Program equivalence with names

Nikos Tzevelekos

Oxford University Computing Laboratory

The nu-calculus

MFCS'93

Observable Properties of Higher Order Functions that Dynamically Create Local Names, or: What's *new*?

Andrew M. Pitts* and Ian D. B. Stark**

University of Cambridge Computer Laboratory,
Pembroke Street, Cambridge CB2 3QG, England

Abstract. The research reported in this paper is concerned with the problem of reasoning about properties of higher order functions involving state. It is motivated by the desire to identify what, if any, are the difficulties

The nu-calculus

Types

$$
\begin{array}{rl}
T ::= & \texttt{Name} \\
\mid & \texttt{Bool} \\
\mid & T \to T
\end{array}
$$

Terms

$$
\begin{array}{rll}
t ::= & x & (x \in \mathbb{V}) \\
\mid & a & (a \in \mathbb{A}) \\
\mid & \nu a.t & \\
\mid & t == t & \\
\mid & \texttt{true} & \\
\mid & \texttt{false} & \\
\mid & \texttt{if } t \texttt{ then } t \texttt{ else } t & \\
\mid & \lambda x.t & \\
\mid & tt &
\end{array}
$$

The nu-calculus

Types

$$
\begin{array}{rl}
T ::= & \texttt{Name} \\
\mid & \texttt{Bool} \\
\mid & T \to T
\end{array}
$$

Terms

$$
\begin{array}{rll}
t ::= & x & (x \in \mathbb{V}) \\
\mid & a & (a \in \mathbb{A}) \\
\mid & \nu a.t & \\
\mid & t == t & \\
\mid & \texttt{true} & \\
\mid & \texttt{false} & \\
\mid & \texttt{if } t \texttt{ then } t \texttt{ else } t & \\
\mid & \lambda x.t & \\
\mid & tt &
\end{array}
$$

Variables ≠ Names

name binder

free names: $\mathsf{fn}(t)$

variable binder

Some typing rules

$$\frac{(x : T) \in \Gamma}{S; \Gamma \vdash x : T} \qquad \frac{a \in S}{S; \Gamma \vdash a : \texttt{Name}}$$

$$\frac{S; \Gamma, x : T \vdash t : T'}{S; \Gamma \vdash \lambda x.t : T \to T'} \qquad \frac{S \uplus \{a\}; \Gamma \vdash t : T}{S; \Gamma \vdash \nu a.t : T}$$

$$\frac{S; \Gamma \vdash t_1 : \texttt{Name} \quad S; \Gamma \vdash t_2 : \texttt{Name}}{S; \Gamma \vdash t_1 == t_2 : \texttt{Bool}}$$

Reduction

$$(S, t) \longrightarrow (S', t')$$

- S and S' are finite subsets of $\mathbb{A}$
- $\mathsf{fn}(t) \subseteq S\,, \quad \mathsf{fn}(t') \subseteq S'$
- t and t' are general (open) terms

Reduction rules

$$v ::= x \mid a \mid \texttt{true} \mid \texttt{false} \mid \lambda x.t$$

$$(S, (\lambda x.t)v) \to (S, t[v/x])$$

$$(S, \nu a.t) \to (S \uplus \{a\}, t) \qquad \boxed{a \notin S}$$

Reduction rules

$$v ::= \; x \mid a \mid \texttt{true} \mid \texttt{false} \mid \lambda x.t$$

$$(S, (\lambda x.t)v) \longrightarrow (S, t[v/x])$$

$$(S, \nu a.t) \longrightarrow (S \uplus \{a\}, t) \qquad \boxed{a \notin S}$$

$$(S, \nu a.t) \longrightarrow (S \uplus \{a'\}, t[a'/a]) \qquad \boxed{a' \notin S \cup \mathsf{fn}(\nu a.t)}$$

Reduction rules

$$v ::= \; x \mid a \mid \texttt{true} \mid \texttt{false} \mid \lambda x.t$$

$$(S, (\lambda x.t)v) \to (S, t[v/x])$$

$$(S, \nu a.t) \to (S \uplus \{a\}, t) \qquad \boxed{a \notin S}$$

$$(S, a == a) \to (S, \texttt{true})$$

$$(S, a == a') \to (S, \texttt{false}) \qquad \boxed{a \neq a'}$$

Reduction rules

$$v ::= \; x \mid a \mid \texttt{true} \mid \texttt{false} \mid \lambda x.t$$

$$(S, (\lambda x.t)v) \to (S, t[v/x])$$

$$(S, \nu a.t) \to (S \uplus \{a\}, t) \qquad \boxed{a \notin S}$$

$$\frac{(S, t) \to (S', t')}{(S, E[t]) \to (S', E[t'])}$$

$$E ::= \; \_t \;\mid\; (\lambda x.t)\_ \;\mid\; \_ == t \;\mid\; a == \_ \;\mid\; \cdots$$

Evaluation and contextual equivalence

- *Thm*: Reduction is SN to UNF modulo fresh names.
- Evaluation: $(S, t) \Downarrow v \quad \equiv \quad \exists S'.\, (S, t) \twoheadrightarrow (S', v)$

