Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

Sylvain Salvati and Igor Waluckiewicz
INRIA, LaBRI, Université de Bordeaux

FREC Ile de Ré
Program schemes model checking

Program $\rightarrow$ Scheme $\rightarrow$ Evaluation tree $\rightarrow$ Semantics

Model checking
verifying finite state properties of evaluation trees
Model checking of higher-order programs and denotational semantics of λ-calculus.

\[
\text{fold } f \ a \ l = \text{if } l = [] \text{ then } a \text{ else } f \ (\text{hd} \ l) \ (\text{fold} \ f \ a \ (\text{tl} \ l))
\]
Higher-order schemes are $\lambda Y$-terms:

**Types**

\[ \mathcal{T} : \quad \alpha, \beta, \gamma ::= 0 \mid (\alpha \rightarrow \beta) \]

**$\lambda Y$-calculus**

\[ \Lambda Y : \quad M^\alpha, N^\beta ::= x^\alpha \mid c^\alpha \mid (\lambda x^\alpha . M^\beta)^{\alpha \rightarrow \beta} \mid (M^{\alpha \rightarrow \beta} N^\alpha)^\beta \]

\[(\beta) \quad (\lambda x . M)N = M[N/x] \]

\[(\eta) \quad \lambda x . Mx = M \text{ when } x \notin \text{fv}(M) \]

\[(\delta) \quad YM = M(YM) \]
Higher-order schemes are $\lambda Y$-terms:

**Types**

$$\mathcal{T} : \quad \alpha, \beta, \gamma ::= 0 \mid (\alpha \to \beta)$$

**$\lambda Y$-calculus**

$$\Lambda Y : \quad M^\alpha, N^\beta ::= x^\alpha \mid c^\alpha \mid (\lambda x^\alpha. M^\beta)^{\alpha \to \beta} \mid (M^{\alpha \to \beta} N^\alpha)^{\beta} \mid (YM^{\alpha \to \alpha})^\alpha$$

1. $$(\beta) \quad (\lambda x. M)N = M[N/x]$$
2. $$(\eta) \quad \lambda x. Mx = M \text{ when } x \notin \text{fv}(M)$$
3. $$(\delta) \quad YM = M(YM)$$
Böhm tree for $\Lambda Y$

Böhm trees are a sort of infinite normal form for $\Lambda Y$-terms

If $M$ reduces to $\lambda x_1 \ldots x_n.hM_1 \ldots M_n$:

$$BT(M) = \lambda x_1 \ldots x_n.h$$

$$BT(M_1) \cdots BT(M_n)$$

otherwise:

$$BT(M) = \Omega$$
Böhm tree for \( \Lambda Y \)

Böhm trees are a sort of infinite normal form for \( \Lambda Y \)-terms.

If \( M \) reduces to \( \lambda x_1 \ldots x_n.hM_1 \ldots M_n \):

\[
\text{BT}(M) = \begin{cases} \\
\lambda x_1 \ldots x_n.h \\
\text{BT}(M_1) \ldots \\
\text{BT}(M_n) \\
\end{cases}
\]

otherwise:

\[
\text{BT}(M) = \Omega
\]
Böhm tree for $\Lambda Y$

Böhm trees are a sort of infinite normal form for $\Lambda Y$-terms

If $M$ reduces to $\lambda x_1 \ldots x_n. h M_1 \ldots M_n$:

$$BT(M) = \lambda x_1 \ldots x_n. h$$

$$BT(M_1) \cdots \quad BT(M_n)$$

otherwise:

$$BT(M) = \Omega$$
Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

\[
\text{fold } f \ a \ l = \text{if } l = [] \text{ then } a \text{ else } f \ (\text{hd } l) \ (\text{fold } f \ a \ (\text{tl } l))
\]

\[
M = Y \lambda \text{fold } f \ a \ l.\text{ite} \ (\text{=}l [] ) \ a \ (f \ (\text{hd } l) \ (\text{fold } f \ a \ (\text{tl } l)))
\]

$BT(M)$ is:

\[
\begin{align*}
\lambda \ f \ a \ l.\text{ite} = \\
\text{ite} \ a \ f \\
\text{=} \ \\
\text{a} \\
l \\
\text{[]} \\
\text{f} \\
\text{hd} \\
\text{ite} \\
\text{a} \\
\text{f} \\
\text{tl} \\
\text{[]} \\
\text{hd} \\
\text{ite} \\
\text{tl} \\
l \\
\end{align*}
\]
Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

Exploring the limits of effective denotational semantics

Advantages of denotational semantics:

- Characterizes invariants modulo computation
- Plays the role of monoids/algebra in usual formal language
Exploring the limits of *effective* denotational semantics

**Effective denotational semantics:**

- Interpretation domains that can be effectively constructed at every types
- The interpretation of terms are all computable
- In practice, we use only finite domains of interpretations
- We want to understand the kinds of properties we can express on results produced by $\Lambda Y$ in this context because:
  - It gives simple decidability results
  - It allows to understand in a deeper way the nature of those properties and the kinds of algorithms they require
  - It may yield original domains of interpretation for $\Lambda Y$
Exploring the limits of *effective* denotational semantics

Effective denotational semantics:

- Interpretation domains that can be effectively constructed at every types
- The interpretation of terms are all computable
- In practice, we use only finite domains of interpretations
- We want to understand the kinds of properties we can express on results produced by \( \Lambda Y \) in this context because:
  - It gives simple decidability results
  - It allows to understand in a deeper way the nature of those properties and the kinds of algorithms they require
  - It may yield original domains of interpretation for \( \Lambda Y \)
Model checking of higher-order programs and denotational semantics of λ-calculus.

