Separating Terms by Means of Multi Types, Coinductively

Adrienne Lancelot

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

June 5th 2025 – Workshop on Reasoning with Quantitative Types

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:
$\lambda x.x + x : \mathsf{int} \to \mathsf{int}$ and $\lambda x.$"Hello World" $: \alpha \to \mathsf{string}$

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 $\Gamma \nvdash u : L$?

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!

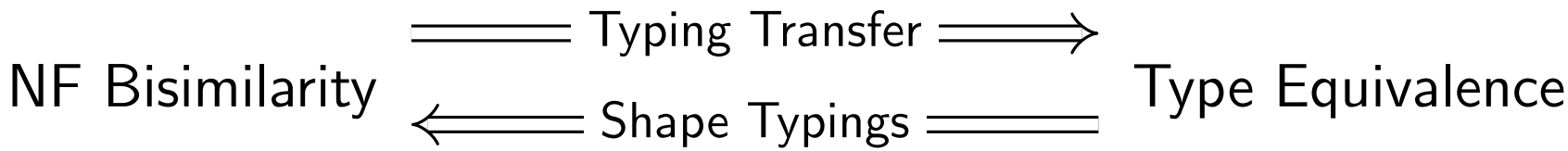
$$x\,\mathtt{I}\,(x\,\Omega\,\mathtt{I}) \quad \text{and} \quad x\,\mathtt{I}\,(x\,\mathtt{I}\,\mathtt{I})$$

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

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

This Talk

For Weak Head Call-by-Name:



- A *coinductive flavor* (normal form bisimulations): no approximants
- **Typing transfer** requires non idempotent intersection types
- **Shape Typings:** a light form of principal types

Weak Head Multi Types

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

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

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

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 $\emptyset \not\vdash x : \star_0$

Type Preorder: $t \precsim_{\text{type}} u$ if $\forall(\Gamma, L)\ \ \Gamma \vdash t : L \implies \Gamma \vdash u : L$

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 $\emptyset \not\vdash x : \star_0$

Type Preorder: $t \precsim_{\text{type}} u$ if $\forall(\Gamma, L)\ \ \Gamma \vdash t : L \implies \Gamma \vdash u : L$

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 $t\ \mathcal{R}\ u$ holds, we have that either:

$(\bot)$ $\quad t \not\Downarrow_{wh}$, or,

(abs) $\quad t \Downarrow_{wh} \lambda x.t'$ and $u \Downarrow_{wh} \lambda x.u'$ with $t'\ \mathcal{R}\ u'$, or,

(app) $\quad t \Downarrow_{wh} y\ t_1 \cdots t_k$ and
$u \Downarrow_{wh} y\ u_1 \cdots u_k$ with $(t_i\ \mathcal{R}\ u_i)_{i \leq k}$.

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

Typing Transfer

From bisimilar terms to type equivalent types

NF Bisimilarity $\Longrightarrow$ Typing Transfer $\Longrightarrow$ 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:

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

$\rightsquigarrow \quad \sigma \triangleright \Gamma \vdash n_u : L$ 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, $\sigma' \triangleright \Gamma \vdash u : L$.

Typing Transfer

From bisimilar terms to type equivalent types

NF Bisimilarity $\Longrightarrow$ Typing Transfer $\Longrightarrow$ 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:

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

$\rightsquigarrow \quad \sigma \triangleright \Gamma \vdash n_u : L$ 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 $\lambda x.t \precsim_{\text{type}} \lambda x.u$

Adequacy: if $t \precsim_{\text{type}} u$ and t is wh-normalizing then so is 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...)

$$y \leq_C^{wh} \lambda x.yx \text{ and } yy \equiv_C^{wh} y\lambda x.yx$$

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 $\lambda x.t \precsim_{\text{type}} \lambda x.u$

Adequacy: if $t \precsim_{\text{type}} u$ and t is wh-normalizing then so is 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...)

$$y \leq_C^{wh} \lambda x.yx \text{ and } yy \equiv_C^{wh} y\lambda x.yx$$

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 $\lambda x.t \precsim_{\text{type}} \lambda x.u$

Adequacy: if $t \precsim_{\text{type}} u$ and t is wh-normalizing then so is 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...)

$$y \leq_C^{wh} \lambda x.yx \text{ and } yy \equiv_C^{wh} y\lambda x.yx$$

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 $\lambda x.t \precsim_{\text{type}} \lambda x.u$