Evaluation and contextual equivalence

- *Thm*: Reduction is SN to UNF modulo fresh names.
- Evaluation: $(S, t) \Downarrow v \quad \equiv \quad \exists S'.\, (S, t) \twoheadrightarrow (S', v)$

$$S; \Gamma \vdash t_1 \cong t_2 : T$$

- for all closing contexts $C[-] : \texttt{Bool}$

$$(S, C[t_1]) \Downarrow \texttt{true} \iff (S, C[t_2]) \Downarrow \texttt{true}$$

Sample (in)equivalences

$$\nu a.t \;\cong\; t \qquad \boxed{a \notin \mathsf{fn}(t)} \tag{1}$$

$$\nu a.\nu a'.t \;\cong\; \nu a'.\nu a.t \tag{2}$$

$$\nu a.\lambda x.\, a \;\ncong\; \lambda x.\nu a.\, a : \texttt{Bool} \to \texttt{Name} \tag{3}$$

$$(\lambda x,y.\, s)\, t_1\, t_2 \;\cong\; (\lambda y,x.\, s)\, t_2\, t_1 \tag{4}$$

Sample (in)equivalences

$$\nu a.t \;\cong\; t \qquad \boxed{a \notin \mathsf{fn}(t)} \tag{1}$$

$$\nu a.\nu a'.t \;\cong\; \nu a'.\nu a.t \tag{2}$$

$$\nu a.\lambda x.\, a \;\ncong\; \lambda x.\nu a.\, a : \mathtt{Bool} \to \mathtt{Name} \tag{3}$$

$$(\lambda x,y.\, s)\, t_1\, t_2 \;\cong\; (\lambda y,x.\, s)\, t_2\, t_1 \tag{4}$$

$$\nu a.\lambda x.\, (x == a) \;\cong\; \lambda x.\, \mathtt{false} : \mathtt{Name} \to \mathtt{Bool} \tag{5}$$

$$\nu a_1,a_2.\, \lambda f.\, (f\, a_1 = f\, a_2) \;\cong\; \lambda f.\, \mathtt{true} : (\mathtt{N} \to \mathtt{B}) \to \mathtt{B} \tag{6}$$

Proofs of (6)

- [Stark'94]: via Predicated Logical Relations
- [AGMOS'04]: Nominal Game Semantics
- [Benton&Koutavas'08]: Environmental Bisimulations

Stark's proof

Names and Higher-Order Functions

Ian David Bede Stark
Queens' College

Stark's proof

Names and Higher-Order Functions

Logical Relations

- Define *spans*: $R : S_1 \rightleftharpoons S_2$
- Extend to: $R_T \subseteq Val_T(S_1) \times Val_T(S_2)$

Logical Relations

- Define *spans*: $R : S_1 \rightleftharpoons S_2$
- Extend to: $R_T \subseteq Val_T(S_1) \times Val_T(S_2)$

$$\mathtt{b}_1 \; R_{\mathtt{Bool}} \; \mathtt{b}_2 \;\equiv\; \mathtt{b}_1 = \mathtt{b}_2$$

$$a_1 \; R_{\mathtt{Name}} \; a_2 \;\equiv\; a_1 \; R \; a_2$$

$$(\lambda x.t_1) \; R_{T \to T'} \; (\lambda x.t_2) \;\equiv$$
$$\forall R' : S_1' \rightleftharpoons S_2', \; v_i \in Val(S_i \uplus S_i').$$
$$v_1 \; (R \uplus R')_T \; v_2 \implies t_1[v_1/x] \; (R \uplus R')^*_{T'} \; t_2[v_2/x]$$

Logical Relations

- Define *spans*: $R : S_1 \rightleftharpoons S_2$
- Extend to: $R_T \subseteq Val_T(S_1) \times Val_T(S_2)$

$\mathtt{b}_1 \; R_{\mathtt{Bo}}$

$a_1 \; R_{\mathtt{Na}}$

$$M_1 \; R_T^* \; M_2 \;\equiv\; \exists R' : S_1' \rightleftharpoons S_2' \,,\; v_i \in Val(S_i \uplus S_i').\; (S_i, M_i) \twoheadrightarrow (S_i', v_i) \;\wedge\; v_1 \; (R \uplus R')_T \; v_2$$

$$(\lambda x.t_1) \; R_{T \to T'} \; (\lambda x.t_2) \;\equiv\; \forall R' : S_1' \rightleftharpoons S_2' \,,\; v_i \in Val(S_i \uplus S_i').\; v_1 \; (R \uplus R')_T \; v_2 \implies t_1[v_1/x] \; (R \uplus R')_{T'}^* \; t_2[v_2/x]$$

Complete at order 1

- For all T: $\quad t_1 \; \mathsf{id}^*_T \; t_2 \implies t_1 \cong t_2 : T$

- For T of order 1 the converse holds too.

$$\nu a.\lambda x.\,(x == a) \cong \lambda x.\,\texttt{false} : \texttt{Name} \to \texttt{Bool} \qquad (5)$$ ✔

$$\nu a_1,a_2.\,\lambda f.\,(f\,a_1 = f\,a_2) \cong \lambda f.\,\texttt{true} : (\texttt{N} \to \texttt{B}) \to \texttt{B} \qquad (6)$$ ✘

