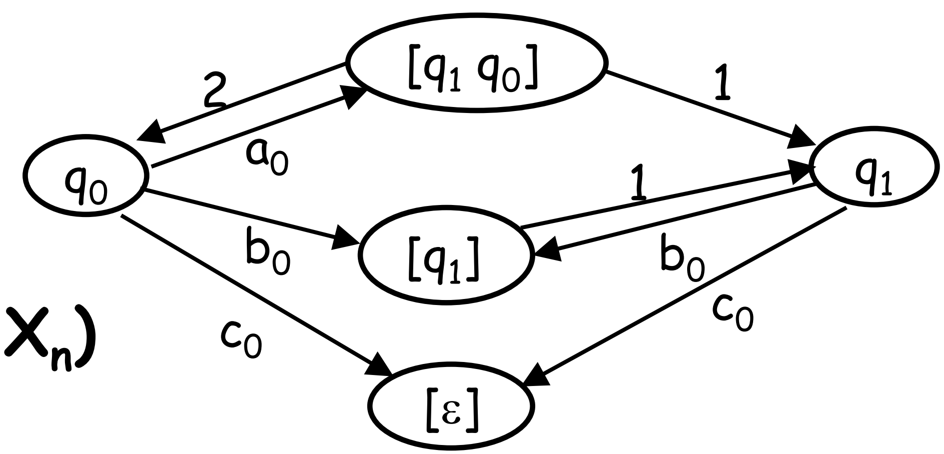
- Reviews of HORS model checking and HFL model checking
- From HORS to HFL model checking
  - construction of L_A
  - construction of $\varphi_{G,p}$
    - case p=0
    - general case
- From HFL to HORS model checking
- Type system for HFL model checking
- Conclusion

From trees to HFL formulas

φ_T: "the current state has transitions corresponding to an accepting run for T"

$\varphi_{a\ c\ (b\ c)}$ =
$\langle a_0 \rangle$ "can visit 1st and 2nd children with states satisfying φ_c and $\varphi_{b\ c}$ respectively"
$= \langle a_0 \rangle(\langle 1 \rangle \varphi_c \wedge \langle 2 \rangle \varphi_{b\ c})$
$= \langle a_0 \rangle(H_2\ \varphi_c\ \varphi_{b\ c})$

$(H_n\ X_1\ \ldots\ X_n \stackrel{\text{def}}{=} \langle 1 \rangle X_1 \wedge \ldots \langle n \rangle X_n)$



From trees to HFL formulas

φ_T: "the current state has transitions corresponding to an accepting run for T"

$\varphi_{a\ c\ (b\ c)}$ =

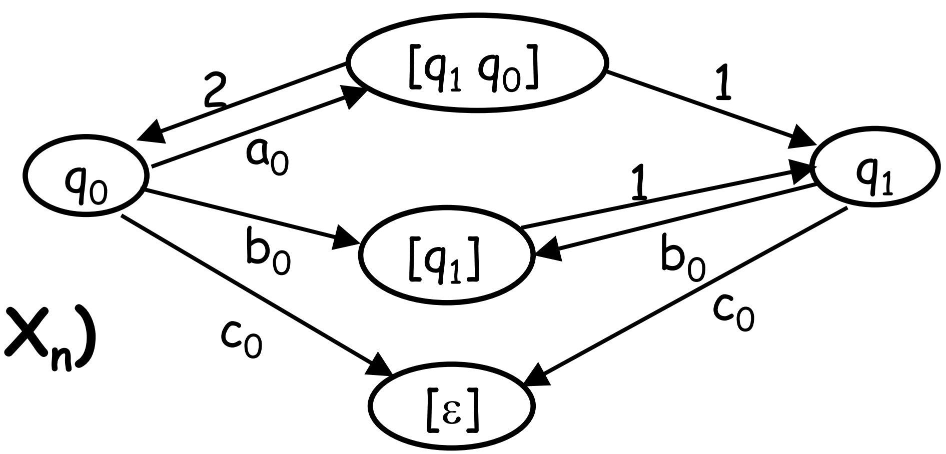
$\langle a_0 \rangle$ "can visit 1st and 2nd children with states satisfying φ_c and $\varphi_{b\ c}$ respectively"

