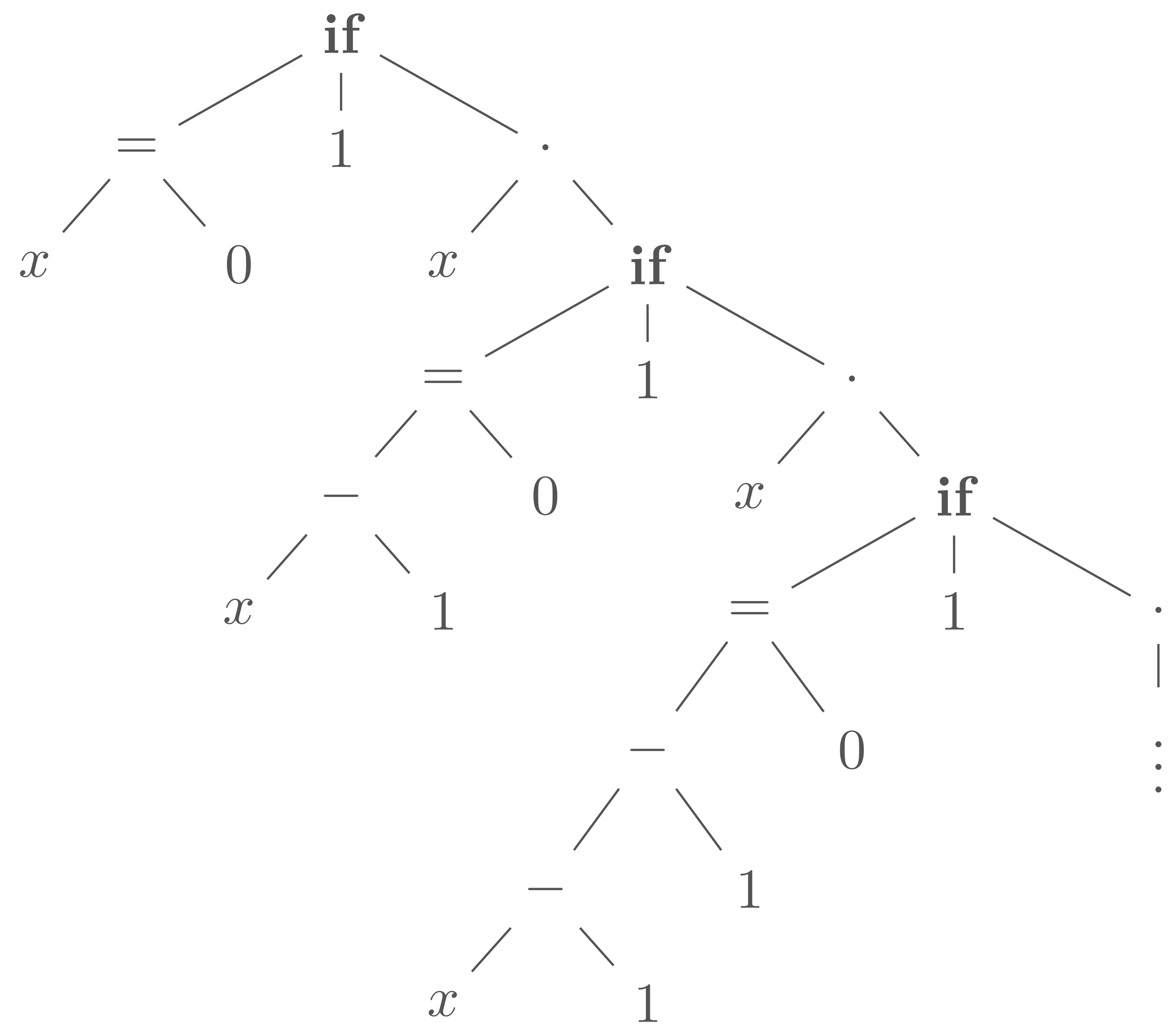
A model for behavioural properties of higher-order programs

Sylvain Salvati and Igor Walukiewicz
Bordeaux University

Verification of behavioural properties of higher-order programs:

