Separating Terms through Multi Types

Adrienne Lancelot

Inria & LIX, École Polytechnique
IRIF, Université Paris Cité & CNRS

July 9th 2024 – ITRS

Are these program equivalent?

When studying program equivalence, one searches for any small hint that programs are not equivalent:

$\lambda x.x + x$ and $\lambda x.x * x$ can be distinguished by the input 1 (but not by the input 2)

$\lambda x.x + x$ and $\lambda x.$"Hello World" can be distinguished by types: $\lambda x.x + x : \text{int} \to \text{int}$ and $\lambda x.$"Hello World" $: \alpha \to \text{string}$

Separating Terms

Let t and u two λ-terms.

- Does there exist a context C such that $C\langle t\rangle$ does not terminate and $C\langle u\rangle$ terminates?
- Does there exist a typing judgment (Γ, L) such that $\Gamma \vdash t : L$ but $\Gamma \nvdash u : L$?

This talk is about separating with **Intersection Types**!

Separating Terms

Let t and u two λ-terms.

- Does there exist a context C such that $C\langle t\rangle$ does not terminate and $C\langle u\rangle$ terminates?
- Does there exist a typing judgment (Γ, L) such that $\Gamma \vdash t : L$ but $\Gamma \not\vdash u : L$?

This talk is about separating with **Intersection Types**!

Böhm Out Technique

Separating Terms with Contexts: let's reduce term to their head normal form and find an appropriate context.

- $\mathtt{I}$ and Ω are separated by the empty context $\langle\cdot\rangle$
- $x\,\mathtt{I}\,\mathtt{I}$ and $x\,\Omega\,\mathtt{I}$ are separated by $(\lambda x.\langle\cdot\rangle)\pi_1$ where $\pi_1 = \lambda x.\lambda y.x$
- $y\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $y\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by $(\lambda y.(\lambda x.\langle\cdot\rangle)\pi_1)\pi_2$ where $\pi_2 = \lambda x.\lambda y.y$
- $x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by **??**

Böhm Out Technique

Separating Terms with Contexts: let's reduce term to their head normal form and find an appropriate context.

- $\mathtt{I}$ and Ω are separated by the empty context $\langle\cdot\rangle$
- $x\,\mathtt{I}\,\mathtt{I}$ and $x\,\Omega\,\mathtt{I}$ are separated by $(\lambda x.\langle\cdot\rangle)\pi_1$ where $\pi_1 = \lambda x.\lambda y.x$
- $y\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $y\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by $(\lambda y.(\lambda x.\langle\cdot\rangle)\pi_1)\pi_2$ where $\pi_2 = \lambda x.\lambda y.y$
- $x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by **??**

Böhm Out Technique

Separating Terms with Contexts: let's reduce term to their head normal form and find an appropriate context.

- $\mathtt{I}$ and Ω are separated by the empty context $\langle\cdot\rangle$
- $x\,\mathtt{I}\,\mathtt{I}$ and $x\,\Omega\,\mathtt{I}$ are separated by $(\lambda x.\langle\cdot\rangle)\pi_1$ where $\pi_1 = \lambda x.\lambda y.x$
- $y\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $y\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by $(\lambda y.(\lambda x.\langle\cdot\rangle)\pi_1)\pi_2$ where $\pi_2 = \lambda x.\lambda y.y$
- $x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by **??**

Böhm Out Technique

Separating Terms with Contexts: let's reduce term to their head normal form and find an appropriate context.

- $\mathtt{I}$ and Ω are separated by the empty context $\langle\cdot\rangle$
- $x\,\mathtt{I}\,\mathtt{I}$ and $x\,\Omega\,\mathtt{I}$ are separated by $(\lambda x.\langle\cdot\rangle)\pi_1$ where $\pi_1 = \lambda x.\lambda y.x$
- $y\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $y\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by $(\lambda y.(\lambda x.\langle\cdot\rangle)\pi_1)\pi_2$ where $\pi_2 = \lambda x.\lambda y.y$
- $x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by **??**

Böhm Out Technique

Separating Terms with Contexts: let's reduce term to their head normal form and find an appropriate context.