$= \langle a_0 \rangle (\langle 1 \rangle \varphi_c \wedge \langle 2 \rangle \varphi_{b\ c})$

$= \langle a_0 \rangle (H_2\ \varphi_c\ \varphi_{b\ c})$

$= \langle a_0 \rangle (H_2\ (\langle c_0 \rangle H_0)$
$(\langle b_0 \rangle H_1\ (\langle c_0 \rangle H_0)))$

$(H_n\ X_1\ \ldots\ X_n \stackrel{def}{=} \langle 1 \rangle X_1 \wedge \ldots \langle n \rangle X_n)$



From HORS to HFL

HORS G

$S \to F\ c$

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

A:

$q_0 \to_a q_1\, q_0$

$q_0 \to_b q_1 \qquad q_1 \to_b q_1$

$q_0 \to_c \qquad q_1 \to_c$

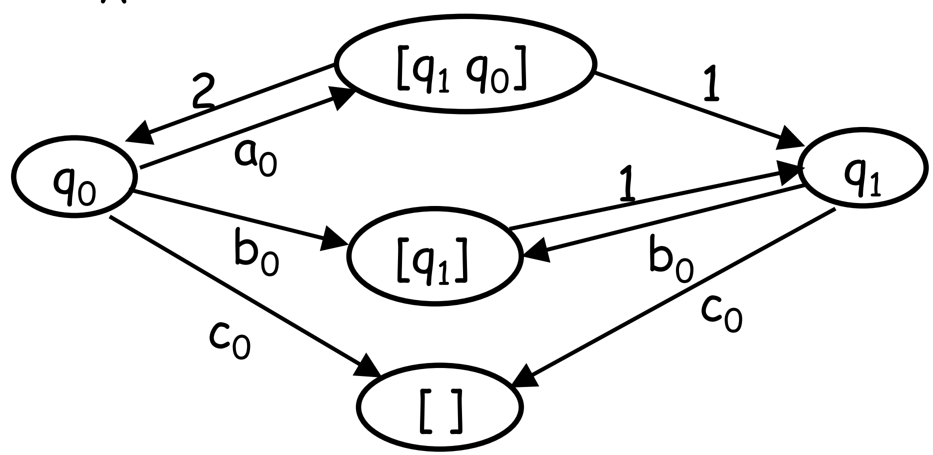
$\varphi_{G,0}$

$S =_\nu F\ (\langle c_0\rangle H_0)$

$F\ x =_\nu$

$\langle a_0\rangle(H_2\ \ x\ \ (F(\langle b_0\rangle(H_1\ \ x)))$

L_A



Outline

- Reviews of HORS model checking and HFL model checking
- From HORS to HFL model checking
  - construction of L_A
  - construction of $\varphi_{G,p}$
    - case p=0
    - general case
- From HFL to HORS model checking
- Type system for HFL model checking
- Conclusion

Difference from the special case

- Replicate each non-terminal/argument for each priority (to translate parity condition to a proper nesting of least/greatest fixpoints)

HORS G: $S \rightarrow F\ c$
$F\ x \rightarrow a\ \ x\ \ (F\ (b\ x))$

$\Downarrow$

HFL $\varphi_{G,1}$

$S^1 =_\mu F^0\ (\langle c_0\rangle H_0)\ (\langle c_1\rangle H_0);$

$F^1\ x^0\ x^1 =_\mu \langle a_0\rangle(H_2\ x^1\ x^1$
$\quad (F^0(\langle b_1\rangle(H_1\ x^1\ x^1)))\ (F^1(\langle b_1\rangle H_1\ x^1\ x^1));$