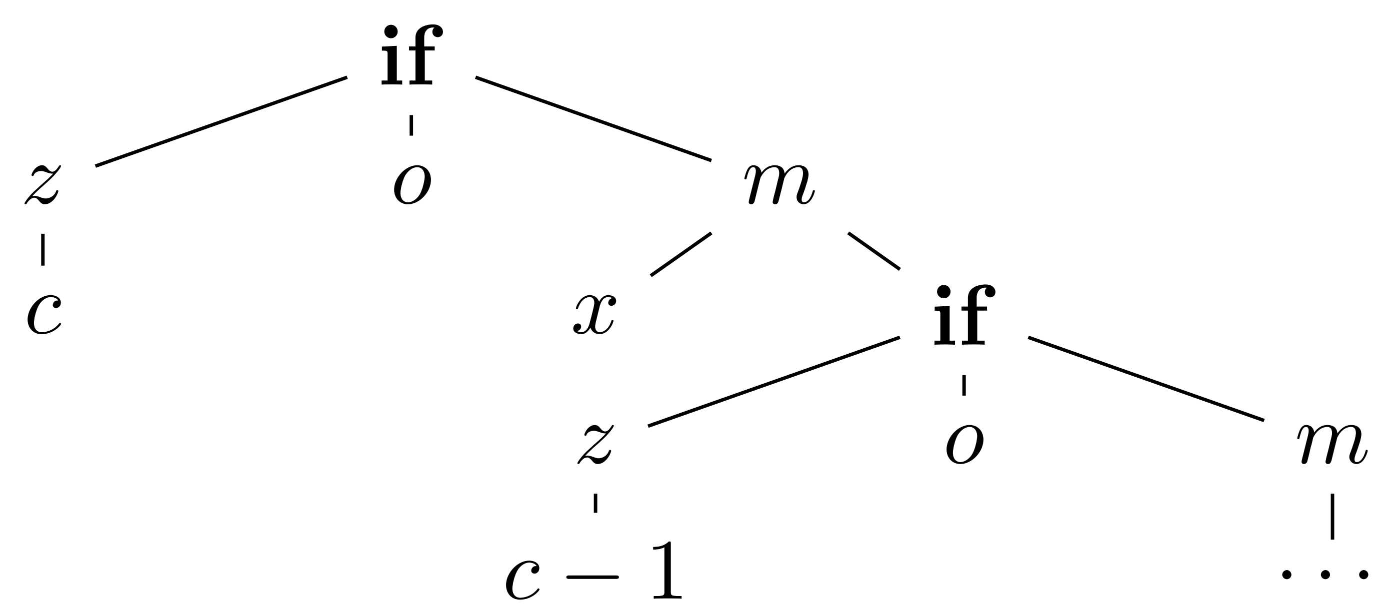
- reachability / safety
  fail constant is reachable / not reachable
- resource usage
  every open file is eventually closed
- method invocation patterns
  m.init should appear before m.usage
- fairness properties
  if access is demanded infinitely often then it is granted infinitely often

$$Fct(x) \equiv \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else } Fct(x-1) \cdot x \, .$$



$$Fct(x) \equiv \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else } Fct(x-1) \cdot x\ .$$

$$\boxed{Y Fct.\ \lambda x.\ \textbf{if-then-else}(z(x), o, m(Fct(x-1), x))}$$



Böhm tree of a term

Verification in three steps

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

1. Program → λ-term

P → M

2. Property → wMSOL-formula

'no fail' → φ

3. Verification

BT(M)$\vDash$φ

Types: $o,\ A \to B$

Typed tems: $c^A,\ x^A,\ (M^{A\to B}N^A)^B,\ (\lambda x^A.M^B)^{A\to B},\ (Yx^A.M^A)^A$

[Kobayashi, POPL'09]

1. Program → **λ**-term
P → M

2. Property → MSOL-formula
'no fail' → **φ**

3. Verification
BT(M)⊨**φ**

$$x \in Z \mid P(x) \mid succ(x, y) \mid \varphi \vee \psi \mid \neg\varphi \mid \forall x.\varphi \mid \forall Z.\varphi$$

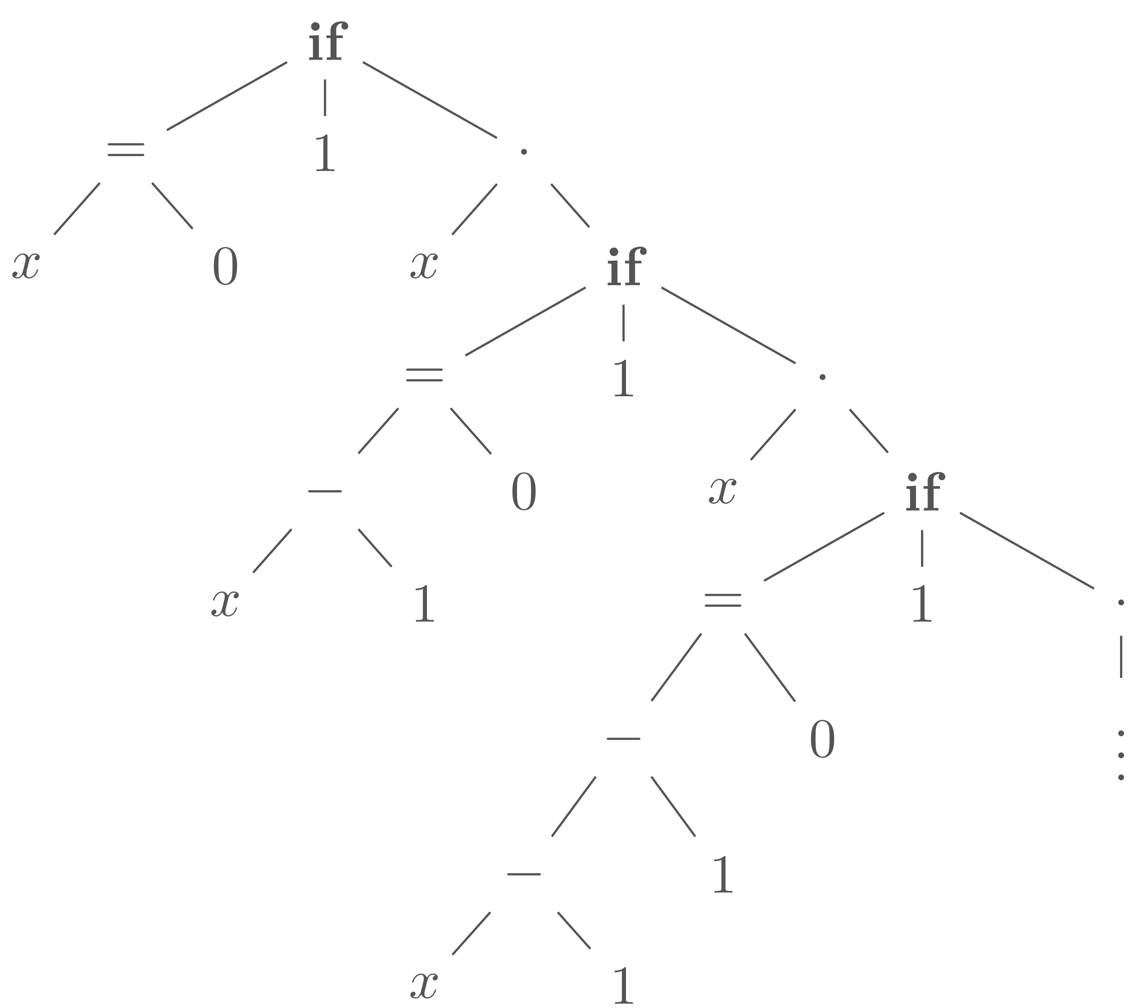
1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