Adequacy: if $t \precsim_{\text{type}} u$ and t is wh-normalizing then so is 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...)

$$y \leq_C^{wh} \lambda x.yx \text{ and } yy \equiv_C^{wh} y\lambda x.yx$$

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 $\lambda x.t \precsim_{\text{type}} \lambda x.u$

Adequacy: if $t \precsim_{\text{type}} u$ and t is wh-normalizing then so is 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...)

$$y \leq_C^{wh} \lambda x.yx \text{ and } yy \equiv_C^{wh} y\lambda x.yx$$

Shape Typings

Ensuring bisimilarity with types

NF Bisimilarity $\Longleftarrow$ Shape Typings $=\!=$ 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:
Let t and u such that $t \Downarrow_{wh} n_t$ and $u \Downarrow_{wh} n_u$
by SR/SE: $n_t \precsim_{\text{type}} n_u$

$\rightsquigarrow$ **Build Shape Typings**

- That ensures that the outer shape of n_t matches the one of n_u
- Also ensures that the inner sub terms of n_t and n_u must be related by $\precsim_{\text{type}}$

Separating Terms with Shape Typings

Some examples

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

$$\frac{}{\Gamma \vdash \lambda x.t : \star_0}\ \lambda_0 \qquad \Bigg| \qquad \frac{\vdots}{\Gamma \vdash x\, t_1 \cdots t_k : \star_0}$$

Then Γ must have the shape:

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

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

´

Separating Terms with Shape Typings

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:*

$$\frac{}{\Gamma \vdash x : \star_0}\ \mathsf{ax} \quad \Big| \quad \frac{\vdots}{\Gamma \vdash x\, t : \star_0}$$

Then Γ must resemble:

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

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

Outer-Shape Typing System

$$\frac{}{x : [\star_k] \vdash_{\text{shape}} x : \star_k}\ \text{shape-ax} \qquad \frac{}{\emptyset \vdash_{\text{shape}} \lambda x.t : \star_k}\ \text{shape-}\lambda_k$$

$$\frac{\Gamma \vdash_{\text{shape}} x\ t_1 \cdots t_i : \star_k \quad \star_k \text{ appears only once in } \Gamma}{\Gamma\{\star_k \leftarrow [\,] \multimap \star_{k'}\} \vdash_{\text{shape}} x\ t_1 \cdots t_i\ t_{i+1} : \star_{k'}}\ \text{shape-@}$$

1. Any normal form n is shape typable:

   there exists $\Gamma, \star_k$ such that $\Gamma \vdash_{\text{shape}} n : \star_k$;
2. The $\vdash_{\text{shape}}$ is sound for $\vdash$:

   if $\Gamma \vdash_{\text{shape}} t : L$ then $\Gamma \vdash t : L$;
3. They distinguish outer shape:

   if $\Gamma \vdash_{\text{shape}} n : \star_k$ and $\Gamma \vdash_{\text{shape}} n' : \star_k$ then either:
   - $n = x = n'$;
   - $n = \lambda x.t$ and $\lambda x.t' = n'$:
   - $n = x\ t_1 \cdots t_i$ and $x\ t'_1 \cdots t'_i = n'$:

Outer-Shape Typing System

$$\frac{}{x : [\star_k] \vdash_{\text{shape}} x : \star_k}\ \text{shape-ax} \qquad \frac{}{\emptyset \vdash_{\text{shape}} \lambda x.t : \star_k}\ \text{shape-}\lambda_k$$

$$\frac{\Gamma \vdash_{\text{shape}} x\ t_1 \cdots t_i : \star_k \quad \star_k \text{ appears only once in } \Gamma}{\Gamma\{\star_k \leftarrow [\,] \multimap \star_{k'}\} \vdash_{\text{shape}} x\ t_1 \cdots t_i\ t_{i+1} : \star_{k'}}\ \text{shape-@}$$

1. Any normal form n is shape typable:

   there exists $\Gamma, \star_k$ such that $\Gamma \vdash_{\text{shape}} n : \star_k$;

2. The $\vdash_{\text{shape}}$ is sound for $\vdash$:

   if $\Gamma \vdash_{\text{shape}} t : L$ then $\Gamma \vdash t : L$;