$S^0 =_\nu F^0\ (\langle c_0\rangle H_0)\ (\langle c_1\rangle H_0);$

$F^0\ x^0\ x^1 =_\nu \ldots$

Correctness of Translation

- **Theorem:**

  Tree(G) |= A

  if and only if

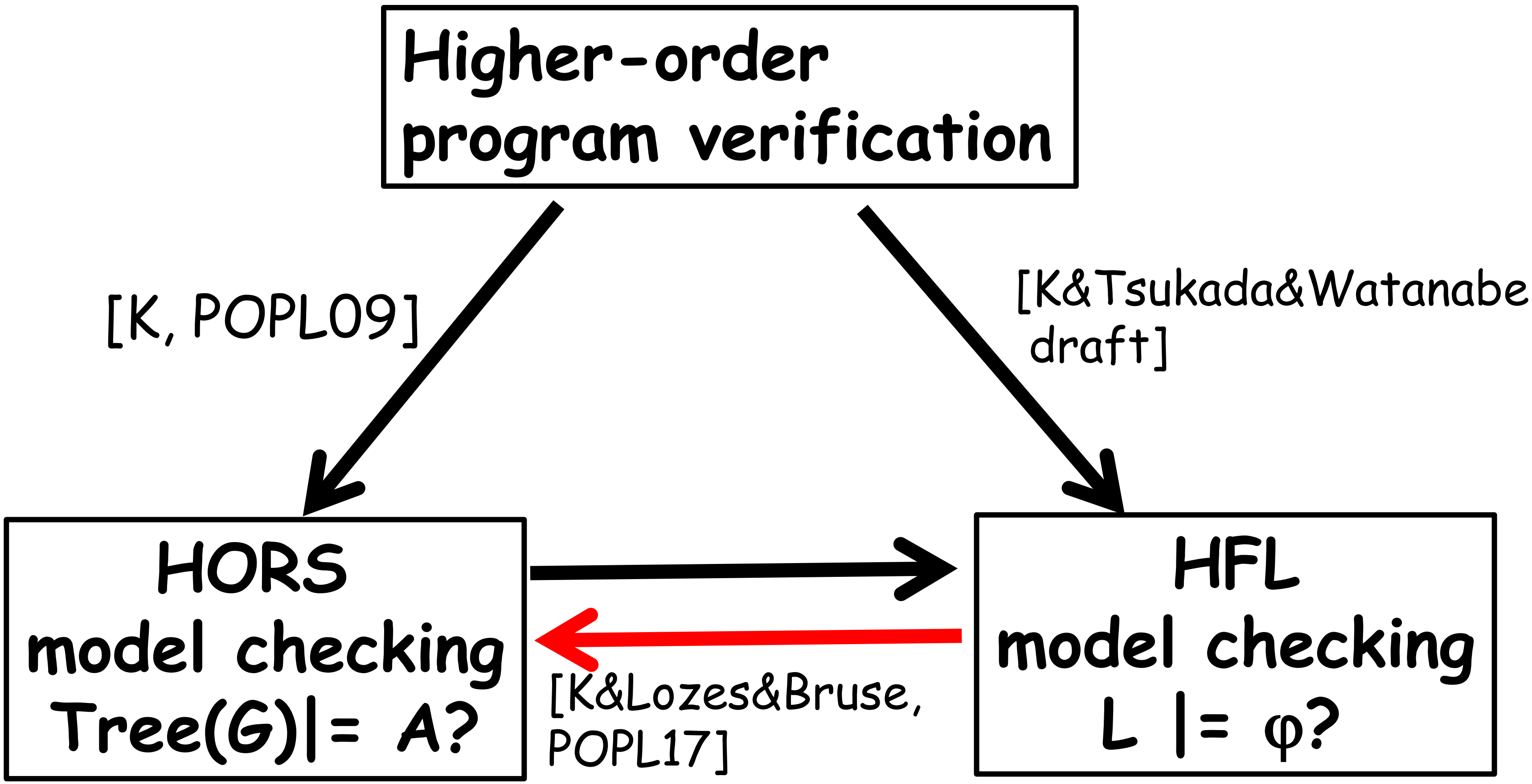
  $L_A \models \varphi_{G,p}$

order(L_A)=order(G)