$$Y\, Fct.\ \lambda x.\ \textbf{if-then-else}(z(x), o, m(Fct(x-1), x))$$

Property:
Every path with a left turn is finite



[Ong, LICS'06]

1. Program → λ-term
P → M

2. Property → wMSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Our contribution

We reduce this problem to evaluation in a finite model:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

This approach links verification to abstract interpretation and to typing.

1. Program → λ-term
$P \to M$

2. Property → MSOL-formula
'no fail' → φ

3. Verification
$BT(M) \vDash \varphi$

Our contribution

We reduce this problem to evaluation in a finite model:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

- To verify program fragments
- To type programs
- To do abstract interpretation

1. Program → λ-term
P → M

2. Property → wMSOL-formula
'no fail' → φ

3. Verification
$BT(M) \vDash \varphi$

- To verify program fragments
- To type programs
- To do abstract interpretation

[Salvati, W. 2013]
[Tsukada, Ong 2014]
[Hofmann, Chen 2014]
[Grellois, Meilles 2015]
[Salvati, W. 2015]

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Verification by evaluation:

For a given φ construct a finitary interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Verification by evaluation:

For a given φ construct a finitary interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

- For finite words ➟ semigroups (algebraic theory of regular lang.)
- For infinite words ➟ Wilke algebras
- For finite trees ➟ pre-clones, forest algebras
- For infinite trees ➟

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Verification by evaluation:

For a given φ construct a finitary interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

Models with least/greatest fix points can only recognise boolean combinations of reachability and safety properties.

Semantics: GFP-models

Types: o, $A \to B$

Typed tems: c^A, x^A, $(M^{A\to B} N^A)^B$, $(\lambda x^A.M^B)^{A\to B}$, $(Y x^A.M^A)^A$

Tree signature: constants have types o or $(o \to \cdots \to o \to o)$.

A GFP model

$\mathcal{D}^{\mathcal{A}} = \langle \{D_A\}_{A\in\mathcal{T}}, [\![c]\!], \ldots \rangle$ where

$$D_o = \text{finite lattice} \qquad D_{A\to B} = \text{mon}[D_A \mapsto D_B]$$

$$[\![Y f^{A\to A}.\ M^A]\!]_v = \mathsf{GFP}(\lambda F.[\![M]\!]_{v[F/f]})$$

A model can *recognise* a set of terms:
a set $F \subseteq \mathcal{D}_0$ defines a set of closed terms $\{M : [\![M]\!]^{\mathcal{D}} \in F\}$.

What can finite GFP-models recognise?

TAC-automata

Tree automata with trivial acceptance conditions

$$\mathcal{A} = \langle Q,\ \Sigma,\ \delta_i : Q \times \Sigma^{(i)} \to \mathcal{P}(Q^i)\rangle$$

Every run is accepting.

$$\text{TAC-automaton} \quad \equiv \quad \nu\text{-formulas} \quad \equiv \text{safety properties}$$

Models with least/greatest fix points can only recognise boolean combinations of reachability and safety properties.

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Verification by evaluation:

For a given φ construct a finitary interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

1. Program → λ-term
P → M

2. Property → wMSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Verification by evaluation:

For a given φ construct a finitary interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

Thm: For every MSOL property there is effectively a finitary model recognising it.

For a given φ construct a finitary interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

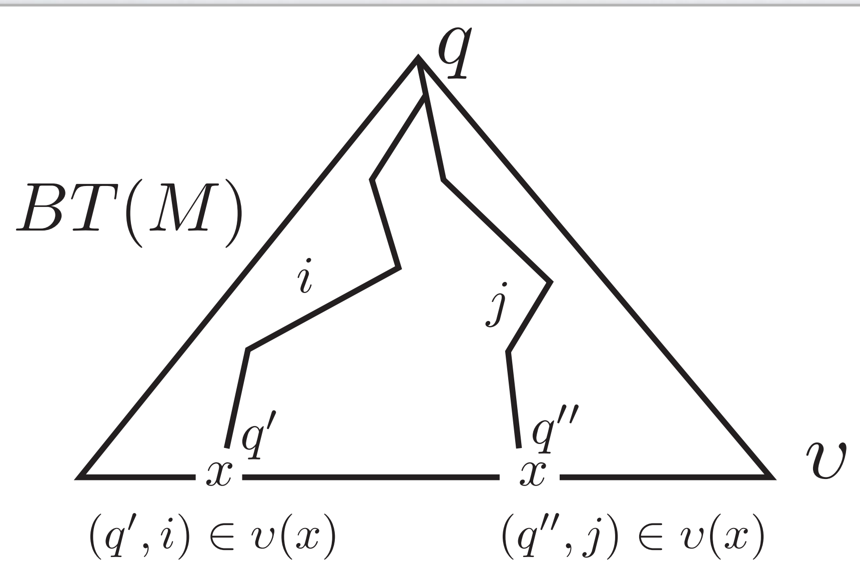
$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