Exploring the limits of effective denotational semantics

Effective denotational semantics:

- Interpretation domains that can be effectively constructed at every types
- The interpretation of terms are all computable
- In practice, we use only finite domains of interpretations
- We want to understand the kinds of properties we can express on results produced by ΛY in this context because:
  - It gives simple decidability results
  - It allows to understand in a deeper way the nature of those properties and the kinds of algorithms they require
  - It may yield original domains of interpretation for ΛY
Exploring the limits of effective denotational semantics

Effective denotational semantics:

- Interpretation domains that can be effectively constructed at every types
- The interpretation of terms are all computable
- In practice, we use only finite domains of interpretations
- We want to understand the kinds of properties we can express on results produced by $\Lambda Y$ in this context because:
  - It gives simple decidability results
  - It allows to understand in a deeper way the nature of those properties and the kinds of algorithms they require
  - It may yield original domains of interpretation for $\Lambda Y$
Exploring the limits of effective denotational semantics

Effective denotational semantics:

- Interpretation domains that can be effectively constructed at every types
- The interpretation of terms are all computable
- In practice, we use only finite domains of interpretations
- We want to understand the kinds of properties we can express on results produced by ΛY in this context because:
  - It gives simple decidability results
  - It allows to understand in a deeper way the nature of those properties and the kinds of algorithms they require
  - It may yield original domains of interpretation for ΛY
Exploring the limits of effective denotational semantics

Effective denotational semantics:

- Interpretation domains that can be effectively constructed at every types
- The interpretation of terms are all computable
- In practice, we use only finite domains of interpretations
- We want to understand the kinds of properties we can express on results produced by $\Lambda Y$ in this context because:
  - It gives simple decidability results
  - It allows to understand in a deeper way the nature of those properties and the kinds of algorithms they require
  - It may yield original domains of interpretation for $\Lambda Y$
Exploring the limits of effective denotational semantics

Effective denotational semantics:

- Interpretation domains that can be effectively constructed at every types
- The interpretation of terms are all computable
- In practice, we use only finite domains of interpretations
- We want to understand the kinds of properties we can express on results produced by $\Lambda Y$ in this context because:
  - It gives simple decidability results
  - It allows to understand in a deeper way the nature of those properties and the kinds of algorithms they require
  - It may yield original domains of interpretation for $\Lambda Y$
Exploring the limits of effective denotational semantics

Effective denotational semantics:

- Interpretation domains that can be effectively constructed at every types
- The interpretation of terms are all computable
- In practice, we use only finite domains of interpretations
- We want to understand the kinds of properties we can express on results produced by $\Lambda Y$ in this context because:
  - It gives simple decidability results
  - It allows to understand in a deeper way the nature of those properties and the kinds of algorithms they require
  - It may yield original domains of interpretation for $\Lambda Y$
Models of $\Lambda Y$

A model $M$ is $((M_\alpha)_{\alpha \in T}, \bullet, \rho)$ where

- for every $f \in M_{\alpha \rightarrow \beta}$ and $g \in M_\alpha$, $f \bullet g \in M_\beta$. 
A model $\mathcal{M}$ is $((\mathcal{M}_\alpha)_{\alpha \in \mathcal{T}}, \bullet, \rho)$ where

- for every $f \in \mathcal{M}_\alpha \rightarrow \beta$ and $g \in \mathcal{M}_\alpha$, $f \bullet g \in \mathcal{M}_\beta$,
- for every $f, f' \in \mathcal{M}_\alpha \rightarrow \beta$, for every $g \in \mathcal{M}_\alpha$
  $f \bullet g = f' \bullet g$ implies $f = f'$

\[\begin{array}{c}
\mathcal{M}_\alpha \\
\mathcal{M}_0 \\
\vdots
\end{array}\]
Model checking of higher-order programs and denotational semantics of \(\lambda\)-calculus.

### Models of \(\Lambda Y\)

A model \(\mathcal{M}\) is \((\mathcal{M}_\alpha)_{\alpha \in \mathcal{T}}, \bullet, \rho)\) where

- for every \(f \in \mathcal{M}_{\alpha \rightarrow \beta}\) and \(g \in \mathcal{M}_\alpha\), \(f \bullet g \in \mathcal{M}_\beta\),
- for every \(f, f' \in \mathcal{M}_{\alpha \rightarrow \beta}\), for every \(g \in \mathcal{M}_\alpha\), \(f \bullet g = f' \bullet g\) implies \(f = f'\)
A model $\mathcal{M}$ is $((\mathcal{M}_\alpha)_{\alpha \in \mathcal{T}}, \bullet, \rho)$ where

- for every $f \in \mathcal{M}_\alpha \rightarrow \beta$ and $g \in \mathcal{M}_\alpha$, $f \bullet g \in \mathcal{M}_\beta$,
- for every $f, f' \in \mathcal{M}_\alpha \rightarrow \beta$, for every $g \in \mathcal{M}_\alpha$, $f \bullet g = f' \bullet g$ implies $f = f'$

**Axioms of Interpretation**
Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

## Models of $\Lambda Y$

A model $\mathcal{M}$ is $((\mathcal{M}_\alpha)_{\alpha \in T}, \bullet, \rho)$ where

- for every $f \in \mathcal{M}_{\alpha \rightarrow \beta}$ and $g \in \mathcal{M}_\alpha$, $f \bullet g \in \mathcal{M}_\beta$,
- for every $f, f' \in \mathcal{M}_{\alpha \rightarrow \beta}$, for every $g \in \mathcal{M}_\alpha$
  $f \bullet g = f' \bullet g$ implies $f = f'$

## Axioms of Interpretation

Given $\nu : \text{Var} \rightarrow \mathcal{M}$.

- $\llbracket x^\alpha \rrbracket_\nu^\mathcal{M} = \nu(x^\alpha)$
- $\llbracket c^\alpha \rrbracket_\nu^\mathcal{M} = \rho(c^\alpha)$
- $\llbracket \lambda x^\alpha. M \rrbracket_\nu^\mathcal{M} \bullet a = \llbracket M \rrbracket_\nu^\mathcal{M}[x:=a]$
- $\llbracket M_{\alpha \rightarrow \beta} N^\alpha \rrbracket_\nu^\mathcal{M} = \llbracket M \rrbracket_\nu^\mathcal{M} \bullet \llbracket N \rrbracket_\nu^\mathcal{M}$
- $\llbracket Y \rrbracket_\nu^\mathcal{M} \bullet a = a \bullet (\llbracket Y \rrbracket_\nu^\mathcal{M} \bullet a)$
A model $\mathcal{M}$ is $((\mathcal{M}_\alpha)_{\alpha \in \mathcal{T}}, \bullet, \rho)$ where

- for every $f \in \mathcal{M}_{\alpha \rightarrow \beta}$ and $g \in \mathcal{M}_\alpha$, $f \bullet g \in \mathcal{M}_\beta$,
- for every $f, f' \in \mathcal{M}_{\alpha \rightarrow \beta}$, for every $g \in \mathcal{M}_\alpha$ $f \bullet g = f' \bullet g$ implies $f = f'$

### Axioms of Interpretation

Given $\nu : \text{Var} \rightarrow \mathcal{M}$.