$|L_A|$ is polynomial in $|A|$

$|\varphi_{G,p}|$ is polynomial in $|G|$, p

HORS vs HFL model checking



From HFL to HORS model checking

- **Input:**
  - LTS L
  - HFL formula φ
- **Output:**
  - HORS $G_{\varphi,c}$
  - APT A_L

  such that $L \models \varphi$ iff $G_{\varphi,c} \models A_L$ for sufficiently large c

Intuition:

- $G_{\varphi,c}$ generates tree representation of the formula equivalent to φ, obtained by unfolding fixpoint formulas sufficiently many times
- A_L accepts trees representing valid formulas

HFL-to-HORS Translation: Overview

$\boxed{F\ X =_\nu \varphi}$

Remove fixpoint operators by finite unfoldings (cf. Kleene fixpoint theorem)

$\boxed{F^{(c)}\ X=[F^{(c-1)}/F]\varphi\ ;\ldots;F^{(1)}\ X=[F^{(0)}/F]\ \varphi;F^{(0)}\ X=\mathrm{true}}$

Convert it to HORS, which generates the tree representation of the formula

$\boxed{F^{(c)}\ X\to[F^{(c-1)}/F]\varphi';\ldots;\ F^{(1)}\ X\to[F^{(0)}/F]\ \varphi';\ F^{(0)}\ X\to\mathrm{true}}$

Parameterize F by a number, and implement numbers (up to $\underbrace{2^{2^{\cdot^{\cdot^{2^{n}}}}}}_{k}$) as functions (cf. [Jones01])

$\boxed{F\ m\ X\to\ \mathrm{if}\ (\mathrm{Zero?}\ m)\ \mathrm{true}\ ([F\ (m-1)\ /F]\varphi')}$