$$\mathcal{A} = \langle Q,\ \Sigma,\ \delta_i : Q \times \Sigma^{(i)} \to \mathcal{P}(Q^i),\ rk : Q \to [m] \rangle$$

max parity condition

$M : o$ $\qquad [\![M]\!] \in \mathcal{P}(Q)$

$M : o \to o$ $\qquad [\![M]\!] \in \mathcal{P}(Q \times [m]) \to \mathcal{P}(Q)$

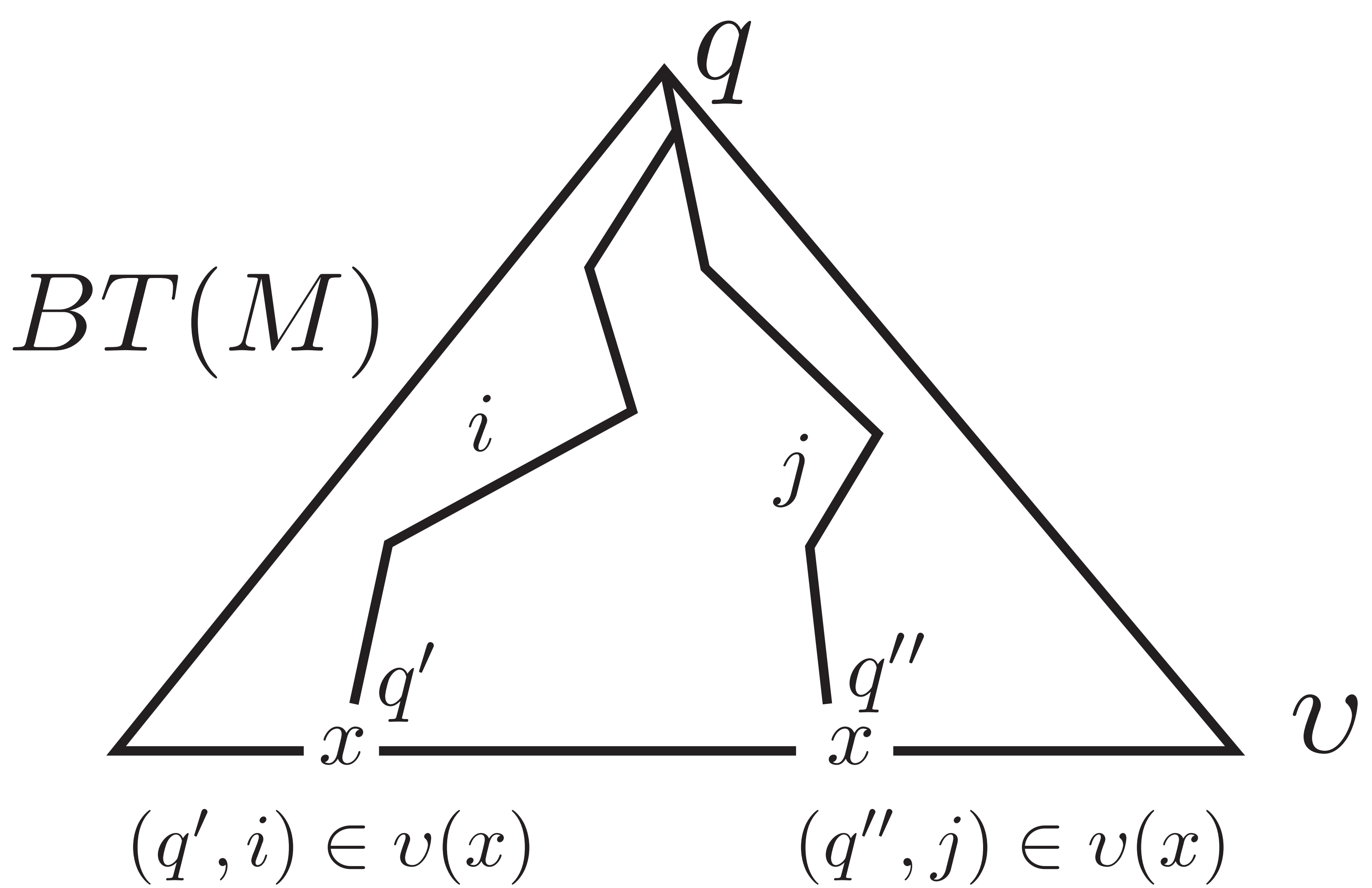


$$q \in [\![M]\!]\{(q', i), (q'', j)\}$$

$M : o$ $\quad\quad [\![M]\!] \in \mathcal{P}(Q)$

$M : o \to o$ $\quad\quad [\![M]\!] \in \mathcal{P}(Q \times [m]) \to \mathcal{P}(Q)$

$q \in [\![M]\!]\{(q', i), (q'', j)\}$



For a given φ construct a finitary interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