Predicated Logical relations

$$\hat{R} = (S_1 \overset{R_1}{\rightleftharpoons} S_1 \overset{R}{\rightleftharpoons} S_2 \overset{R_2}{\rightleftharpoons} S_2)$$

- *Augmented spans*: $\hat{R} : S_1 \rightleftharpoons S_2$

$$\mathtt{b}_1 \; \hat{R}_{\mathtt{Bool}} \; \mathtt{b}_2 \;\equiv\; \mathtt{b}_1 = \mathtt{b}_2$$

$$a_1 \; \hat{R}_{\mathtt{Name}} \; a_2 \;\equiv\; a_1 \; R_1 \; a_1 \; R \; a_2 \; R_2 \; a_2$$

Predicated Logical relations

$$\hat{R} = (S_1 \overset{R_1}{\rightleftharpoons} S_1 \overset{R}{\rightleftharpoons} S_2 \overset{R_2}{\rightleftharpoons} S_2)$$

- *Augmented spans*: $\hat{R} : S_1 \rightleftharpoons S_2$

$$\mathtt{b}_1 \ \hat{R}_{\mathtt{Bool}} \ \mathtt{b}_2 \ \equiv \ \mathtt{b}_1 = \mathtt{b}_2$$

$$a_1 \ \hat{R}_{\mathtt{Name}} \ a_2 \ \equiv \ a_1 \ R_1 \ a_1 \ R \ a_2 \ R_2 \ a_2$$

$$(\lambda x.t_1) \ \hat{R}_{T \to T'} \ (\lambda x.t_2) \ \equiv$$

$$(\lambda x.t_1) \ (R_1)_{T \to T'} \ (\lambda x.t_1) \wedge (\lambda x.t_2) \ (R_2)_{T \to T'} \ (\lambda x.t_2)$$

$$\wedge \forall \hat{R}' : S_1' \rightleftharpoons S_2', \ v_1 \in \mathit{Val}(S_1 \uplus S_1'), \ v_2 \in \mathit{Val}(S_2 \uplus S_2').$$

$$v_1 \ (\hat{R} \uplus \hat{R}')_T \ v_2 \implies t_1[v_1/x] \ (\hat{R} \uplus \hat{R}')^*_{T'} \ t_2[v_2/x]$$

Proof of (6)

- For all T: $\quad t_1 \mathrel{\hat{\mathsf{id}}^*_T} t_2 \implies t_1 \cong t_2 : T$

- Take $\hat{R} : \{a_1, a_2\} \rightleftharpoons \emptyset$

  with $R = R_2 = \emptyset\,, \quad R_1 = \{(a_1, a_2), (a_2, a_1)\}$

Proof of (6)

- For all T: $\quad t_1 \mathrel{\hat{\mathsf{id}}^*_T} t_2 \implies t_1 \cong t_2 : T$

- Take $\hat{R} : \{a_1, a_2\} \rightleftharpoons \emptyset$
  with $R = R_2 = \emptyset\,, \quad R_1 = \{(a_1, a_2), (a_2, a_1)\}$

- $(\lambda f.\,(f a_1 = f a_2))\ \hat{R}_{(\mathtt{N}\to\mathtt{B})\to\mathtt{B}}\ (\lambda f.\,\mathtt{true})$

- $\nu a_1,a_2.\,(\lambda f.\,(f a_1 = f a_2))\ \hat{\mathsf{id}}^*_{(\mathtt{N}\to\mathtt{B})\to\mathtt{B}}\ (\lambda f.\,\mathtt{true})$

~~

- Simple robust method
- Logical relations are derivable from spans

- Complete only up to order 1
- Predicated version good for (6); ow?

AGMOS' Proof

Nominal Games and Full Abstraction for the Nu-Calculus

S. Abramsky, Oxford University
D. R. Ghica, Oxford University
A. S. Murawski, Oxford University
C.-H. L. Ong, Oxford University
I. D. B. Stark, Edinburgh University

Abstract

We introduce nominal games for modelling programming languages with dynamically generated local names, as exemplified by Pitts and Stark's nu-calculus. Inspired by Pitts and Gabbay's recent work on nominal sets, we construct arenas and strategies in the world (or topos) of Fraenkel-Mostowski sets (or simply FM-sets). We fix an infinite set $\mathcal{N}$ of names to be the "atoms" of the FM-theory, and interpret the type ν of names as the flat arena whose move-set is $\mathcal{N}$. This approach leads to a clean and precise treatment of fresh names and standard game constructions to capture notions of scope, privacy and sharing.