Correctness of Translation

- **Theorem:**

  $L \models \varphi$

  if and only if

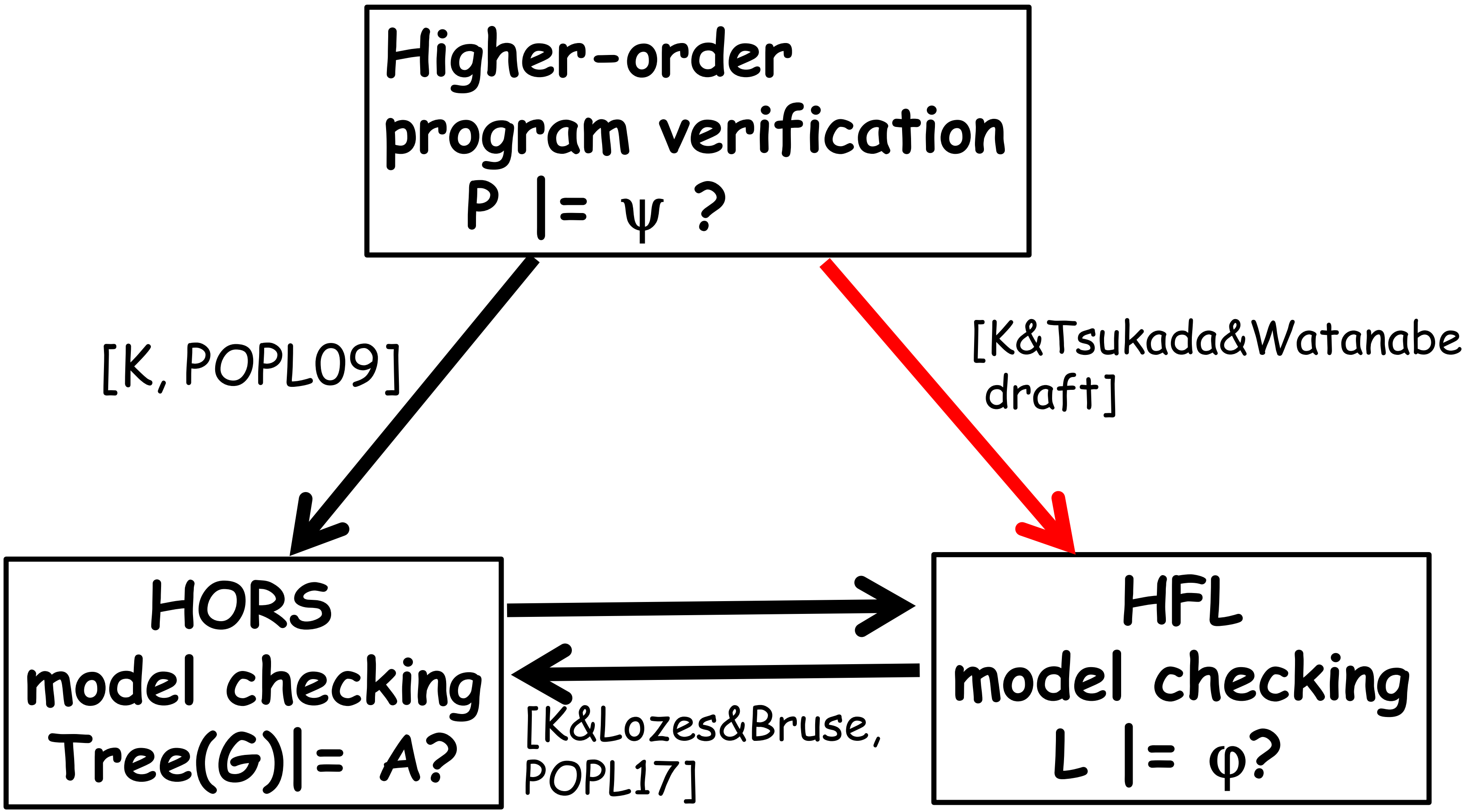
  $G_{\varphi,|L|} \models A_L$

  $\text{order}(G_{\varphi,|L|}) = \text{order}(L)$

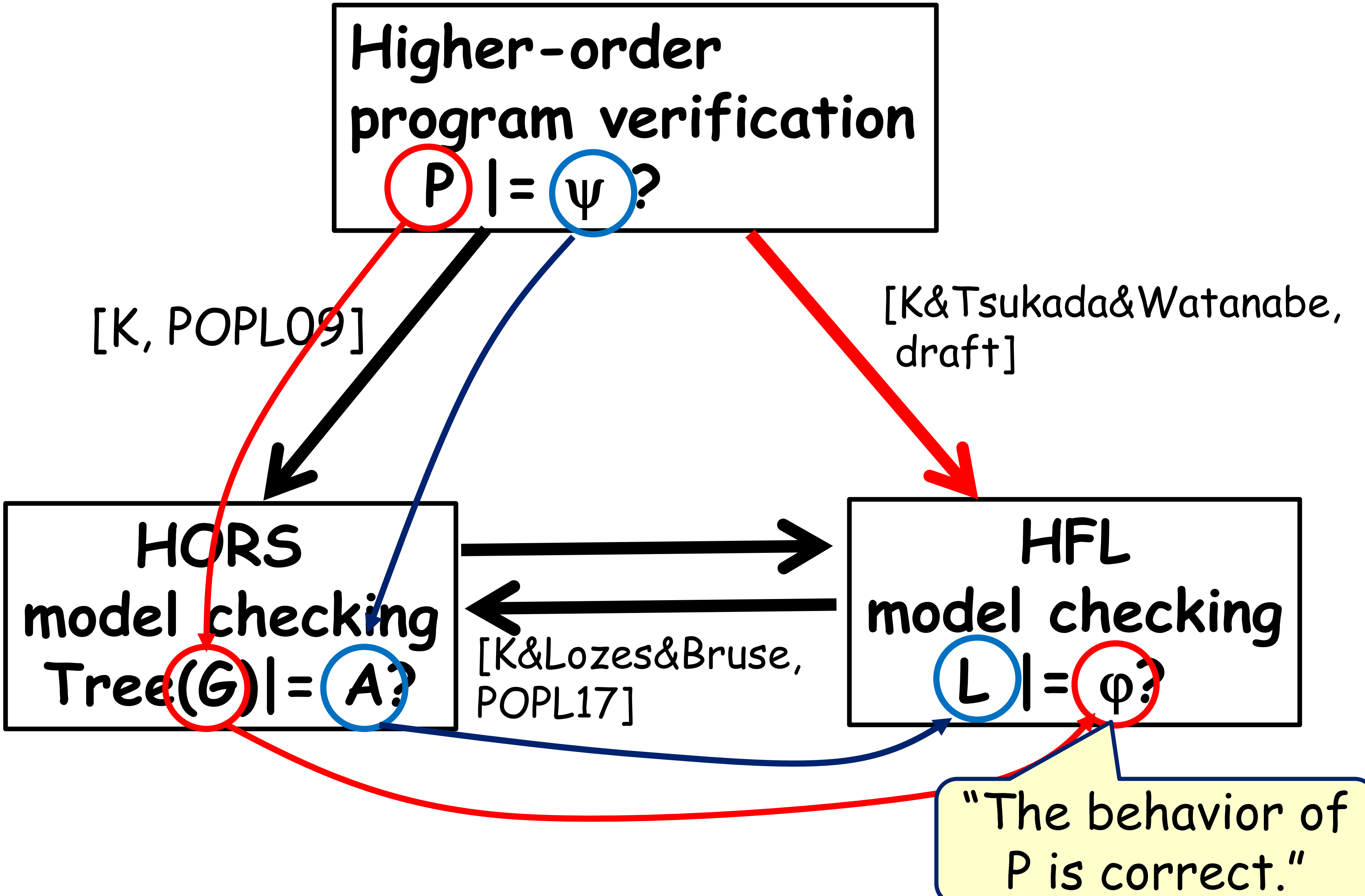
  $|G_{\varphi,|L|}|$ is polynomial in $|\varphi|$ and $|L|$

  $|A_L|$ is polynomial in $|L|$

HORS vs HFL model checking



HORS vs HFL model checking