$M : o$	$[\![M]\!] \in \mathcal{P}(Q)$
$M : o \to o$	$[\![M]\!] \in \mathcal{P}(Q \times [m]) \to \mathcal{P}(Q)$

$$\mathcal{S}_o = \mathcal{P}(Q) \qquad \mathcal{S}_{A \to B} = \mathcal{D}_A \to \mathcal{S}_B$$

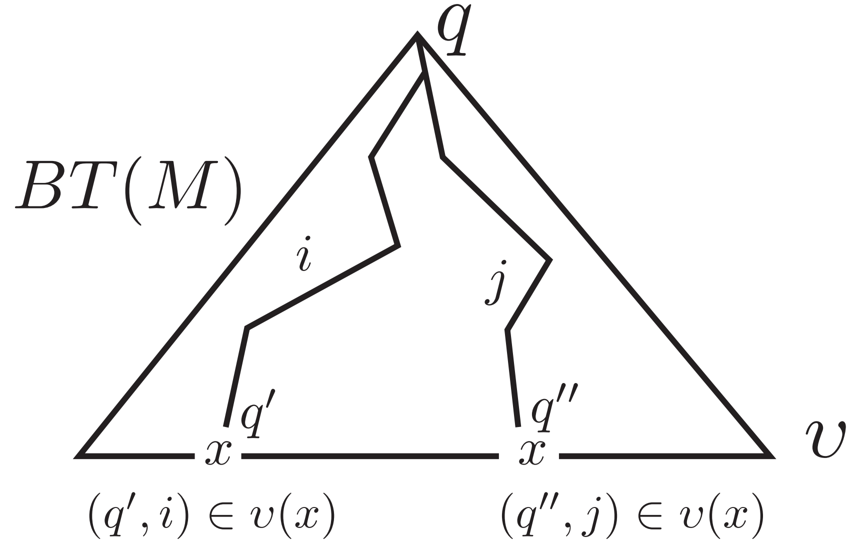
$$\mathcal{D}_o = \mathcal{P}(Q \times [m]) \qquad \mathcal{D}_{A \to B} = \mathcal{D}_A \to \mathcal{D}_B$$

$$\mathcal{S}_o = \mathcal{P}(Q) \qquad \mathcal{S}_{A\to B} = \mathcal{D}_A \to \mathcal{S}_B$$
$$\mathcal{D}_o = \mathcal{P}(Q \times [m]) \qquad \mathcal{D}_{A\to B} = \mathcal{D}_A \to \mathcal{D}_B$$

Valuation: $v(x^A) \in \mathcal{D}_A$

Semantics: $[\![M^A, v]\!] \in \mathcal{S}_A$

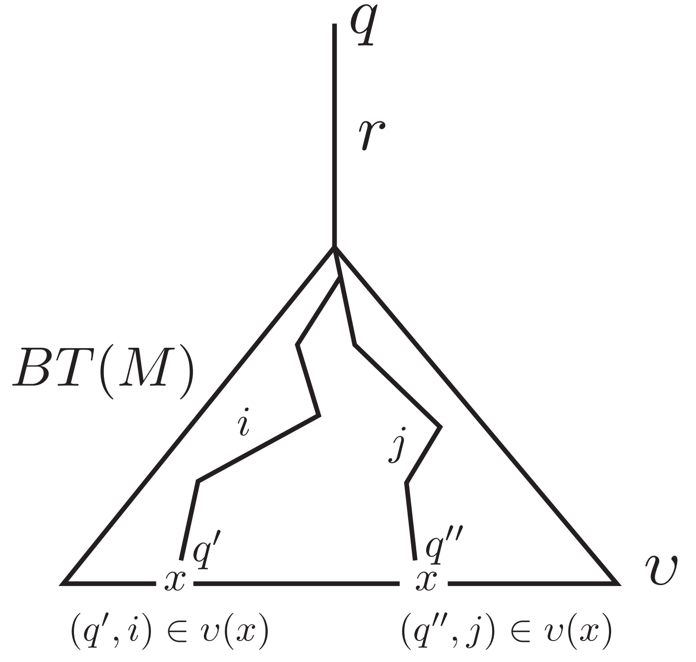
$q \in [\![M, v]\!]$ if



$$\mathcal{S}_o = \mathcal{P}(Q) \qquad \mathcal{S}_{A\to B} = \mathcal{D}_A \to \mathcal{S}_B$$
$$\mathcal{D}_o = \mathcal{P}(Q \times [m]) \qquad \mathcal{D}_{A\to B} = \mathcal{D}_A \to \mathcal{D}_B$$

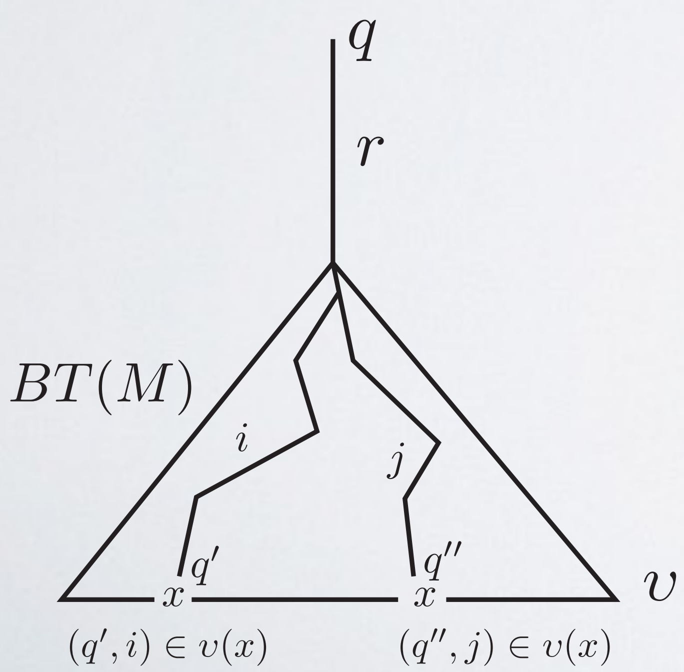
Lifting operation $d\lfloor_r$ for $d \in D_A$ and $r \in [m]$.