Game semantics has emerged as a powerful paradigm for giving accurate semantics for a wide spectrum of programming languages. Fully abstract[1] game models have now been constructed for languages with references of various kinds, such as statically-scoped assignable variables [2] and general references [1]. Game models that have been constructed for these languages (with the notable exception of [8]) follow Reynolds in viewing a reference type $\mathtt{ref}[A]$ as a product of "read method" and "write method". As O'Hearn has pointed out, using this interpretation, functoriality of the $\mathtt{ref}[-]$ type constructor requires the presence of

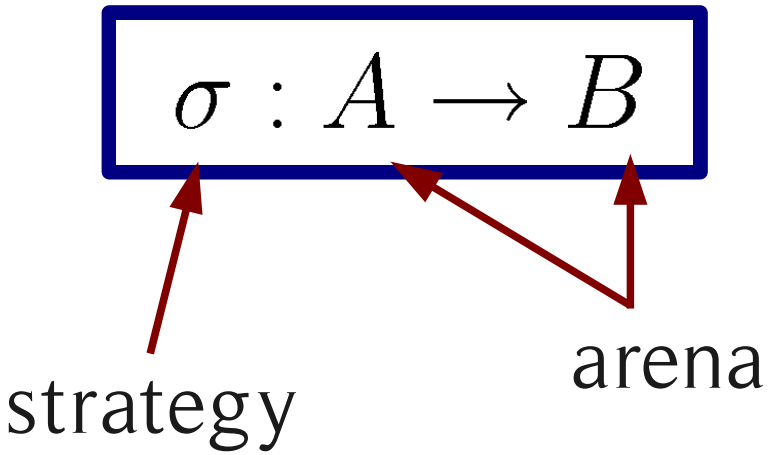
Game Semantics with Names

Computation is a game between P and O

Game Semantics with Names

Computation is a game between P and O

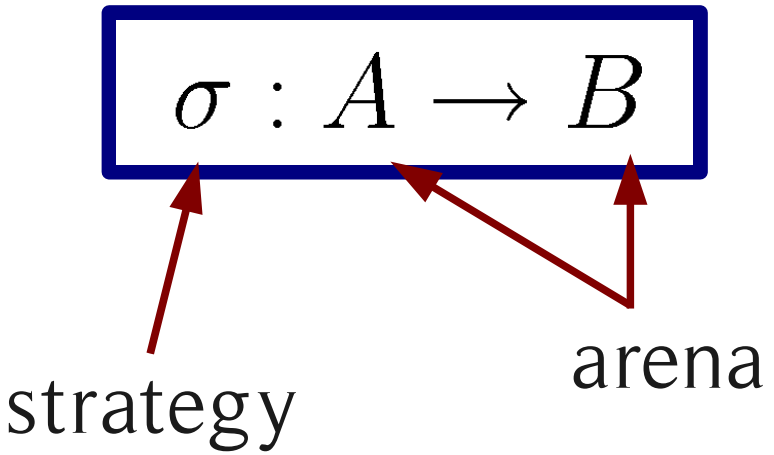
- For each S construct a *category of games* $\mathcal{G}_S$:



Game Semantics with Names

Computation is a game between P and O

- For each S construct a *category of games* $\mathcal{G}_S$:



- $S; \Gamma \vdash M : T \longmapsto [\![M]\!] : [\![\Gamma]\!] \longrightarrow [\![T]\!]$

Examples

$$[\![\lambda x, y.\, x == y]\!] : [\![\emptyset]\!] \longrightarrow [\![\texttt{Name} \to \texttt{Name} \to \texttt{Bool}]\!]$$

$$1 \longrightarrow [\![\ (\texttt{Name}_x \to (\texttt{Name}_y \to \texttt{Bool}^z)^g)^f\ ]\!]$$

$$\begin{array}{cccccc} * & *_f & a_x & *_g & a'_y & \texttt{b}_z \\ O & P & O & P & O & P \end{array}$$

Examples

- $[\![\lambda x, y.\, x == y]\!] : [\![\emptyset]\!] \longrightarrow [\![\mathtt{Name} \to \mathtt{Name} \to \mathtt{Bool}]\!]$

  $1 \longrightarrow [\![\ (\mathtt{Name}_x \to (\mathtt{Name}_y \to \mathtt{Bool}^z)^g)^f\ ]\!]$

$$\begin{array}{cccccc} * & *_f & a_x & *_g & a'_y & \mathtt{b}_z \\ O & P & O & P & O & P \end{array}$$

- $[\![\nu a.\lambda x.\, a]\!] : 1 \longrightarrow [\![\mathtt{B} \to \mathtt{N}]\!]$ $\left\{ \begin{array}{cccccc} * & * & \mathtt{b} & a & \mathtt{b} & a \ \cdots \\ O & P & O & P & O & P \end{array} \right.$

$[\![\lambda x.\nu a.\, a]\!] : 1 \longrightarrow [\![\mathtt{B} \to \mathtt{N}]\!]$ $\left\{ \begin{array}{cccccc} * & * & \mathtt{b} & a & \mathtt{b} & a' \ \cdots \\ O & P & O & P & O & P \end{array} \right.$

Full Abstraction

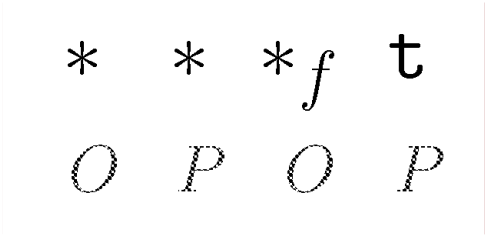
- Define semantic equivalence:

$$\sigma_1 \approx \sigma_2 : 1 \to A \;\equiv$$
$$\forall \rho : A \to [\![\mathtt{B}]\!].\ \sigma_1; \rho = \{ * \mathtt{t} \} \iff \sigma_2; \rho = \{ * \mathtt{t} \}$$