- $\llbracket x^\alpha \rrbracket_{\nu} = \nu(x^\alpha)$
- $\llbracket c^\alpha \rrbracket_{\nu} = \rho(c^\alpha)$
- $\llbracket \lambda x^\alpha. M \rrbracket_{\nu} \bullet a = [M]_{\nu[x:=a]}$
- $\llbracket M^{\alpha \rightarrow \beta} N^\alpha \rrbracket_{\nu} = [M]_{\nu} \bullet [N]_{\nu}$
- $\llbracket Y \rrbracket_{\nu} \bullet a = a \bullet ([Y]_{\nu} \bullet a)$
Model checking of higher-order programs and denotational semantics of λ-calculus.

Models of \( \Lambda Y \)

A model \( \mathcal{M} \) is \( ((\mathcal{M}_\alpha)_{\alpha \in \mathcal{T}}, \bullet, \rho) \) where

- for every \( f \in \mathcal{M}_{\alpha \rightarrow \beta} \) and \( g \in \mathcal{M}_\alpha \), \( f \bullet g \in \mathcal{M}_\beta \),
- for every \( f, f' \in \mathcal{M}_{\alpha \rightarrow \beta} \), for every \( g \in \mathcal{M}_\alpha \)
  \( f \bullet g = f' \bullet g \) implies \( f = f' \)

Axioms of Interpretation

Given \( \nu : \text{Var} \rightarrow \mathcal{M} \).

- \( \llbracket x^{\alpha} \rrbracket_{\nu}^{\mathcal{M}} = \nu(x^{\alpha}) \)
- \( \llbracket c^{\alpha} \rrbracket_{\nu}^{\mathcal{M}} = \rho(c^{\alpha}) \)
- \( \llbracket \lambda x^{\alpha}. M \rrbracket_{\nu}^{\mathcal{M}} \bullet a = \llbracket M \rrbracket_{\nu[x:=a]}^{\mathcal{M}} \)
- \( \llbracket M^{\alpha \rightarrow \beta} N^{\alpha} \rrbracket_{\nu}^{\mathcal{M}} = \llbracket M \rrbracket_{\nu}^{\mathcal{M}} \bullet \llbracket N \rrbracket_{\nu}^{\mathcal{M}} \)
- \( \llbracket Y \rrbracket_{\nu}^{\mathcal{M}} \bullet a = a \bullet (\llbracket Y \rrbracket_{\nu}^{\mathcal{M}} \bullet a) \)
Models of $\Lambda Y$

A model $M$ is $((M_\alpha)_{\alpha \in \tau}, \bullet, \rho)$ where

- for every $f \in M_{\alpha \to \beta}$ and $g \in M_{\alpha}$, $f \bullet g \in M_{\beta}$,
- for every $f, f' \in M_{\alpha \to \beta}$, for every $g \in M_{\alpha}$
  $f \bullet g = f' \bullet g$ implies $f = f'$

Axioms of Interpretation

Given $\nu : \text{Var} \to M$.

- $\llbracket x^\alpha \rrbracket_\nu^M = \nu(x^\alpha)$
- $\llbracket c^\alpha \rrbracket_\nu^M = \rho(c^\alpha)$
- $\llbracket \lambda x^\alpha.M \rrbracket_\nu^M \bullet a = \llbracket M \rrbracket_\nu^{M_{\nu[x:=a]}}$
- $\llbracket M^{\alpha \to \beta}N^\alpha \rrbracket_\nu^M = \llbracket M \rrbracket_\nu^M \bullet \llbracket N \rrbracket_\nu^M$
- $\llbracket Y \rrbracket_\nu^M \bullet a = a \bullet (\llbracket Y \rrbracket_\nu^M \bullet a)$
Model checking of higher-order programs and denotational semantics of λ-calculus.

Models of Λ

A model $\mathcal{M}$ is $(\{\mathcal{M}_\alpha\}_{\alpha \in \mathcal{T}}, \bullet, \rho)$ where

- for every $f \in \mathcal{M}_{\alpha \rightarrow \beta}$ and $g \in \mathcal{M}_\alpha$, $f \bullet g \in \mathcal{M}_\beta$, 
- for every $f, f' \in \mathcal{M}_{\alpha \rightarrow \beta}$, for every $g \in \mathcal{M}_\alpha$ $f \bullet g = f' \bullet g$ implies $f = f'$

Axioms of Interpretation

Given $\nu : \text{Var} \rightarrow \mathcal{M}$.

- $\llbracket x^\alpha \rrbracket_\nu = \nu(x^\alpha)$
- $\llbracket c^\alpha \rrbracket_\nu = \rho(c^\alpha)$
- $\llbracket \lambda x^\alpha . M \rrbracket_\nu \bullet a = \llbracket M \rrbracket_\nu[x := a]$,
- $\llbracket M^\alpha \rightarrow \beta N^\alpha \rrbracket_\nu = \llbracket M \rrbracket_\nu \bullet \llbracket N \rrbracket_\nu$
- $\llbracket Y \rrbracket_\nu \bullet a = a \bullet (\llbracket Y \rrbracket_\nu \bullet a)$
Models and recognition

Given a model $\mathcal{M}$, and $A \subseteq \mathcal{M}_\alpha$, $A$ recognizes the language $\{M \mid \llbracket M \rrbracket^\mathcal{M} \in A\}$. 

Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.
Known properties of models

Theorem (Henkin)

The following are equivalent:

- $M =_{\beta\delta\eta} N$
- for all model $M$ and every valuation $\nu$, $\llbracket M \rrbracket^M_{\nu} = \llbracket N \rrbracket^M_{\nu}$

Theorem (Statman 03)

Whether $M =_{\beta\delta\eta} N$ is undecidable.