$q \in [\![M, v\lfloor_r]\!]$ if



$$\mathcal{S}_o = \mathcal{P}(Q) \qquad\qquad \mathcal{S}_{A\to B} = \mathcal{D}_A \to \mathcal{S}_B$$
$$\mathcal{D}_o = \mathcal{P}(Q \times [m]) \qquad\qquad \mathcal{D}_{A\to B} = \mathcal{D}_A \to \mathcal{D}_B$$

$$[\![M, v]\!] \in \mathcal{S}_A \qquad \langle\!\langle M, v\rangle\!\rangle \in \mathcal{D}_A$$



$$\langle\!\langle M, v\rangle\!\rangle(\vec{f}) = \{(q, r) : q \in [\![M, v\downharpoonleft_r]\!](\vec{f})\}$$
$$[\![M, v]\!](\vec{f}) = \{q : (q, 0) \in \langle\!\langle M, v\rangle\!\rangle(\vec{f})\}\ .$$

$$\mathcal{S}_o = \mathcal{P}(Q) \qquad \mathcal{S}_{A\to B} = \mathcal{D}_A \to \mathcal{S}_B$$

$$\mathcal{D}_o = \mathcal{P}(Q \times [m]) \qquad \mathcal{D}_{A\to B} = \mathcal{D}_A \to \mathcal{D}_B$$

$$[\![M, v]\!] \in \mathcal{S}_A \qquad \langle\!\langle M, v\rangle\!\rangle \in \mathcal{D}_A$$

$$\begin{aligned}
[\![x, v]\!]\vec{f} &= \{q : (q, 0) \in v(x)(\vec{f})\} \\
[\![a, v]\!]f_1 \dots f_k &= \{q : \exists_{(q_1,\dots,q_k)\in\delta_{\mathcal{A}}(q,a)}.\ \forall_{i=1,\dots,k}.\ (q_i, 0) \in f_i\lfloor_{rk(q_i)}\} \\
[\![\lambda x.M, v]\!]f &= [\![M, v[f/x]]\!] \\
[\![MN, v]\!] &= [\![M, v]\!]\langle\!\langle N, v\rangle\!\rangle \\
[\![Y, v]\!]f &= \dots
\end{aligned}$$

[Kobayashi, Ong LICS'09]
[Salvati, W. ICALP'11][Tsukada, Ong LICS'14][Grellois, Mellies, MFCS'15]

$$
\begin{aligned}
[\![x,v]\!]\vec{f} =& \{q : (q,0) \in v(x)(\vec{f})\} \\
[\![a,v]\!]f_1 \dots f_k =& \{q : \exists_{(q_1,\dots,q_k)\in\delta_{\mathcal{A}}(q,a)}.\ \forall_{i=1,\dots,k}.\ (q_i,0) \in f_i\!\downharpoonright_{rk(q_i)}\} \\
[\![\lambda x.M,v]\!]f =& [\![M, v[f/x]]\!] \\
[\![MN,v]\!] =& [\![M,v]\!]\langle\!\langle N,v\rangle\!\rangle \\
[\![Y,v]\!]f =& \dots
\end{aligned}
$$

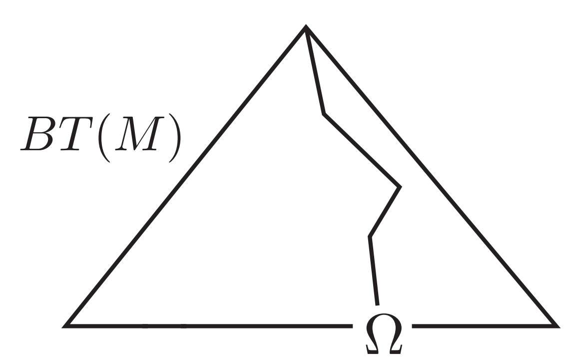
The above semantics works only for **Ω**-blind automata

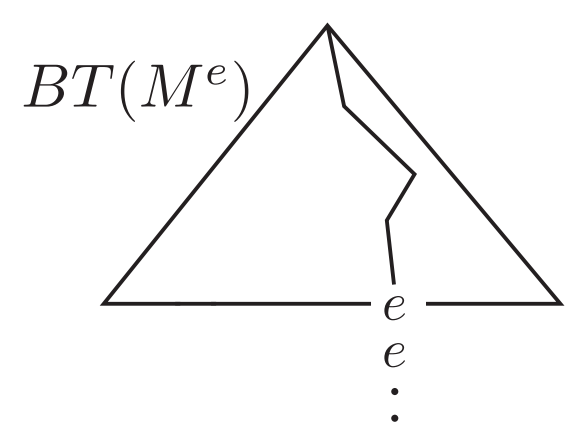
$$\delta(\Omega, q) = \{\emptyset\} \qquad \text{for all } q$$

automaton accept unconditionally on **Ω**

$$Y M \quad \mapsto \quad Y(\lambda\vec{x}.\ e(M\vec{x}))$$

Translation to eliminate **Ω** from Böhm trees:





How to deal directly with all automata?