- $S; \emptyset \vdash t_1 \cong t_2 \iff [\![t_1]\!] \approx [\![t_2]\!]$

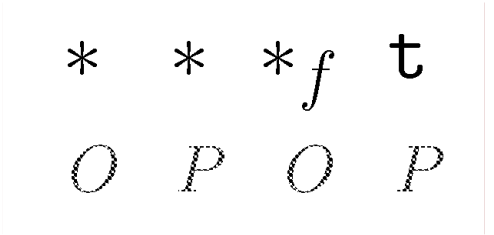
Proof of (6)

- $[\![\lambda f.\,\mathtt{true}]\!] : 1 \longrightarrow [\![(\mathtt{N} \to \mathtt{B}) \to \mathtt{B}]\!]$

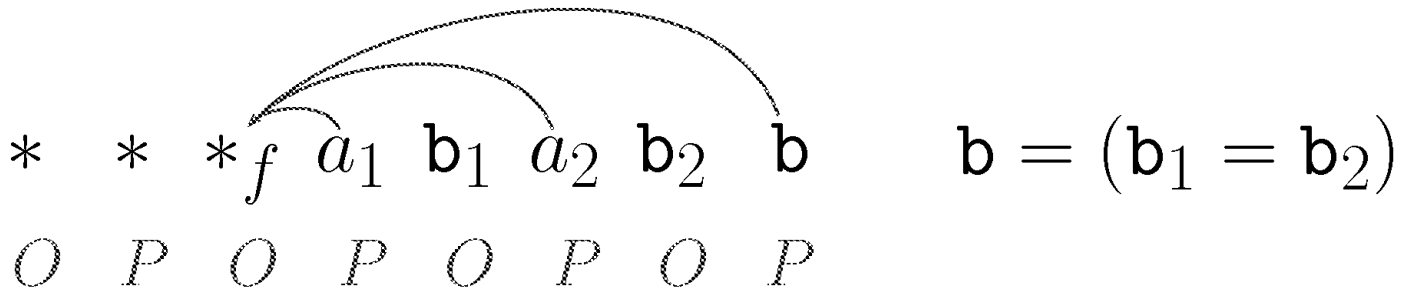


Proof of (6)

- $[\![\lambda f.\,\mathtt{true}]\!] : 1 \longrightarrow [\![(\mathtt{N} \to \mathtt{B}) \to \mathtt{B}]\!]$



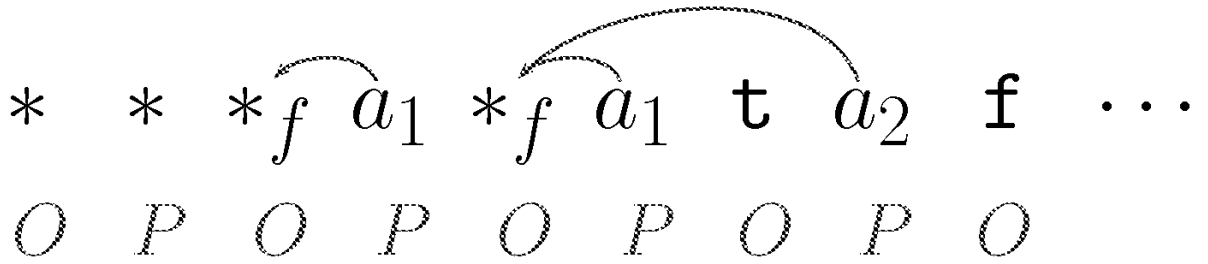
- $[\![\nu a_1, a_2.\lambda f.\, f a_1 = f a_2]\!] : 1 \longrightarrow [\![(\mathtt{N} \to \mathtt{B}) \to \mathtt{B}]\!]$



Proof of (6) (II)

$$[\![\nu a_1, a_2.\lambda f.\, f a_1 = f a_2]\!] : 1 \longrightarrow [\![(\mathtt{N} \to \mathtt{B}) \to \mathtt{B}]\!]$$

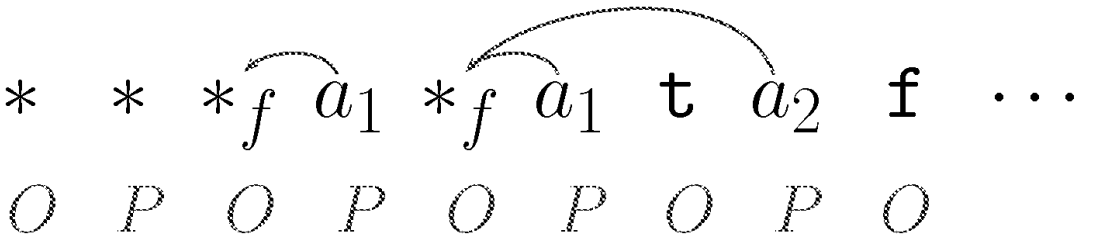
- Recursive calls?