3. They distinguish outer shape:

   if $\Gamma \vdash_{\text{shape}} n : \star_k$ and $\Gamma \vdash_{\text{shape}} n' : \star_k$ then either:
   - $n = x = n'$;
   - $n = \lambda x.t$ and $\lambda x.t' = n'$:
   - $n = x\ t_1 \cdots t_i$ and $x\ t'_1 \cdots t'_i = n'$:

Separating Terms with Shape Typings

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 abstractions with <u>different inner bodies</u>:*

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

   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 <u>arguments differ, say xt and xu</u>:*

Separating Terms with Shape Typings

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 abstractions with <u>different inner bodies</u>:*
4. *Separating applied nf where <u>arguments differ, say xt and xu</u>:*

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

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

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

Then it must start with:

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

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

Factorizing the need of countable atoms

The *Normal Inversion Lemma* of $\precsim_{\text{type}}$ is the only part that uses many distinguishable ground types!

Intuitively: $ts \precsim_{\text{type}} ur$ implies $t \precsim_{\text{type}} u$ and $s \precsim_{\text{type}} r$

Lemma (Normal Inversion Lemma for $\precsim_{\text{type}}$)

Let t, t', t_i, t'_i be terms.

1. *Body of Abstractions: if $\lambda x.t \precsim_{\text{type}} \lambda x.t'$ then $t \precsim_{\text{type}} t'$.*
2. *Arguments: if $x\, t_1 \cdots t_k \precsim_{\text{type}} x\, t'_1 \cdots t'_k$ then $t_i \precsim_{\text{type}} t'_i$ for $i \leq k$.*

In fact, we can do without and only use a single ground type $\star$!

Factorizing the need of countable atoms

The *Normal Inversion Lemma* of $\precsim_{\text{type}}$ is the only part that uses many distinguishable ground types!

Intuitively: $ts \precsim_{\text{type}} ur$ implies $t \precsim_{\text{type}} u$ and $s \precsim_{\text{type}} r$

Lemma (Normal Inversion Lemma for $\precsim_{\text{type}}$)

Let t, t', t_i, t'_i be terms.

1. *Body of Abstractions: if $\lambda x.t \precsim_{\text{type}} \lambda x.t'$ then $t \precsim_{\text{type}} t'$.*
2. *Arguments: if $x\, t_1 \cdots t_k \precsim_{\text{type}} x\, t'_1 \cdots t'_k$ then $t_i \precsim_{\text{type}} t'_i$ for $i \leq k$.*

In fact, we can do without and only use a single ground type $\star$!

Factorizing the need of countable atoms

The *Normal Inversion Lemma* of $\precsim_{\text{type}}$ is the only part that uses many distinguishable ground types!

Intuitively: $ts \precsim_{\text{type}} ur$ implies $t \precsim_{\text{type}} u$ and $s \precsim_{\text{type}} r$

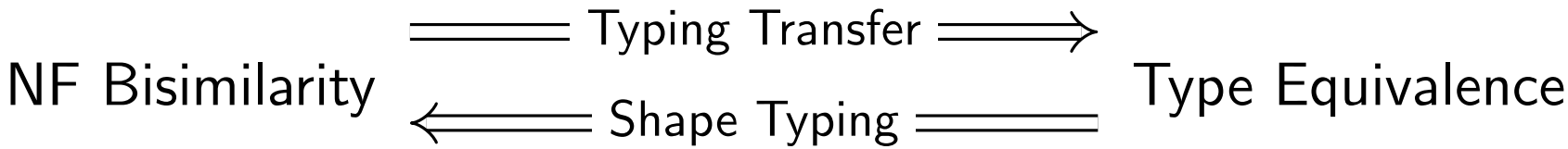
Lemma (Normal Inversion Lemma for $\precsim_{\text{type}}$)

Let t, t', t_i, t'_i be terms.

1. *Body of Abstractions: if $\lambda x.t \precsim_{\text{type}} \lambda x.t'$ then $t \precsim_{\text{type}} t'$.*
2. *Arguments: if $x\, t_1 \cdots t_k \precsim_{\text{type}} x\, t'_1 \cdots t'_k$ then $t_i \precsim_{\text{type}} t'_i$ for $i \leq k$.*

In fact, we can do without and only use a single ground type $\star$!