There are too many functions in $\mathcal{S}_A$.

It is not decidable which of those are semantics of terms [Loader].

How to deal directly with all automata?

There are too many functions in $\mathcal{S}_A$.

It is not decidable which of those are semantics of terms [Loader].

It suffices to require that all functions $f \in \mathcal{D}_{A\to B}$ satisfy:

$$\forall g \in \mathcal{D}_A.\ \forall q \in Q.\ (f(g)){\Downarrow_q} = (f(g{\downharpoonleft_{rk(q)}})){\Downarrow_q} \qquad (\texttt{strat})$$

where $f{\Downarrow_q} = \{r : (q,r) \in f\}$, and $f{\Downarrow_q}(g) = (f(g)){\Downarrow_q}$.

Thm: For a given φ we can construct an interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

Applications of models

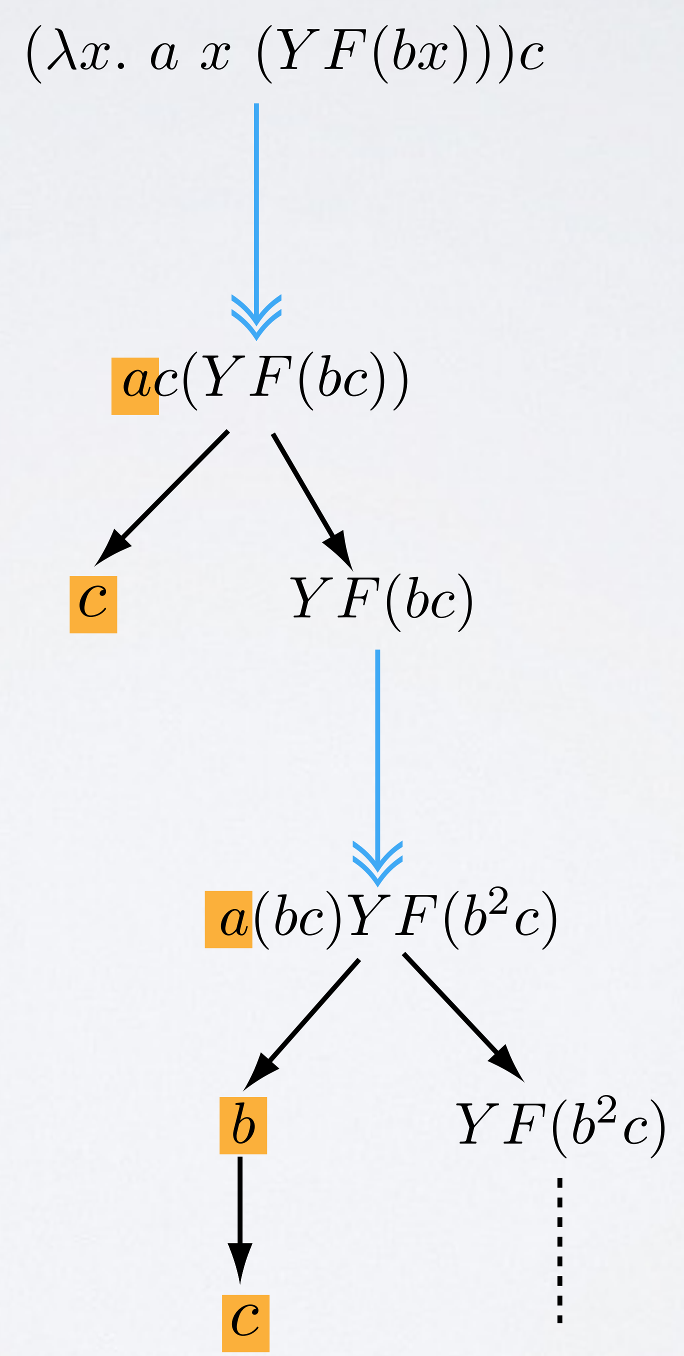
1. Decidability of the model-checking problem for MSO

Given an property φ and term M:

- construct the model $\mathcal{D}^{\varphi}$, and
- calculate the semantics of M in $\mathcal{D}^{\varphi}$.

Applications of models

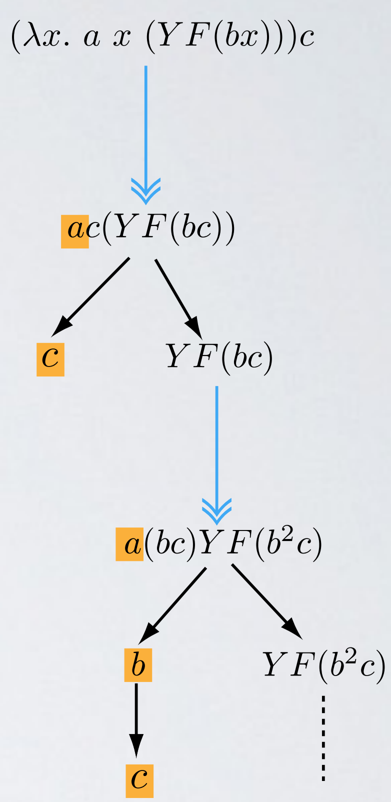
3. Program transformation



Applications of models

3. Program transformation

$$
\begin{aligned}
\alpha^{\bullet} &= \alpha \text{ when } \alpha \text{ is atomic}\\
(\alpha \to \beta)^{\bullet} &= \alpha^{\bullet} \to [\alpha] \to \beta^{\bullet}
\end{aligned}
$$