- $\mathtt{I}$ and Ω are separated by the empty context $\langle\cdot\rangle$
- $x\,\mathtt{I}\,\mathtt{I}$ and $x\,\Omega\,\mathtt{I}$ are separated by $(\lambda x.\langle\cdot\rangle)\pi_1$ where $\pi_1 = \lambda x.\lambda y.x$
- $y\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $y\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by $(\lambda y.(\lambda x.\langle\cdot\rangle)\pi_1)\pi_2$ where $\pi_2 = \lambda x.\lambda y.y$
- $x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I})$ and $x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$ are separated by **??**

Type-Böhm Out Technique

Separating Terms with Types: easier than context Böhm out to select subterms!

$$x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I}) \quad \text{and} \quad x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$$

$$x : [\,[A] \multimap [B] \multimap B, [A] \multimap [B] \multimap A\,] \quad \vdash \_ : \quad B$$

$$\Gamma \vdash x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I}) : B \quad \text{but} \quad \Gamma \nvdash x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I}) : B$$

Type-Böhm Out Technique

Separating Terms with Types: easier than context Böhm out to select subterms!

$$x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I}) \quad \text{and} \quad x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$$

$$x : [\,[A] \multimap [B] \multimap B, [A] \multimap [B] \multimap A\,] \quad \vdash \_ : \quad B$$

$$\Gamma \vdash x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I}) : B \quad \text{but} \quad \Gamma \nvdash x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I}) : B$$

This Talk

For Weak Head Call-by-Name:

$$\text{NF Bisimilarity} \quad \underset{\text{Type-Böhm Out}}{\overset{\text{Derivation Transfer}}{\rightleftarrows}} \quad \text{Type Equivalence}$$

- A *coinductive flavor* (normal form bisimulations): no approximants
- **Derivation transfer** requires non idempotent intersection types

Weak Head Multi Types

$$\begin{array}{lccl} \textsc{Linear Types} & L, L' & ::= & \star_k \mid M \multimap L \quad k \in \mathbb{N} \\ \textsc{Multi Types} & M, N & ::= & [L_1, \ldots, L_n] \quad n \geq 0 \end{array}$$

$$\frac{}{x : [L] \vdash x : L}\ \mathsf{ax} \qquad \frac{}{\emptyset \vdash \lambda x.t : \star_k}\ \lambda_k \qquad \frac{\Gamma, x : M \vdash t : L}{\Gamma \vdash \lambda x.t : M \multimap L}\ \lambda$$

$$\frac{\Gamma \vdash t : [L_i]_{i \in I} \multimap L' \quad (\Gamma_i \vdash u : L_i)_{i \in I} \quad I \text{ finite}}{\Gamma \uplus \Delta \vdash tu : L'}\ @$$

Type Equivalence

Type Equivalence: $t \simeq_{type} u$ if $\forall(\Gamma, L)\ \ \Gamma \vdash t : L \iff \Gamma \vdash u : L$

β-equivalence is included in $\simeq_{type}$:

- by Subject Reduction and Expansion

η-equivalence is not included in $\simeq_{type}$:

- $x : [\star_0] \vdash x : \star_0$ but $x : [\star_0] \not\vdash \lambda y.x\, y : \star_0$
- $\emptyset \vdash \lambda y.x\, y : \star_0$ but $\emptyset \not\vdash x : \star_0$

Normal form bisimilarity

The syntactical characterization is phrased in a coinductive way, using Sangiorgi's normal form bisimilarity. It coincides with the inequational theory induced by Lévy-Longo trees.