Conclusion

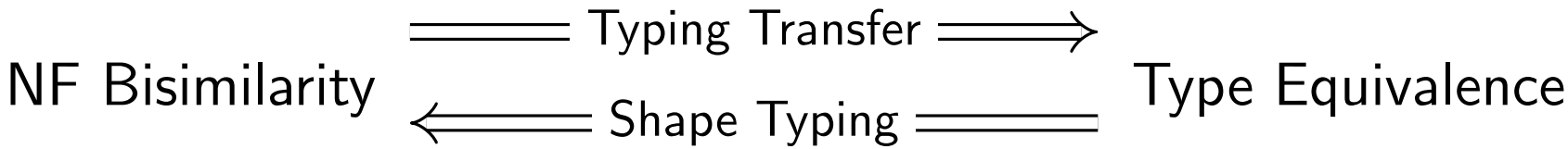


Perspectives:

- Towards Call-by-Value equivalences, where NF bisimilarities are more involved (and Shape Typings less easy to define)
- Can we adapt this coinductive proof technique (without approximants) to the *idempotent* setting?
- Can we adapt type equivalence to reach full abstraction?

Thank you!

Conclusion



Perspectives:

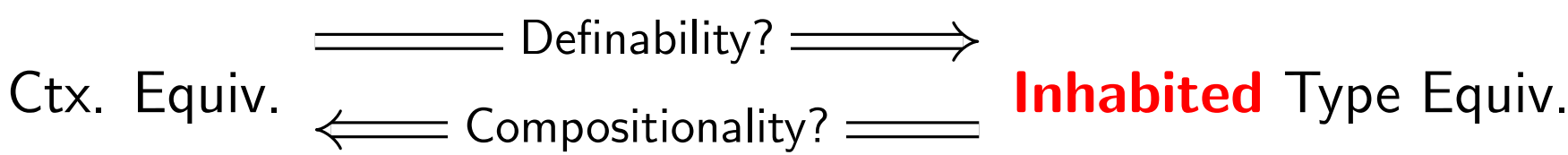
- Towards Call-by-Value equivalences, where NF bisimilarities are more involved (and Shape Typings less easy to define)
- Can we adapt this coinductive proof technique (without approximants) to the *idempotent* setting?
- Can we adapt type equivalence to reach full abstraction?

Thank you!

Conclusion

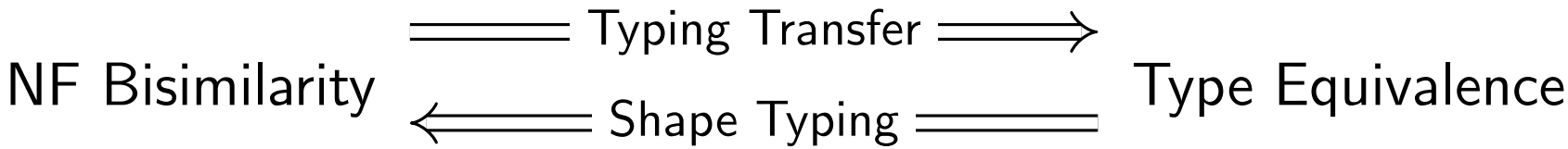
Perspectives:

- Towards Call-by-Value equivalences, where NF bisimilarities are more involved (and Shape Typings less easy to define)
- Can we adapt this coinductive proof technique (without approximants) to the *idempotent* setting?
- Can we adapt type equivalence to reach full abstraction?



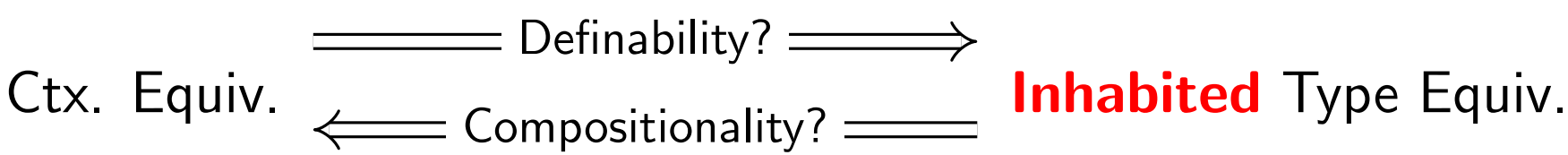
Thank you!

Conclusion



Perspectives:

- Towards Call-by-Value equivalences, where NF bisimilarities are more involved (and Shape Typings less easy to define)
- Can we adapt this coinductive proof technique (without approximants) to the *idempotent* setting?
- Can we adapt type equivalence to reach full abstraction?



Thank you!