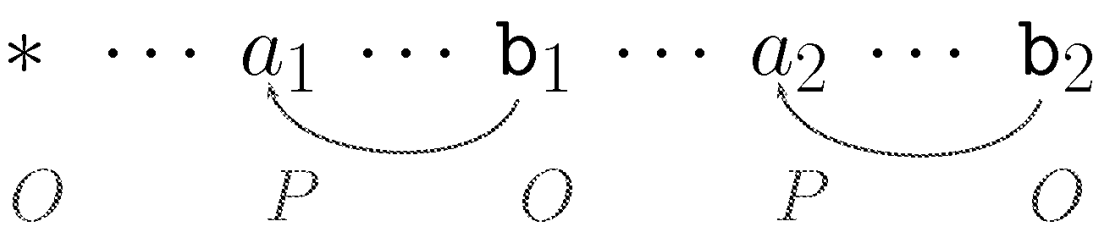
Proof of (6) (II)

$$[\![\nu a_1, a_2.\lambda f.\, f a_1 = f a_2]\!] : 1 \longrightarrow [\![(\mathtt{N} \to \mathtt{B}) \to \mathtt{B}]\!]$$

- Recursive calls?



- *Lemma*: For any

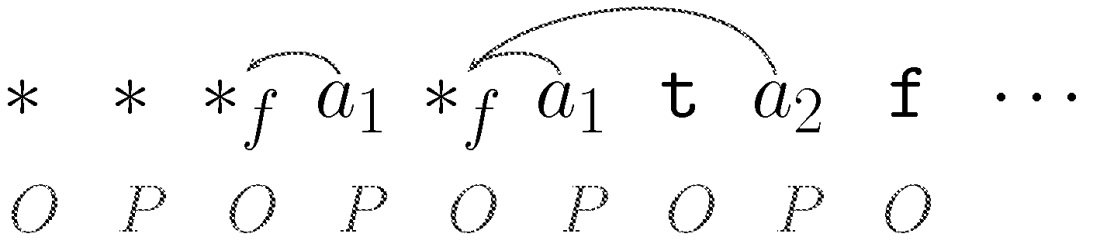


$$\mathsf{view}(* \cdots a_1) = (a_1\ a_2) \cdot \mathsf{view}(* \cdots a_2) \implies \mathtt{b}_1 = \mathtt{b}_2 .$$

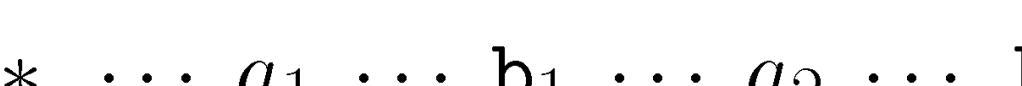
Proof of (6) (II)

$$[\![\nu a_1, a_2.\lambda f.\, f a_1 = f a_2]\!] : 1 \longrightarrow [\![(\mathtt{N} \to \mathtt{B}) \to \mathtt{B}]\!]$$

- Recursive calls?

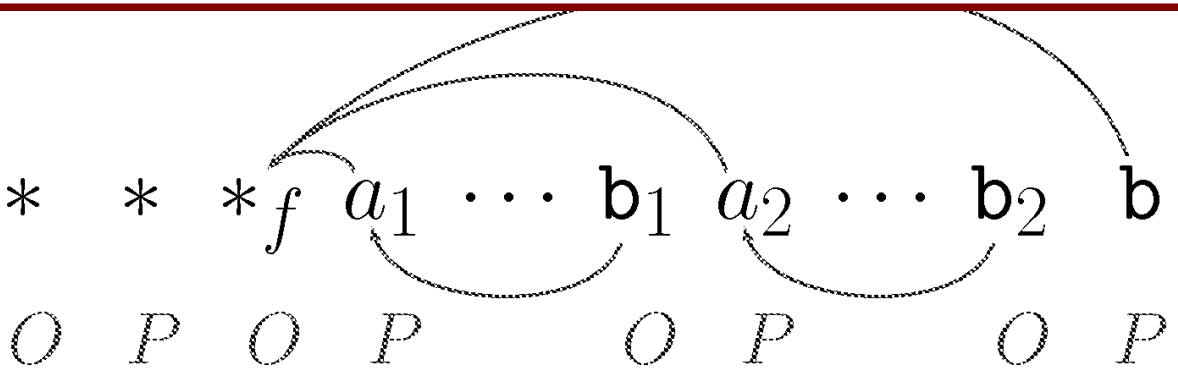


- *Lemma*: For any $* \;\cdots\; a_1 \;\cdots\; \mathtt{b}_1 \;\cdots\; a_2 \;\cdots\; \mathtt{b}_2$,



$$\mathsf{view}(* \cdots a_1) = (a_1\ a_2) \cdot \mathsf{view}(* \cdots a_2) \implies \mathtt{b}_1 = \mathtt{b}_2 .$$

- *Apply to*:



~~

- Full abstraction
- Strategies derived from the syntax compositionally

- Checking semantic equivalence not generally easy

Benton & Koutavas' proof

A Mechanized Bisimulation for the Nu-Calculus

Nick Benton
Microsoft Research, Cambridge

Vasileios Koutavas*
Trinity College, Dublin

Abstract.