Models are in general not effective.
The Monotone Model of $\lambda Y$

The **monotone model** over a finite lattice $(\mathcal{P}(X), \subseteq)$ is

$$D_X = \{(D_\alpha, \sqsubseteq_\alpha)\}_{\alpha \in T}, \rho : \text{Cst} \to D_X$$

where

- $D_o = \mathcal{P}(X)$ and $f \sqsubseteq_o g$ iff $f \subseteq g$,
The Monotone Model of $\lambda Y$

The monotone model over a finite lattice $(\mathcal{P}(X), \subseteq)$ is

$$\mathcal{D}_X = \{(D_\alpha, \sqsubseteq_\alpha)\}_{A \in \mathcal{T}, \rho} \quad \rho : \text{Cst} \to \mathcal{D}_X$$

where

- $\mathcal{D}_o = \mathcal{P}(X)$ and $f \sqsubseteq_o g$ iff $f \subseteq g$. 

$\mathcal{P}(X)$
The Monotone Model of $\lambda Y$

The monotone model over a finite lattice $(\mathcal{P}(X), \subseteq)$ is

$$D_X = \left\{ (D_\alpha, \sqsubseteq_\alpha) \mid A \in T, \rho \right\} \quad \rho : \text{Cst} \to D_X$$

where

- $D_o = \mathcal{P}(X)$ and $f \subseteq_o g$ iff $f \subseteq g$,
- $D_{\beta \to \gamma} = [D_\beta \to^m D_\gamma]$  
  $\sqsubseteq_{\beta \to \gamma} =$ pointwise ordering.

\[
\begin{array}{c}
[D_\beta \to^m D_\gamma] \\
\vdots \\
D_\gamma \\
\vdots \\
D_\beta \\
\vdots \\
\mathcal{P}(X) \to^m \mathcal{P}(X) \\
\mathcal{P}(X)
\end{array}
\]
Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

The Monotone Model of $\lambda Y$

The monotone model over a finite lattice $(\mathcal{P}(X), \subseteq)$ is

$$\mathcal{D}_X = \{((\mathcal{D}_\alpha, \sqsubseteq_\alpha))_{A \in T}, \rho \} \quad \rho : \text{Cst} \to \mathcal{D}_X$$

where

- $\mathcal{D}_o = \mathcal{P}(X)$ and $f \sqsubseteq_o g$ iff $f \subseteq g$,
- $\mathcal{D}_{\beta \to \gamma} = [\mathcal{D}_\beta \to_m \mathcal{D}_\gamma]$

$\sqsubseteq_{\beta \to \gamma} = \text{pointwise ordering}.$

$$[Y^\alpha]^{\mathcal{D}_X}(a) = \bigwedge_{n \in \mathbb{N}} a^n(\top^\alpha)$$
Known properties of monotone models

**Theorem (Statman 82)**

The following are equivalent:

- $BT(M) = BT(N)$,
- for every monotone models $\mathcal{M}$ and every valuation $\nu$,
  
  \[
  \llbracket M \rrbracket_{\mathcal{M}}^{\mathcal{M}} = \llbracket N \rrbracket_{\mathcal{M}}^{\mathcal{M}}.
  \]

**Theorem (Loader 00)**

Given a monotone model $\mathcal{M}$ and $f$ in $\mathcal{M}_\alpha$, whether there is $M$ so that $\llbracket M \rrbracket_{\mathcal{M}}^{\mathcal{M}} = f$ is undecidable.
Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

Automata with Trivial Acceptance Condition (TAC) and Böhm trees

Ω-blind TAC: $\Omega$ is accepted by any state
Insightful TAC: $\Omega$ is accepted only by certain states
Ω-blind TAC and monotone models

Theorem

Given $D_X$ and $A$, $M$ is recognized by $A$ iff $BT(M)$ is accepted by a boolean combination of automata with $Ω$-blind TAC.
Ω-blind TAC and monotone models

Theorem

Given $D_X$ and $A$, $M$ is recognized by $A$ iff $BT(M)$ is accepted by a boolean combination of automata with Ω-blind TAC.

Proof $\iff$

$A = (Q, \delta)$

Take the monotone model $M$ so that $M_0 = \mathcal{P}(Q)$ and

$\begin{align*}
\llbracket a \rrbracket(Q_1, Q_2) &= \{ q \mid \exists (q_1, q_2). (q_1, q_2) \in (Q_1 \times Q_2) \cap \delta(a, q) \} \\
\llbracket c \rrbracket &= \delta(c)
\end{align*}$

$q \in \llbracket M \rrbracket^M$ iff $A$ accepts $BT(M)$ from $q$. 

Model checking of higher-order programs and denotational semantics of λ-calculus.
Ω-blind TAC and monotone models

**Theorem**

Given $D_X$ and $A$, $M$ is recognized by $A$ iff $BT(M)$ is accepted by a boolean combination of automata with Ω-blind TAC.

**Proof ⇒**

Given a monotone model $M$ we define $A_M = (M_0, \delta)$ so that:

$\delta(q, a) = \{(q_1, q_2) \mid q \leq \llbracket a \rrbracket(q_1, q_2)\}$

$\delta(c) = \llbracket c \rrbracket.$

$A_M$ accepts $BT(M)$ from $q$ iff $\llbracket M \rrbracket \geq q.$
Beyond monotone models

We are going to construct models for

- Insightful automata
- weak MSO

Important ingredients

- Logical relations
- Galois connections
Logical relations

Lemma (Fundamental Lemma)

If $[Y]^{M_1} \mathcal{R} [Y]^{M_2}$, $[c]^{M_1} \mathcal{R} [c]^{M_2}$, $\nu_1$ and $\nu_2$ so that for every $x$, $\nu_1(x) \mathcal{R} \nu_2(x)$ then for every $M$: $[M]^{\nu_1}_{M_1} \mathcal{R} [M]^{\nu_2}_{M_2}$
Logical relations

Lemma (Fundamental Lemma)