Definition (Weak head normal form similarity)

A relation $\mathcal{R}$ is a **weak head normal (whnf) form simulation** if whenever $t \mathrel{\mathcal{R}} u$ holds, we have that either:

- $(\bot)$ $t \not\Downarrow_{wh}$, or,
- (abs) $t \Downarrow_{wh} \lambda x.t'$ and $u \Downarrow_{wh} \lambda x.u'$ with $t' \mathrel{\mathcal{R}} u'$, or,
- (app) $t \Downarrow_{wh} y\, t_1 \cdots t_k$ and
  $u \Downarrow_{wh} y\, u_1 \cdots u_k$ with $(t_i \mathrel{\mathcal{R}} u_i)_{i \leq k}$.

Whnf similarity, noted $\precsim_{\mathrm{nf}}$, is the largest whnf simulation.

Derivation Transfer

From bisimilar terms to type equivalent types

$$\text{NF Bisimilarity} \xRightarrow{\text{Derivation Transfer}} \text{Type Equivalence}$$

The proof goes by induction on the size of derivations.

Assume $t \precsim_{\mathrm{nf}} u$ and $\pi \triangleright \Gamma \vdash t : L$.

By Quantitative **Subject Reduction**, $\pi' \triangleright \Gamma \vdash n_t : L$ s.t. $|\pi'| \leq |\pi|$.

By case analysis on the shape of n_t:

Application case:

$$\pi' : \frac{\Gamma \vdash x t_1 \cdots t_{k-1} : [L_i]_{i \in I} \multimap L \quad (\Delta_i \vdash t_k : L_i)_{i \in I}}{\Gamma \uplus (\uplus_{i \in I} \Delta_i) \vdash n_t = x t_1 \cdots t_k : L} \;@$$

$\rightsquigarrow \quad \sigma \triangleright \Gamma \vdash n_u : L \quad$ as $x t_1 \cdots t_{k-1} \precsim_{\mathrm{nf}} x u_1 \cdots u_{k-1}$ and $t_k \precsim_{\mathrm{nf}} u_k$.

By Subject Expansion, $\sigma' \triangleright \Gamma \vdash u : L$.

Derivation Transfer

From bisimilar terms to type equivalent types

NF Bisimilarity $\xRightarrow{\text{Derivation Transfer}}$ Type Equivalence

The proof goes by induction on the size of derivations.
Assume $t \precsim_{\mathrm{nf}} u$ and $\pi \triangleright \Gamma \vdash t : L$.
By **Quantitative Subject Reduction**, $\pi' \triangleright \Gamma \vdash n_t : L$ s.t. $|\pi'| \leq |\pi|$.
By case analysis on the shape of n_t:
Application case:

$$\pi' : \frac{\Gamma \vdash x t_1 \cdots t_{k-1} : [L_i]_{i \in I} \multimap L \quad (\Delta_i \vdash t_k : L_i)_{i \in I}}{\Gamma \uplus (\uplus_{i \in I} \Delta_i) \vdash n_t = x t_1 \cdots t_k : L} \; \texttt{@}$$

$\rightsquigarrow \quad \sigma \triangleright \Gamma \vdash n_u : L$ as $x t_1 \cdots t_{k-1} \precsim_{\mathrm{nf}} x u_1 \cdots u_{k-1}$ and $t_k \precsim_{\mathrm{nf}} u_k$.
By Subject Expansion, $\sigma' \triangleright \Gamma \vdash u : L$.

Type Equivalence & Compositionality

Type equivalence is **compositional**.

- $t \precsim_{\text{type}} u$ and $s \precsim_{\text{type}} r$ implies $ts \precsim_{\text{type}} ur$
- $t \precsim_{\text{type}} u$ implies $\lambda x.t \precsim_{\text{type}} \lambda x.u$

Proposition (Soundness wrto Contextual Equivalence)

If $t \precsim_{\text{type}} u$ then $t \leq_C^{wh} u$

Side note: $t \leq_C^{wh} u$ does not imply $t \precsim_{\text{type}} u$ (problems with η...)

Type Equivalence & Compositionality

Type equivalence is **compositional**.