We introduce a Sumii-Pierce-Koutavas-Wand-style bisimulation for Pitts and Stark's nu-calculus, a simply-typed lambda calculus with fresh name generation. This bisimulation coincides with contextual equivalence and provides a usable and elementary method for establishing all the subtle equivalences given by Stark [29]. We also describe the formalization of soundness and of the examples in the Coq proof assistant.

Benton & Koutavas' proof

A Mechanized Bisimulation for the Nu-Calculus

Nick Benton
Microsoft Research, Cambridge

Vasileios Koutavas*
Trinity College, Dublin

Abstract.

Abstract.

We introduce a Sumii-Pierce-Koutavas-Wand-style bisimulation for Pitts and Stark's nu-calculus, a simply-typed lambda calculus with fresh name generation. This bisimulation coincides with contextual equivalence and provides a usable and elementary method for establishing all the subtle equivalences given by Stark [29]. We also describe the formalization of soundness and of the examples in the Coq proof assistant.

Annotated relations

- Define *annotated relations*: (S_1, S_2, R)

  with $\quad R = \langle\, R_T \,\rangle_{all\ T}$

  $$R_T \subseteq Val_T(S_1) \times Val_T(S_2)$$

- $$\frac{\emptyset; \overline{x : T} \vdash s : T \qquad \overline{v_1\ R_T\ v_2}}{s[\overline{v_1/x}]\ \ R_T^*\ \ s[\overline{v_2/x}]}$$

Adequate Bisimulations Sets

- Examine sets $\mathcal{X}$ of annotated relations
- $\mathsf{Adequate}(\mathcal{X}) \equiv \forall (S_1, S_2, R) \in \mathcal{X}.\, \forall t_1, t_2, S_1', v_1\,.$

$$
\begin{aligned}
& t_1\, R_T^*\, t_2 \;\wedge\; (S_1, t_1) \twoheadrightarrow (S_1', v_1) \\
\Longrightarrow\; & \exists S_2', v_2, Q\,.\; R \subseteq Q \\
& \wedge\; (S_1 \uplus S_1', S_2 \uplus S_2', Q) \in \mathcal{X} \\
& \wedge\; (S_2, t_2) \twoheadrightarrow (S_2', v_2) \\
& \wedge\; (T = \mathsf{B}) \implies (v_1 = v_2) \\
& \wedge\; v_1\, Q_T^*\, v_2
\end{aligned}
$$

Completeness

- Define *flattening*:

$$S; \overline{x : T} \vdash t_1 \ (\mathcal{X})^\circ \ t_2 : T \ \equiv$$
$$\exists (S, S, R) \in \mathcal{X}.\ (\lambda \overline{x}.t_1 \ R_T \ \lambda \overline{x}.t_2) \wedge (\forall a \in S.\ a \ R_{\mathtt{N}} \ a)$$

- $t_1 \cong t_2 \iff \exists \mathsf{Adequate}(\mathcal{X}).\ t_1 (\mathcal{X})^\circ t_2$

Proof of (6)

- $U_{a_1 a_2} \equiv \lambda f.\,(f a_1 = f a_2)$

  $\tau_1 \equiv \nu a_1, a_2.\, U_{a_1 a_2}$

  $\tau_2 \equiv \lambda f.\, \texttt{true}$

$$\frac{}{(\emptyset, \emptyset, \{\,(\lambda y.\tau_1\,, \lambda y.\tau_2)\,\}) \in \mathcal{X}}$$

$$\frac{(S_1, S_2, R) \in \mathcal{X} \qquad S_1 \cap \{a_1, a_2\} = \emptyset}{(S_1 \uplus \{a_1, a_2\}, S_2, R \cup \{\,(U_{a_1 a_2}, \tau_2)\,\}) \in \mathcal{X}}$$

$$\frac{(S_1, S_2, R) \in \mathcal{X} \quad R' : S_1' \rightleftharpoons S_2' \quad S_i \cap S_i' = \emptyset}{(S_1 \uplus S_1', S_2 \uplus S_2', R \uplus R') \in \mathcal{X}}$$

Check first rule

$$\frac{}{(\emptyset, \emptyset, \{\, (\lambda y.\tau_1\,, \lambda y.\tau_2)\,\}) \in \mathcal{X}}$$

$$\begin{aligned}
\mathsf{Adequate}(\mathcal{X}) \;\equiv\; & \forall (S_1, S_2, R) \in \mathcal{X}.\, \forall t_1, t_2, S_1', v_1\,. \\
& t_1\, R_T^*\, t_2 \;\wedge\; (S_1, t_1) \twoheadrightarrow (S_1', v_1) \\
& \Longrightarrow\; \exists S_2', v_2, Q\,.\; R \subseteq Q \\
& \qquad \wedge\; (S_1 \uplus S_1', S_2 \uplus S_2', Q) \in \mathcal{X} \\
& \qquad \wedge\; (S_2, t_2) \twoheadrightarrow (S_2', v_2) \\
& \qquad \wedge\; (T = \mathsf{B}) \;\Longrightarrow\; (v_1 = v_2) \\
& \qquad \wedge\; v_1\, Q_T^*\, v_2
\end{aligned}$$