If $[Y]^{\mathcal{M}_1} \mathrel{R} [Y]^{\mathcal{M}_2}$, $[c]^{\mathcal{M}_1} \mathrel{R} [c]^{\mathcal{M}_2}$, $\nu_1$ and $\nu_2$ so that for every $x$, $\nu_1(x) \mathrel{R} \nu_2(x)$ then for every $M$: $[M]^{Y}_\nu_1 \mathrel{R} [M]^{Y}_\nu_2$
Logical relations

Lemma (Fundamental Lemma)
If $[Y]^{M_1} R [Y]^{M_2}$, $[c]^{M_1} R [c]^{M_2}$, $\nu_1$ and $\nu_2$ so that for every $x$, $\nu_1(x) R \nu_2(x)$ then for every $M$: $[M]^{\nu_1}_{M_1} R [M]^{\nu_2}_{M_2}$
Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

Logical relations

Lemma (Fundamental Lemma)
If $[Y]^{M_1} \mathcal{R} [Y]^{M_2}, [c]^{M_1} \mathcal{R} [c]^{M_2}$, $\nu_1$ and $\nu_2$ so that for every $x$, $\nu_1(x) \mathcal{R} \nu_2(x)$ then for every $M$: $[M]^{\nu_1}_{M_1} \mathcal{R} [M]^{\nu_2}_{M_2}$
Logical relations

Lemma (Fundamental Lemma)

If $[Y]_{M_1} \mathcal{R} [Y]_{M_2}$, $[c]_{M_1} \mathcal{R} [c]_{M_2}$, $\nu_1$ and $\nu_2$ so that for every $x$, $
u_1(x) \mathcal{R} \nu_2(x)$ then for every $M$: $[M]_{\nu_1} \mathcal{R} [M]_{\nu_2}$
Model checking of higher-order programs and denotational semantics of λ-calculus.

 Logical relations

\[ M_1 \quad \xrightarrow{R_0} \quad M_2 \]

\[ \xrightarrow{R_\alpha \rightarrow \beta} \]

\[ f \quad \xrightarrow{R_\alpha} \quad g \]

\[ f' \quad \xrightarrow{R_\beta} \quad g' \]

Lemma (Fundamental Lemma)

If \([Y]^{M_1} \xrightarrow{R} [Y]^{M_2}\), \([c]^{M_1} \xrightarrow{R} [c]^{M_2}\), \(\nu_1\) and \(\nu_2\) so that for every \(x\), \(\nu_1(x) \xrightarrow{R} \nu_2(x)\) then for every \(M: [M]^{M_1} \xrightarrow{R} [M]^{M_2}\).
Logical relations

Lemma (Fundamental Lemma)

If $\llbracket Y \rrbracket^M_1 \mathcal{R} \llbracket Y \rrbracket^M_2$, $\llbracket c \rrbracket^M_1 \mathcal{R} \llbracket c \rrbracket^M_2$, $\nu_1$ and $\nu_2$ so that for every $x$, $\nu_1(x) \mathcal{R} \nu_2(x)$ then for every $M$: $\llbracket M \rrbracket^M_1 \mathcal{R} \llbracket M \rrbracket^M_2$
A model for detecting $\Omega$

The monotone model $D$ where $Y$ is interpreted as least fixpoint and which is generated by the lattice:

$$
\begin{array}{c}
\top \\
\downarrow \\
\bot
\end{array}
$$

$$\llbracket a \rrbracket(x, y) = \top$$

is so that $BT(M) = \Omega$ iff $\llbracket M \rrbracket^D = \bot$. 
Constructing a model for insightful automata

We want to accept $\Omega$ only with state $q_0$

\[
\begin{array}{c}
\top \\
\downarrow \\
\bot \\
\end{array}
\quad
\begin{array}{c}
\{q_0; q_1\} \\
\{q_0\} \\
\{q_1\} \\
\emptyset \\
\end{array}
\]

\[
[a](x, y) = \top \quad [a](Q, Q') = \{q \mid \delta(a, q) \subseteq Q \times Q'\}
\]

-we need to interpret $\Omega$ as $(\bot, \{q_0\})$,

-make that compatible with higher-order and $\Psi$ interpretation
Constructing a model for insightful automata

We want to accept $\Omega$ only with state $q_0$

\[
\begin{array}{c}
\top \\
\downarrow \\
\bot
\end{array} \quad \downarrow \quad \begin{cases}
\{q_0; q_1\} \\
\{q_0\} \\
\emptyset \\
\{q_1\}
\end{cases}
\]

\[
\llbracket a \rrbracket(x, y) = \top \\
\llbracket a \rrbracket(Q, Q') = \{q \mid \delta(a, q) \subseteq Q \times Q'\}
\]

- we need to interpret $\Omega$ as $(\bot, \{q_0\})$,
- make that compatible with higher-order and $\mathcal{Y}$ interpretation

\[
\begin{array}{c}
(\top, \{q_0; q_1\}) \\
\downarrow \\
(\top, \{q_0\}) \quad (\top, \{q_1\})
\end{array} \quad \downarrow \quad \begin{cases}
(\bot, \{q_0\}) \\
(\top, \emptyset)
\end{cases}
\]
Model checking of higher-order programs and denotational semantics of \( \lambda \)-calculus.

**Constructing a model for insightful automata**

We want to accept \( \Omega \) only with state \( q_0 \)

\[
\begin{array}{c}
\top \\
\downarrow \\
\bot
\end{array}
\quad
\begin{array}{c}
\{q_0; q_1\} \\
\{q_0\} \\
\{q_1\} \\
\emptyset
\end{array}
\]

\[
[a](x, y) = \top \\
[a](Q, Q') = \{q | \delta(a, q) \subseteq Q \times Q'\}
\]

- we need to interpret \( \Omega \) as \((\bot, \{q_0\})\),
- make that compatible with higher-order and \( Y \) interpretation
Constructing a model for insightful automata

We want to accept $\Omega$ only with state $q_0$

$$
\begin{array}{c}
\top \\
\downarrow \\
\bot
\end{array}
\quad
\begin{array}{c}
\{q_0; q_1\} \\
\downarrow \\
\{q_0\} \\
\downarrow \\
\emptyset
\end{array}
$$