$$
\begin{aligned}
[\lambda x^{\alpha}.M, \upsilon] =& \lambda x^{\alpha^{\bullet}} \lambda y^{[\alpha]}.\ \text{case } y^{[\alpha]}\{d \to [M, \upsilon[d/x^{\alpha}]]\}_{d \in \mathcal{S}_{\alpha}}\\
[MN, \upsilon] =& [M, \upsilon]\,[N, \upsilon]\,[\![N]\!]^{\upsilon}\\
[a, \upsilon] =& \lambda x_1^{0} \lambda y_1^{[0]} \lambda x_2^{0} \lambda y_2^{[0]}.\\
& \quad \text{case } y_1^{[0]}\{d_1 \to \text{case } y_2^{[0]}\{d_2 \to a^{\rho(a)\, d_1\, d_2} x_1 x_2\}_{d_2 \in \mathcal{S}_0}\}_{d_1 \in \mathcal{S}_0}\\
\left[Y^{(\alpha \to \alpha) \to \alpha} M, \upsilon\right] =& Y^{(\alpha^{\bullet} \to \alpha^{\bullet}) \to \alpha^{\bullet}}(\lambda x^{\alpha^{\bullet}}.\, [M, \upsilon]\, x^{\alpha^{\bullet}} [\![YM]\!]^{\upsilon})\ .
\end{aligned}
$$

Applications of models

4. Transfer theorem for MSO

Thm (Transfer)[Salvati & W.]
Fix a signature Σ, set of types $\mathcal{T}$, and a set of variables $\mathcal{X}$ (all finite sets).
For every MSOL formula φ there is an MSOL formula $\widehat{\varphi}$ s.t. for every term M over Σ, $\mathcal{T}$, $\mathcal{X}$:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

Consequences of the transfer theorem

$$M \vDash \widehat{\varphi} \qquad \text{iff} \qquad BT(M) \vDash \varphi$$

The set of SN terms over fixed set of variables is definable in MSOL

For a fixed $\mathcal{T}$ and $\mathcal{X}$ there is an MSOL formula defining the set of terms $M \in \mathit{Terms}(\Sigma, \mathcal{T}, \mathcal{X})$ having a normal form.

Take φ defining the set of finite trees and consider $\widehat{\varphi}$.

Consequences of the transfer theorem

$$M \vDash \widehat{\varphi} \qquad \text{iff} \qquad BT(M) \vDash \varphi$$

A « synthesis from modules » framework

Given λY-terms $M_1, \ldots, M_k$ and a formula φ.
Decide if one can construct from these terms a λY term K such that $eval(K) \vDash \varphi$.

- We can restrict to solutions K of the form
  $(\lambda x_1 \ldots x_k.\ N)M_1, \ldots, M_k$
  for some term N without constants and λ-abstractions.
- Let ψ be a formula defining terms of this form.
- There is a solution iff the formula $\psi \wedge \widehat{\varphi}$ is satisfiable.

$$M \vDash \widehat{\varphi} \qquad \text{iff} \qquad eval(M) \vDash \varphi$$

Consequences of the transfer theorem

$$M \vDash \widehat{\varphi} \qquad \text{iff} \qquad BT(M) \vDash \varphi$$

Higher-order matching with restricted no of variables

For a fixed $\mathcal{X}$. Given M and K (without fixpoints) decide if there is a substitution σ such that

$$M\sigma =_\beta K$$

Substitution Σ can use only terms from $\mathit{Terms}(\Sigma, \mathcal{T}, \mathcal{X})$.

- Let $\mathit{shape}(N)$ be MSOL formula defining the set of terms in $\mathit{Terms}(\Sigma, \mathcal{T}, \mathcal{X})$ that can be obtained from N by substitutions.
- Let $\varphi \equiv \mathit{shape}(K)$.
- There is desired σ iff the formula $\mathit{shape}(M) \wedge \widehat{\varphi}$ is satisfiable.

If there is a solution then there is a finite one.

CONCLUSIONS

A semantics for all MSOL properties of Böhm trees

- stratification property
- a formula for the fix point.

Most results on higher-order verification follow from this construction.

More abstract description of the model

Determine expressive power of finitary models.

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Extending verification methods, from transition systems to a higher-order program calculus.

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Verification by evaluation:

For a given φ construct an interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)$\vDash\varphi$

Verification by evaluation

Extending verification methods, from transition systems to a higher-order program calculus.

Extending abstract interpretation to new kinds of models, and higher-order.

Extending typing with new kinds of types, namely behavioural types.

1. Program → λ-term
P → M

2. Property → MSOL-formula
'no fail' → φ

3. Verification
BT(M)⊨φ

Verification by evaluation:

For a given φ construct an interpretation of λY-terms D, and a set $F \subseteq D$ s.t. for every λY-term M:

$$BT(M) \vDash \varphi \quad \text{iff} \quad [\![M]\!]^D \in F.$$

- Type systems
- Program tranformation
- Transfer theorem

- Verification by evaluation
- Abstraction/refinement
- Evaluating programs directly