Check first rule

$$\overline{(\emptyset, \emptyset, \{\, (\lambda y.\tau_1\,, \lambda y.\tau_2)\,\}) \in \mathcal{X}}$$

- Suppose $(S_1, S_2, R) \in \mathcal{X}\,,\ t_1\ R^*\ t_2$
- wlog: $(\lambda y.\tau_1)\,\mathtt{b}_1\ R^*\ (\lambda y.\tau_2)\,\mathtt{b}_2$

$$\mathtt{b}_1\ R^*\ \mathtt{b}_2$$

Check first rule

$$\frac{}{(\emptyset, \emptyset, \{\, (\lambda y.\tau_1\,, \lambda y.\tau_2)\,\}) \in \mathcal{X}}$$

- Suppose $(S_1, S_2, R) \in \mathcal{X}\,,\ t_1\ R^*\ t_2$
- wlog: $(\lambda y.\tau_1)\,\mathtt{b}_1\ R^*\ (\lambda y.\tau_2)\,\mathtt{b}_2$

$$\mathtt{b}_1\ R^*\ \mathtt{b}_2$$

- Then $(S_1, (\lambda y.\tau_1)\,\mathtt{b}_1) \twoheadrightarrow (S_1 \uplus \{a_1, a_2\}, U_{a_1 a_2})$
  $(S_2, (\lambda y.\tau_2)\,\mathtt{b}_2) \twoheadrightarrow (S_2, \tau_2)$

  and $S_1 \uplus \{a_1, a_2\}, S_2, R \cup \{\, (U_{a_1 a_2}, \tau_2)\,\}) \in \mathcal{X}$

Check second rule

$$\frac{(S_1, S_2, R) \in \mathcal{X} \qquad S_1 \cap \{a_1, a_2\} = \emptyset}{(S_1 \uplus \{a_1, a_2\}, S_2, R \cup \{\, (U_{a_1 a_2}, \tau_2) \,\}) \in \mathcal{X}}$$

- Suppose $(S_1, S_2, R) \in \mathcal{X}\,,\ t_1\ R^*\ t_2$
- wlog: $(U_{a_1 a_2}\, v_1)\ R^*\ (\tau_2\, v_2)$

$$v_1\ R^*\ v_2$$

Check second rule

$$\frac{(S_1, S_2, R) \in \mathcal{X} \qquad S_1 \cap \{a_1, a_2\} = \emptyset}{(S_1 \uplus \{a_1, a_2\}, S_2, R \cup \{\,(U_{a_1 a_2}, \tau_2)\,\}) \in \mathcal{X}}$$

- Suppose $(S_1, S_2, R) \in \mathcal{X}\,,\ t_1\ R^*\ t_2$
- wlog: $(U_{a_1 a_2}\, v_1)\ R^*\ (\tau_2\, v_2)$

$$v_1\ R^*\ v_2$$

- Then $(S_1, U_{a_1 a_2}\, v_1) \twoheadrightarrow (S_1 \uplus S', \mathtt{b})$

$$(S_2, \tau_2\, v_2) \twoheadrightarrow (S_2, \mathtt{true})$$

- *show*: $\mathtt{b} = \mathtt{true}$

Check second rule (II)

- *Show*: $U_{a_1 a_2}\, v_1 \twoheadrightarrow \texttt{true}$
- Note $v_1\ R^*\ v_2 \implies v_1 \equiv \lambda z.\, s[\overline{u_1/x}]\,, \quad \overline{u_1\ R\ u_2}$
- so: $U_{a_1 a_2}\, v_1 \twoheadrightarrow\ s(\overline{u_1})[a_1/z] = s(\overline{u_1})[a_2/z]$

Check second rule (II)

- *Show*: $U_{a_1 a_2}\, v_1 \twoheadrightarrow \texttt{true}$
- Note $v_1\ R^*\ v_2 \implies v_1 \equiv \lambda z.\, s[\overline{u_1/x}]\,, \quad \overline{u_1\ R\ u_2}$
- so: $U_{a_1 a_2}\, v_1 \twoheadrightarrow\ s(\overline{u_1})[a_1/z] = s(\overline{u_1})[a_2/z]$

$$
\begin{aligned}
& s(\overline{u_1})[a_1/z] \twoheadrightarrow \texttt{b} \\
& \quad \implies\ (a_1\, a_2) \cdot (s(\overline{u_1})[a_1/z]) \twoheadrightarrow \texttt{b} \\
& \quad \implies\ s(\overline{u_1})[a_2/z][U_{a_2 a_1}/U_{a_1 a_2}] \twoheadrightarrow \texttt{b} \\
& \quad \implies\ s(\overline{u_1})[a_2/z] \twoheadrightarrow \texttt{b}
\end{aligned}
$$

~~

- A complete method
- Proofs can be checked

- Bisimulation set needs to be guessed
- Showing adequacy is not automatic

Last slide

- Several methods for contextual equivalence
- The language is simply difficult

Last slide

- Several methods for contextual equivalence
- The language is simply difficult

  (is equivalence decidable?)

- *Logical relations / bisimulations on strategies?*