- $t \precsim_{\text{type}} u$ and $s \precsim_{\text{type}} r$ implies $ts \precsim_{\text{type}} ur$
- $t \precsim_{\text{type}} u$ implies $\lambda x.t \precsim_{\text{type}} \lambda x.u$

Proposition (Soundness wrto Contextual Equivalence)

If $t \precsim_{\text{type}} u$ then $t \leq_C^{wh} u$

Side note: $t \leq_C^{wh} u$ does not imply $t \precsim_{\text{type}} u$ (problems with η...)

Type-Böhm Out

Separating un-bisimilar terms

NF Bisimilarity $\Longleftarrow$ Type-Böhm Out $=\!=\!=$ Type Equivalence

Coinduction principle: $\precsim_{\text{type}}$ is a whnf simulation $\Rightarrow$ $\precsim_{\text{type}} \subseteq \precsim_{\text{nf}}$

We show that $\precsim_{\text{type}}$ is a whnf simulation by contrapositive:
Let t and u such that $t \Downarrow_{wh} n_t$ and $u \Downarrow_{wh} n_u$ with **different** normal forms

$\leadsto$ **Build a Separating Type**

Separating Terms through Multi Types

Some examples

1. *Separating any abstraction $\lambda x.t$ and any applied nf $x\, t_1 \cdots t_k$:*

$$\dfrac{}{\Gamma \vdash \lambda x.t : \star_0}\ \lambda_0$$	$$\dfrac{\vdots}{\Gamma \vdash x\, t_1 \cdots t_k : \star_0}$$

Then Γ must have the shape:

$\Gamma = \emptyset$	$\Gamma = x : [M_1 \multimap \cdots, \ldots]$

2. *Separating a variable x and an applied nf xt:*
3. *Separating different abstractions:*
4. *Separating applied nf where arguments differ, say xt and xu:*

′

Separating Terms through Multi Types

Some examples

1. *Separating any abstraction* $\lambda x.t$ *and any applied nf* $x\, t_1 \cdots t_k$*:*
2. *Separating a variable* x *and an applied nf* xt*:*

$$\frac{}{\Gamma \vdash x : \star_0}\ \text{ax} \qquad \Bigg| \qquad \frac{\vdots}{\Gamma \vdash x\, t : \star_0}$$

Then Γ must resemble:

$$\Gamma = x : [\star_0] \qquad \Big| \qquad \Gamma \supseteq x : [M \multimap \cdots, \ldots]$$

3. *Separating different abstractions:*
4. *Separating applied nf where arguments differ, say* xt *and* xu*:*

′

Separating Terms through Multi Types

Some examples

1. *Separating any abstraction $\lambda x.t$ and any applied nf $x\, t_1 \cdots t_k$:*
2. *Separating a variable x and an applied nf xt:*
3. *Separating different abstractions:*

Suppose we have $\Gamma, x : M \vdash t : L$ but $\Gamma, x : M \not\vdash u : L$

$$\frac{\Gamma, x : M \vdash t : L}{\Gamma \vdash \lambda x.t : M \multimap L}\ \lambda$$

If $$\frac{\vdots}{\Gamma \vdash \lambda x.u : M \multimap L}$$

Then it must start with:

$$\frac{\Gamma, x : M \vdash u : L}{\Gamma \vdash \lambda x.u : M \multimap L}\ \lambda$$

4. *Separating applied nf where arguments differ, say xt and xu:*

′

Separating Terms through Multi Types

Some examples

1. *Separating any abstraction $\lambda x.t$ and any applied nf $x\, t_1 \cdots t_k$:*
2. *Separating a variable x and an applied nf xt:*
3. *Separating different abstractions:*
4. *Separating applied nf where arguments differ, say xt and xu:*

Suppose we have $\Gamma \vdash t : L$ but $\Gamma \nvdash u : L$

$$\frac{\Delta \vdash x : [L] \multimap \star_k \quad \Gamma \vdash t : L}{\Gamma \uplus \Delta \vdash x\, t : \star_i}$$