$\llbracket a \rrbracket(x, y) = \top$

$\llbracket a \rrbracket(Q, Q') = \{q \mid \delta(a, q) \subseteq Q \times Q'\}$

- we need to interpret $\Omega$ as $(\bot, \{q_0\})$, 
- make that compatible with higher-order and $Y$ interpretation

$(\top, \{q_0; q_1\})$

$(\top, \{q_0\})$

$(\top, \{q_1\})$

$(\bot, \{q_0\})$

$(\top, \emptyset)$
Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

Going higher-order with a logical relation

\[ L_0 = \{ ((d, P), d) \mid (d, P) \in K_0 \}, \]
\[ K_{\alpha \rightarrow \beta} = \{ f \in \text{mon}[K_{\alpha} \rightarrow K_{\beta}] \mid \exists d \in D_{\alpha \rightarrow \beta}. \forall (g, e) \in L_{\alpha}. (f(g), d(e)) \in L_{\beta} \}, \]
\[ L_{\alpha \rightarrow \beta} = \{ (f, d) \in K_{\alpha \rightarrow \beta} \times D_{\alpha \rightarrow \beta} \mid \forall (g, e) \in L_{\alpha}. (f(g), d(e)) \in L_{\beta} \}. \]
Model checking of higher-order programs and denotational semantics of λ-calculus.

Going higher-order with a logical relation

\[
\begin{align*}
\top & \quad (\top, \{q_0; q_1\}) \\
\bot & \quad (\top, \{q_0\}) \quad (\top, \{q_1\}) \\
& \quad (\bot, \{q_0\}) \quad (\top, \emptyset)
\end{align*}
\]

\[
\begin{align*}
\mathcal{L}_0 &= \{((d, P), d) \mid (d, P) \in \mathcal{K}_0\}, \\
\mathcal{K}_{\alpha \rightarrow \beta} &= \{f \in \text{mon}[\mathcal{K}_\alpha \rightarrow \mathcal{K}_\beta] \mid \exists d \in D_{\alpha \rightarrow \beta}. \forall (g, e) \in \mathcal{L}_\alpha. (f(g), d(e)) \in \mathcal{L}_\beta\}, \\
\mathcal{L}_{\alpha \rightarrow \beta} &= \{(f, d) \in \mathcal{K}_{\alpha \rightarrow \beta} \times D_{\alpha \rightarrow \beta} \mid \forall (g, e) \in \mathcal{L}_\alpha. (f(g), d(e)) \in \mathcal{L}_\beta\}.
\end{align*}
\]
Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.

Going higher-order with a logical relation

$$
\begin{align*}
\top & \to \bot
\end{align*}
$$

$$
\begin{align*}
(\top, \{q_0; q_1\}) & \to (\top, \{q_0\}) & (\top, \{q_1\}) \\
(\top, \{q_0\}) & \to (\bot, \{q_0\}) & (\top, \emptyset)
\end{align*}
$$

$$
\begin{align*}
\mathcal{L}_0 &= \{((d, P), d) | (d, P) \in K_0\}, \\
\mathcal{K}_{\alpha \to \beta} &= \{f \in \text{mon}[K_\alpha \to K_\beta] | \exists d \in D_{\alpha \to \beta}. \forall (g, e) \in L_\alpha. (f(g), d(e)) \in L_\beta\}, \\
\mathcal{L}_{\alpha \to \beta} &= \{(f, d) \in \mathcal{K}_{\alpha \to \beta} \times D_{\alpha \to \beta} | \forall (g, e) \in L_\alpha. (f(g), d(e)) \in L_\beta\}.
\end{align*}
$$
Going higher-order with a logical relation

\[ \begin{align*}
\top &\to \{q_0; q_1\} \\
\top &\to \{q_0\} \quad \top &\to \{q_1\} \\
\bot &\to \{q_0\} \quad & (\top, \emptyset)
\end{align*} \]

\[ \begin{align*}
\mathcal{L}_0 &= \{((d, P), d) \mid (d, P) \in \mathcal{K}_0\}, \\
\mathcal{K}_{\alpha \to \beta} &= \{f \in \text{mon}[\mathcal{K}_\alpha \to \mathcal{K}_\beta] \mid \exists d \in \mathcal{D}_{\alpha \to \beta}. \forall (g, e) \in \mathcal{L}_\alpha. (f(g), d(e)) \in \mathcal{L}_\beta\}, \\
\mathcal{L}_{\alpha \to \beta} &= \{(f, d) \in \mathcal{K}_{\alpha \to \beta} \times \mathcal{D}_{\alpha \to \beta} \mid \forall (g, e) \in \mathcal{L}_\alpha. (f(g), d(e)) \in \mathcal{L}_\beta\}.
\end{align*} \]
A Galois connection

The logical relation $\mathcal{L}$ induces two functors:

- for every $f$ in $\mathcal{K}_\alpha$ there is a unique $\bar{f}$ in $\mathcal{D}_\alpha$ so that $f \mathcal{L}_\alpha \bar{f}$,
- for $d$ in $\mathcal{D}_\alpha$, let $d^\uparrow = \bigvee \{ f \mid d \mathcal{L}_\alpha f \}$.

Functoriality

- $f(g) = \bar{f}(\bar{g})$ and $f \leq g$ implies $\bar{f} \leq \bar{g}$,
- $d(e)^\uparrow = d^\uparrow(e^\uparrow)$, $d \mathcal{L} d^\uparrow$ and $d \leq e$ implies $d^\uparrow \leq e^\uparrow$.

Galois connection

- $\bar{f} \leq d$ iff $f \leq d^\uparrow$,
- in particular $d^\uparrow = d$
A Galois connection

The logical relation $\mathcal{L}$ induces two functors:

- for every $f$ in $\mathcal{K}_\alpha$ there is a unique $\bar{f}$ in $\mathcal{D}_\alpha$ so that $f \mathcal{L}_\alpha \bar{f}$,
- for $d$ in $\mathcal{D}_\alpha$, let $d^\uparrow = \bigvee \{ f \mid d \mathcal{L}_\alpha f \}$.