From Program Verification to HFL Model Checking: Example

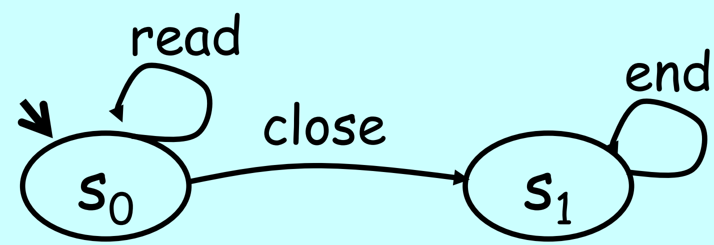
```
let f x =
 if * then close(x)
 else (read(x); f x)
in
let y = open "foo"
in
      f (y)
```

$$F\ x\ k =_{\nu} \langle close \rangle k \land (\langle read \rangle (F\ x\ k))$$

$$S =_{\nu} F\ true\ (\langle end \rangle true)$$

Is the file "foo" accessed according to read* close?

Does LTS:



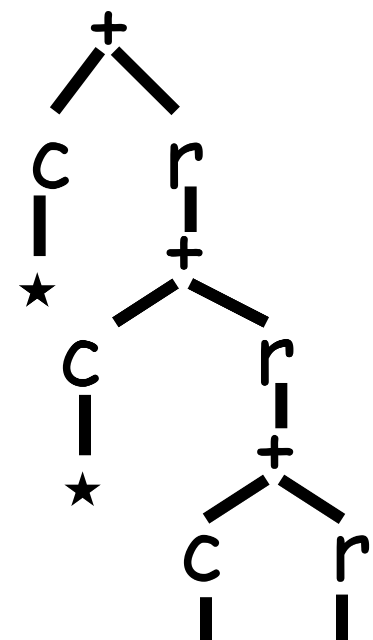
satisfy the formula S?

