Generic Reasoning of the Locally Nameless Representation

Yicheng Ni   Yuting Wang

Shanghai Jiao Tong University, China

APLAS, October 2024

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

$\lambda x.(z\ x) =?\ \lambda y.(z\ y)$

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names with α-equivalence

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

$\lambda x.(z\ x) =?\ \lambda y.(z\ y)$

$[\lambda x.(z\ x)]_\alpha = [\lambda y.(z\ y)]_\alpha$

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names with α-equivalence

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

de Bruijn indices

$$t := i \mid \lambda.t_1 \mid (t_1\ t_2)$$

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names with α-equivalence

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

de Bruijn indices

$$t := i \mid \lambda.t_1 \mid (t_1\ t_2)$$

$$[a, z] \quad \lambda x.(z\ x) \to \lambda.(2\ 0) = \lambda.(2\ 0) \leftarrow \lambda y.(z\ y)$$

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names with α-equivalence

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

de Bruijn indices with shift operations

$$t := i \mid \lambda.t_1 \mid (t_1\ t_2)$$

$$\begin{array}{ll} [a, z] & \lambda x.(z\ x) \to \lambda.(2\ 0) = \lambda.(2\ 0) \leftarrow \lambda y.(z\ y) \\ [a, z] & [1 \mapsto 0]\,\lambda.(2\ 0) = \lambda.\,[(1+1) \mapsto (0+1)]\,(2\ 0) = \lambda.(1\ 0) \end{array}$$

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names with α-equivalence

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

de Bruijn indices with shift operations

$$t := i \mid \lambda .t_1 \mid (t_1\ t_2)$$

Locally nameless representation (LNR)

$$t := i \mid x \mid \lambda .t_1 \mid (t_1\ t_2)$$

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names with α-equivalence

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

de Bruijn indices with shift operations

$$t := i \mid \lambda.t_1 \mid (t_1\ t_2)$$

Locally nameless representation (LNR) with locally closed relation

$$t := i \mid x \mid \lambda.t_1 \mid (t_1\ t_2)$$

$$\lambda x.(z\ x) \rightarrow \lambda.(z\ 0) = \lambda.(z\ 0) \leftarrow \lambda y.(z\ y)$$
$$[z \mapsto a]\, \lambda.(z\ 0) = \lambda.\, [z \mapsto a]\, (z\ 0) = \lambda.(a\ 0)$$

Context

Fundamental issue

Treatment of binding structure and variables

Explicit names with α-equivalence

$$t := x \mid \lambda x.t_1 \mid (t_1\ t_2)$$

de Bruijn indices with shift operations

$$t := i \mid \lambda .t_1 \mid (t_1\ t_2)$$

Locally nameless representation (LNR) with locally closed relation

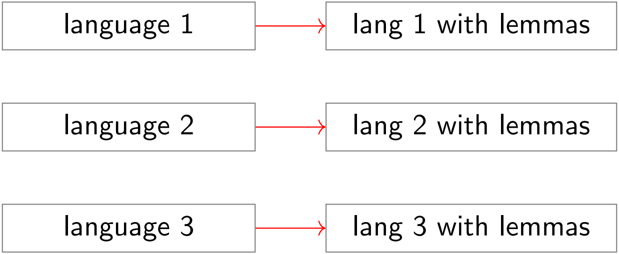
$$t := i \mid x \mid \lambda .t_1 \mid (t_1\ t_2)$$

- type soundness for System $F_{<:}$ and core ML
- subject reduction for CoC

"Out of a total of around 550 lemmas, 400 were tedious infrastructure lemmas (for LNR)."

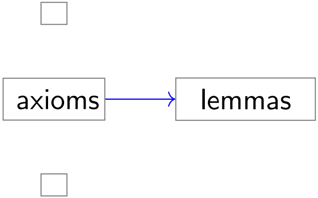
– Rossberg, A., Russo, C.V., Dreyer, D.: F-ing modules.

Motivation



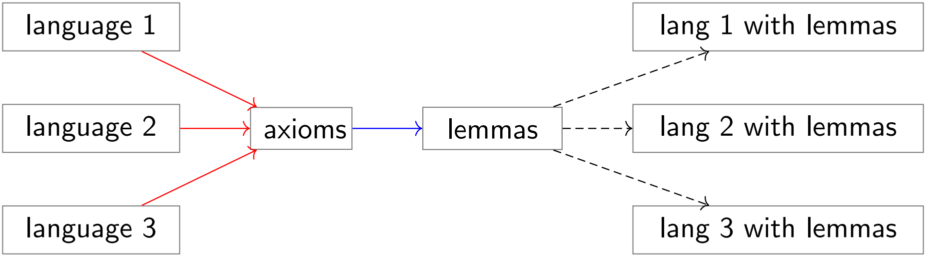
$\longrightarrow$: prove lemmas about locally nameless representation

Contribution: Generic Reasoning of LNR



⟶ : derive lemmas from axioms

Contribution: Generic Reasoning of LNR



⟶ : derive lemmas from axioms

⟶ : prove axioms

⇢ : instantiate with the language, automatically

Contribution

- Formalized theory for generic reasoning of LNR
- Applications of our generic reasoning library
  - equivalence properties of STLC
  - compactness theorem for PCF

Plan

- Challenges for adopting LNR (Part 0)
- Our approach and applications (Part 1-3)

Operations and Predicates

	Inductive	Example
open	$\{i \to x\}t$	
close	$\{i \leftarrow x\}t$	
freshness	$x \notin fv(t)$	
locally closed	$(lc\ t)$	
substitution	$[x \mapsto t_2]t_1$	
open-term	$\{i \mapsto t_2\}t_1$	