Functoriality

- $\bar{f}(g) = \bar{f}(g)$ and $f \leq g$ implies $\bar{f} \leq \bar{g}$,
- $d(e)^\uparrow = d^\uparrow(e^\uparrow)$, $d \mathcal{L} d^\uparrow$ and $d \leq e$ implies $d^\uparrow \leq e^\uparrow$.

Gallois connection

- $\bar{f} \leq d$ iff $f \leq d^\uparrow$,
- in particular $d^\uparrow = d$
A Galois connection

The logical relation $\mathcal{L}$ induces two functors:

- for every $f$ in $\mathcal{K}_\alpha$ there is a unique $\overline{f}$ in $\mathcal{D}_\alpha$ so that $f \mathcal{L}_\alpha \overline{f}$,
- for $d$ in $\mathcal{D}_\alpha$, let $d^\uparrow = \bigvee \{ f \mid d \mathcal{L}_\alpha f \}$.

Functoriality

- $\overline{f(g)} = \overline{f(\overline{g})}$ and $f \leq g$ implies $\overline{f} \leq \overline{g}$,
- $d(e)^\uparrow = d^\uparrow(e^\uparrow)$, $d \mathcal{L} d^\uparrow$ and $d \leq e$ implies $d^\uparrow \leq e^\uparrow$.

Galois connection

- $\overline{f} \leq d$ iff $f \leq d^\uparrow$,
- in particular $\overline{d^\uparrow} = d$
Fixpoint

Fixpoint definition

For $f \in \mathcal{K}_{\alpha \rightarrow \alpha}$, we let $\text{Fix}_\alpha(f) = \bigwedge_{n \in \mathbb{N}} f^n(\text{fix}_\alpha(f)^\uparrow)$ with $\text{fix}_\alpha(d) = \bigvee_{n \in \mathbb{N}} d^n(\bot_\alpha)$

Lemma ($\mathcal{K}$ is a model of $\Lambda Y$)

$\text{Fix}_\alpha$ is a fixpoint and is in $\mathcal{K}_{(\alpha \rightarrow \alpha) \rightarrow \alpha}$. 
Fixpoint

Fixpoint definition

For $f \in \mathcal{K}_{\alpha \rightarrow \alpha}$, we let $\text{Fix}_\alpha(f) = \bigwedge_{n \in \mathbb{N}} f^n(\text{fix}_\alpha(f)^\uparrow)$ with $\text{fix}_\alpha(d) = \bigvee_{n \in \mathbb{N}} d^n(\perp_{\alpha})$

Lemma ($\mathcal{K}$ is a model of $\Lambda Y$)

$\text{Fix}_\alpha$ is a fixpoint and is in $\mathcal{K}_{(\alpha \rightarrow \alpha) \rightarrow \alpha}$.

Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.
$\mathcal{K}$ does the job

**Theorem**

For a given insightful automaton $\mathcal{A}$, the construction of $\mathcal{K}$ for $\mathcal{A}$, is so that if $[M]^\mathcal{K} = (d, Q)$, $BT(M)$ is accepted by $\mathcal{A}$ in state $q$ iff $q \in Q$. 
Weak MSO and weak parity automata

A weak parity automaton is $A = (Q, \delta, rk)$ so that:

$rk : Q \mapsto \mathbb{N}, (q_1, q_2) \in \delta(a, q) \text{ implies } rk(q_1) \leq rk(q) \text{ and } rk(q_2) \leq rk(q)$. 

$A$ accepts a tree $t$ from state $q$ if there is a run satisfying the usual parity condition.
Idea of the construction

- We construct the model by induction on the parity,
- At each step the model $\mathcal{M}_k$ at base type is $\mathcal{P}(Q_k)$ where $Q_k = \{ q \in Q \mid \text{rk}(q) \leq k \}$,
- and $q \in \llbracket M \rrbracket^{\mathcal{M}_k}$ iff $A$ accepts $BT(M)$ from $q$. 
Basic block: domain extension

Given finite sets $X_1$ and $X_2$, so that $X_1 \subseteq X_2$, a model $\mathcal{M}_1$ so that $\mathcal{M}_1^0 = \mathcal{P}(X_1)$, we define $\mathcal{M}_2 = \text{ext}(\mathcal{M}_1, X_2)$ as:

- $\mathcal{L}^0 = \{(Q, P) \mid P = Q \cap X_1\}$
- $\mathcal{M}_2^0 = \mathcal{P}(X_2)$
- $\mathcal{M}_{2}^{\alpha \rightarrow \beta} = \{f \in \text{mon}[\mathcal{M}_2^\alpha \mapsto \mathcal{M}_2^\beta] \mid \exists d \in \mathcal{M}_1^\alpha. \forall (g, e) \in \mathcal{L}_\alpha^\beta. (f(g), d(e)) \in \mathcal{L}_\beta^\beta\}$
- $\mathcal{L}_{\alpha \rightarrow \beta} = \{(f, d) \mid \forall (g, e) \in \mathcal{L}_\alpha^\beta. (f(g), d(e)) \in \mathcal{L}_\beta^\beta\}$
Two Galois connections

\[ \mathcal{L} \text{ induces three functors:} \]

1. for every \( f \) in \( \mathcal{M}_2^\alpha \) there is a unique \( \bar{f} \) in \( \mathcal{M}_1^\alpha \) so that \( f \mathcal{L}^\alpha \bar{f}, \)
2. for \( d \) in \( \mathcal{M}_1^\alpha \), let \( d^\uparrow = \bigvee \{ f \mid d \mathcal{L}^\alpha f \} \),
3. for \( d \) in \( \mathcal{M}_1^\alpha \), let \( d^\downarrow = \bigwedge \{ f \mid d \mathcal{L}^\alpha f \} \).

Functoriality

1. \( \bar{f} (g) = \bar{f} (\bar{g}) \), and \( f \leq g \) implies \( \bar{f} \leq \bar{g} \),
2. \( d(e)^\uparrow = d^\uparrow (e^\uparrow) \), \( d \mathcal{L} d^\uparrow \) and \( d \leq e \) implies \( d^\uparrow \leq e^\uparrow \),
3. \( d(e)^\downarrow = d^\downarrow (e^\downarrow) \), \( d \mathcal{L} d^\downarrow \) and \( d \leq e \) implies \( d^\downarrow \leq e^\downarrow \).