If $\dfrac{\vdots}{\Gamma \vdash x\, u : \star_i}$

Then it must start with:

$$\frac{\Delta' \vdash x : [L_i]_i \multimap \star_k \quad (\Gamma_i \vdash u : L_i)_i}{\Gamma \uplus \Delta' \vdash x\, u : \star_i}$$

If k is well chosen, then $\Delta' = x : [[L] \multimap \star_k]$

′

Factorizing the need of countable atoms

Compositionality of $\not\precsim_{\mathrm{type}}$ is the only part that requires many distinguishable ground types!

Intuitively: $t \not\precsim_{\mathrm{type}} u$ or $s \not\precsim_{\mathrm{type}} r$ implies $ts \not\precsim_{\mathrm{type}} ur$

- *Left difference:* $x\, t_1 \cdots t_k \not\precsim_{\mathrm{type}} y\, u_1 \cdots u_{k'}$ implies $x\, t_1 \cdots t_k\, \mathrm{s} \not\precsim_{\mathrm{type}} y\, u_1 \cdots u_{k'}\, \mathrm{r}$.
- *Right difference:* $x\, t_1 \cdots t_k \precsim_{\mathrm{type}} y\, u_1 \cdots u_{k'}$ and $s \not\precsim_{\mathrm{type}} r$ implies $x\, t_1 \cdots t_k\, \mathrm{s} \not\precsim_{\mathrm{type}} y\, u_1 \cdots u_k\, \mathrm{r}$.

Factorizing the need of countable atoms

Compositionality of $\not\precsim_{\mathrm{type}}$ is the only part that requires many distinguishable ground types!

Intuitively: $t \not\precsim_{\mathrm{type}} u$ or $s \not\precsim_{\mathrm{type}} r$ implies $ts \not\precsim_{\mathrm{type}} ur$

- *Left difference:* $x\, t_1 \cdots t_k \not\precsim_{\mathrm{type}} y\, u_1 \cdots u_{k'}$ implies $x\, t_1 \cdots t_k\, \mathsf{s} \not\precsim_{\mathrm{type}} y\, u_1 \cdots u_{k'}\, \mathsf{r}$.
- *Right difference:* $x\, t_1 \cdots t_k \precsim_{\mathrm{type}} y\, u_1 \cdots u_{k'}$ and $s \not\precsim_{\mathrm{type}} r$ implies $x\, t_1 \cdots t_k\, \mathsf{s} \not\precsim_{\mathrm{type}} y\, u_1 \cdots u_k\, \mathsf{r}$.

Conclusion

$$\text{NF Bisimilarity} \quad \underset{\Longleftarrow\text{ Type-Böhm Out }=\!=}{=\!=\text{ Derivation Transfer }\Longrightarrow} \quad \text{Type Equivalence}$$

Perspectives:

- Towards Call-by-Value equivalences, where NF bisimilarities are more involved
- Can we adapt type equivalence to reach full abstraction?

$$\text{Ctx. Equiv.} \quad \underset{\Longleftarrow\text{ Compositionality? }=\!=}{=\!=\text{ Definability? }\Longrightarrow} \quad \textbf{Inhabited}\text{ Type Equiv.}$$

- Can Type-Böhm out help to simplify (Context-)Böhm out ?

Thank you!

Conclusion

NF Bisimilarity ═══ Derivation Transfer ⟹ Type Equivalence

NF Bisimilarity ⟸ Type-Böhm Out ═══ Type Equivalence

Perspectives:

- Towards Call-by-Value equivalences, where NF bisimilarities are more involved
- Can we adapt type equivalence to reach full abstraction?

Ctx. Equiv. ═══ Definability? ⟹ **Inhabited** Type Equiv.

Ctx. Equiv. ⟸ Compositionality? ═══ **Inhabited** Type Equiv.

- Can Type-Böhm out help to simplify (Context-)Böhm out ?

Thank you!