From Program Verification to HORS Model Checking

```
let f x =
  if * then close(x)
  else (read(x); f x)
in
let y = open "foo"
in
      f (y)
```

$F\ x\ k \rightarrow +\ (c\ k)\ (r(F\ x\ k))$

$S \rightarrow F\ d\ \star$



Is the file "foo" accessed according to read* close?

Is each path of the tree labeled by r*c?

From Program Verification to extended HFL Model Checking

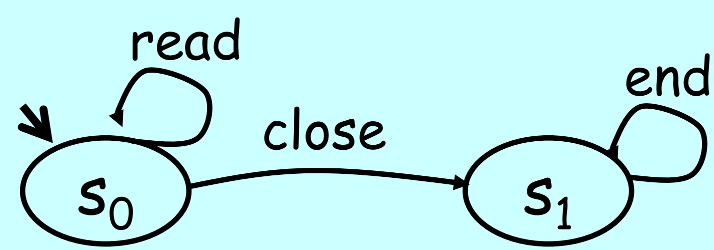
```
let f n x =
 if n≤0 then close(x)
 else
  (read(x); f (n-1) x)
in
let y = open "foo"
in
     f m (y)
```

$F\ n\ x\ k =_{\mu}$
$(n \leq 0 \Rightarrow \langle close \rangle k)$
$\land\ (\neg n \leq 0 \Rightarrow$
$\langle read \rangle (F\ (n-1)\ x\ k))$

$S =_{\mu} F\ m\ true\ (\langle end \rangle true)$

Is the file "foo" accessed according to read* close?

Does LTS:



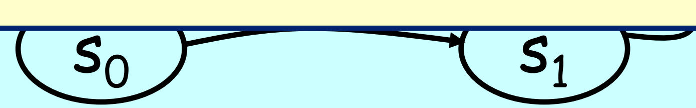
satisfy the formula S?

From Program Verification to extended HFL Model Checking

This approach provides a sound and complete logical characterization of:
- reachability problem
- termination problem
- linear/branching-time temporal properties

for higher-order functional programs with infinite data

Is the file "foo" accessed according to read* close?



satisfy the formula S?

From Termination Verification to extended HFL Model Checking

```
let sum n k =
  if n≤0 then k 0
  else
    sum (n-1) λr.k(r+n)
in sum m (λx.())
```

$\Longrightarrow$

Termination:

$(\mu\ sum.\lambda n.\lambda k.$
$\quad (n\leq 0 \Rightarrow k\ 0) \wedge$
$\quad (n>0 \Rightarrow sum(n-1)\ \lambda r.k(r+n)))$
$m\ (\lambda x.true)$

Non-Termination:

$(\nu\ sum.\lambda n.\lambda k.$
$\quad (n\leq 0 \wedge k\ 0) \vee$
$\quad (n>0 \wedge\ sum(n-1)\ \lambda r.k(r+n)))$
$m\ (\lambda x.false)$

Related Work

- **From HORS to HFL model checking:**
  - Reduction from HORS model checking to nested least/greatest fixedpoint computation [Salvati&Walukiewicz, CSL15]
- **From program verification to HFL model checking:**
  - program verification via:
    - (Constraint) Horn clauses [Bjorner, Gurfinkel, McMillan, Rybalchenko, Unno, ...]
    - Higher-order constraint Horn clauses [Burn, Ong&Ramsay 2017]

      Can be viewed as a restriction to the fragment of HFL without fixpoint alternation and modal operators

Conclusion

- Revealed close relationships among:
  - program verification
  - HFL/HORS model checking
- Reduction from program verification to HFL model checking provides a new, uniform approach to verification of infinite-data higher-order programs
- Future work:
  - development of extended HFL model checkers (cf. recent integration of Horn clause solvers into SMT solvers)