Galois connections

1. \( \bar{f} \leq d \) iff \( f \leq d^\uparrow \),
2. \( d \leq \bar{f} \) iff \( d^\downarrow \leq f \),
3. in particular \( d = \bar{d}^\uparrow = \bar{d}^\downarrow \).
Two Galois connections

\( \mathcal{L} \) induces three functors:

- for every \( f \) in \( \mathcal{M}_2^\alpha \) there is a unique \( \overline{f} \) in \( \mathcal{M}_1^\alpha \) so that \( f \mathcal{L}^\alpha \overline{f} \),
- for \( d \) in \( \mathcal{M}_1^\alpha \), let \( d^\uparrow = \bigvee \{ f \mid d \mathcal{L}^\alpha f \} \),
- for \( d \) in \( \mathcal{M}_1^\alpha \), let \( d^\downarrow = \bigwedge \{ f \mid d \mathcal{L}^\alpha f \} \).

**Functoriality**

- \( \overline{f(g)} = \overline{f(g)} \), and \( f \leq g \) implies \( \overline{f} \leq \overline{g} \),
- \( d(e)^\uparrow = d^\uparrow(e^\uparrow) \), \( d \mathcal{L} d^\uparrow \) and \( d \leq e \) implies \( d^\uparrow \leq e^\uparrow \),
- \( d(e)^\downarrow = d^\downarrow(e^\downarrow) \), \( d \mathcal{L} d^\downarrow \) and \( d \leq e \) implies \( d^\downarrow \leq e^\downarrow \).

**Galois connections**

- \( \overline{f} \leq d \) iff \( f \leq d^\uparrow \),
- \( d \leq \overline{f} \) iff \( d^\downarrow \leq f \),
- in particular \( d = d^\uparrow = d^\downarrow \).
Two Galois connections

\( \mathcal{L} \) induces three functors:

- for every \( f \) in \( M_2^\alpha \) there is a unique \( \bar{f} \) in \( M_1^\alpha \) so that \( f \mathcal{L}^\alpha \bar{f} \),
- for \( d \) in \( M_1^\alpha \), let \( d^{\uparrow} = \bigvee \{ f \mid d \mathcal{L}^\alpha f \} \),
- for \( d \) in \( M_1^\alpha \), let \( d^{\downarrow} = \bigwedge \{ f \mid d \mathcal{L}^\alpha f \} \).

Functoriality

- \( \bar{f(g)} = \bar{f}(\bar{g}) \), and \( f \leq g \) implies \( \bar{f} \leq \bar{g} \),
- \( d(e)^{\uparrow} = d^{\uparrow}(e^{\uparrow}) \), \( d \mathcal{L} d^{\uparrow} \) and \( d \leq e \) implies \( d^{\uparrow} \leq e^{\uparrow} \),
- \( d(e)^{\downarrow} = d^{\downarrow}(e^{\downarrow}) \), \( d \mathcal{L} d^{\downarrow} \) and \( d \leq e \) implies \( d^{\downarrow} \leq e^{\downarrow} \).

Galois connections

- \( \bar{f} \leq d \) iff \( f \leq d^{\uparrow} \),
- \( d \leq \bar{f} \) iff \( d^{\downarrow} \leq f \),
- in particular \( d = d^{\uparrow} = d^{\downarrow} \).
Two possible fixpoints

Fixpoints definition

If $\text{fix}_\alpha$ is the interpretation of the fixpoint at type $\alpha$ in $\mathcal{M}_1$, given $f \in \mathcal{M}_2^{\alpha \to \alpha}$ we let:

- $\text{Fix}_0^\alpha(f) = \bigwedge_{n \in \mathbb{N}} f^n(\text{fix}_\alpha(f)^\uparrow)$
- $\text{Fix}_1^\alpha(f) = \bigvee_{n \in \mathbb{N}} f^n(\text{fix}_\alpha(f)^\downarrow)$

Lemma ($\mathcal{M}_2$ is a model of $\Lambda Y$)

$\text{Fix}_0^\alpha$ and $\text{Fix}_1^\alpha$ are fixpoints and are in $\mathcal{M}_2^{(\alpha \to \alpha) \to \alpha}$. 

Model checking of higher-order programs and denotational semantics of $\lambda$-calculus.
Two possible fixpoints

Fixpoints definition

If $\text{fix}_\alpha$ is the interpretation of the fixpoint at type $\alpha$ in $\mathcal{M}_1$, given $f \in \mathcal{M}_2^{\alpha \rightarrow \alpha}$ we let:

$\begin{align*}
\text{Fix}_\alpha^0(f) &= \bigwedge_{n \in \mathbb{N}} f^n(\text{fix}_\alpha(f)\uparrow) \\
\text{Fix}_\alpha^1(f) &= \bigvee_{n \in \mathbb{N}} f^n(\text{fix}_\alpha(f)\downarrow)
\end{align*}$

Lemma ($\mathcal{M}_2$ is a model of $\Lambda Y$)

$\text{Fix}_\alpha^0$ and $\text{Fix}_\alpha^1$ are fixpoints and are in $\mathcal{M}_2^{(\alpha \rightarrow \alpha) \rightarrow \alpha}$.
The model for wMSO

- We let $\mathcal{M}(0)$ be the monotone model generated by $\mathcal{P}(Q_0)$ where $Q_0 = \{ q \in Q \mid \text{rk}(q) = 0 \}$.
- $\mathcal{M}(k + 1) = \text{ext}(\mathcal{M}_k, Q_{k+1})$ where $Q_{k+1} = \{ q \in Q \mid \text{rk}(q) \leq k + 1 \}$, and $\lfloor Y \rfloor^{\mathcal{M}(k+1)} = \text{Fix}^p$ where $p \equiv k + 1[2]$.

Theorem

Given $q \in Q_k$, the following are equivalent:

- $q \in \lfloor \mathcal{M} \rfloor^\mathcal{M}_k$,
- $A$ accepts $BT(M)$ from the state $q$. 