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive	Example
open	$\{i \to x\}t$	$\{0 \to x\}\lambda.1 = \lambda.\{1 \to x\}1 = \lambda.x$
close	$\{i \leftarrow x\}t$	
freshness	$x \notin fv(t)$	
locally closed	$(lc\ t)$	
substitution	$[x \mapsto t_2]t_1$	
open-term	$\{i \mapsto t_2\}t_1$	

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive	Example
open	$\{i \to x\}t$	$\{0 \to x\}\lambda.1 = \lambda.\{1 \to x\}1 = \lambda.x$
close	$\{i \leftarrow x\}t$	$\{0 \leftarrow x\}\lambda.x = \lambda.\{1 \leftarrow x\}x = \lambda.1$
freshness	$x \notin fv(t)$	
locally closed	$(lc\ t)$	
substitution	$[x \mapsto t_2]t_1$	
open-term	$\{i \mapsto t_2\}t_1$	

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive	Example
open	$\{i \to x\}t$	$\{0 \to x\}\lambda.1 = \lambda.\{1 \to x\}1 = \lambda.x$
close	$\{i \leftarrow x\}t$	$\{0 \leftarrow x\}\lambda.x = \lambda.\{1 \leftarrow x\}x = \lambda.1$
freshness	$x \notin fv(t)$	$x \notin fv(\lambda.(y\ z))$
locally closed	$(lc\ t)$	
substitution	$[x \mapsto t_2]t_1$	
open-term	$\{i \mapsto t_2\}t_1$	

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive	Example
open	$\{i \to x\}t$	$\{0 \to x\}\lambda.1 = \lambda.\{1 \to x\}1 = \lambda.x$
close	$\{i \leftarrow x\}t$	$\{0 \leftarrow x\}\lambda.x = \lambda.\{1 \leftarrow x\}x = \lambda.1$
freshness	$x \notin fv(t)$	$x \notin fv(\lambda.(y\ z))$
locally closed	$(lc\ t)$	$lc\ \lambda.0\ but\ \neg(lc\ 0)$
substitution	$[x \mapsto t_2]t_1$	
open-term	$\{i \mapsto t_2\}t_1$	

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive	Example
open	$\{i \to x\}t$	$\{0 \to x\}\lambda.1 = \lambda.\{1 \to x\}1 = \lambda.x$
close	$\{i \leftarrow x\}t$	$\{0 \leftarrow x\}\lambda.x = \lambda.\{1 \leftarrow x\}x = \lambda.1$
freshness	$x \notin fv(t)$	$x \notin fv(\lambda.(y\ z))$
locally closed	$(lc\ t)$	$lc\ \lambda.0$ *but* $\neg(lc\ 0)$
substitution	$[x \mapsto t_2]t_1$	$[x \mapsto t_2](\lambda.x) = (\lambda.t_2)$
open-term	$\{i \mapsto t_2\}t_1$	

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive	Example
open	$\{i \to x\}t$	$\{0 \to x\}\lambda.1 = \lambda.\{1 \to x\}1 = \lambda.x$
close	$\{i \leftarrow x\}t$	$\{0 \leftarrow x\}\lambda.x = \lambda.\{1 \leftarrow x\}x = \lambda.1$
freshness	$x \notin fv(t)$	$x \notin fv(\lambda.(y\ z))$
locally closed	$(lc\ t)$	$lc\ \lambda.0\ but\ \neg(lc\ 0)$
substitution	$[x \mapsto t_2]t_1$	$[x \mapsto t_2](\lambda.x) = (\lambda.t_2)$
open-term	$\{i \mapsto t_2\}t_1$	$\{0 \mapsto t_2\}\lambda.1 = \lambda.t_2$

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive
open	$\{i \to x\}t$
close	$\{i \leftarrow x\}t$
freshness	$x \notin fv(t)$
locally closed	$(lc\ t)$
substitution	$[x \mapsto t_2]t_1$
open-term	$\{i \mapsto t_2\}t_1$

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive	Abstract	
open	$\{i \to x\}t$	$\{\_ \to \_\}\_$	$:\mathbf{N} \times \mathbf{A} \times \mathbf{X} \to \mathbf{X}$
close	$\{i \leftarrow x\}t$	$\{\_ \leftarrow \_\}\_$	$:\mathbf{N} \times \mathbf{A} \times \mathbf{X} \to \mathbf{X}$
freshness	$x \notin fv(t)$	$a \# t$	$:=\{0 \leftarrow a\}t = t$
locally closed	$(lc\ t)$	$i \succ t$	$:=\forall j \geq i, \exists a \in \mathbf{A}, \{j \to a\}t = t$
substitution	$[x \mapsto t_2]t_1$		
open-term	$\{i \mapsto t_2\}t_1$		

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Operations and Predicates

	Inductive	Abstract	
open	$\{i \to x\}t$	$\{\_ \to \_\}\_$	$: \mathbf{N} \times \mathbf{A} \times \mathbf{X} \to \mathbf{X}$
close	$\{i \leftarrow x\}t$	$\{\_ \leftarrow \_\}\_$	$: \mathbf{N} \times \mathbf{A} \times \mathbf{X} \to \mathbf{X}$
freshness	$x \notin fv(t)$	$a \# t$	$:= \{0 \leftarrow a\}t = t$
locally closed	$(lc\ t)$	$i \succ t$	$:= \forall j \geq i, \exists a \in \mathbf{A}, \{j \to a\}t = t$
substitution	$[x \mapsto t_2]t_1$	$\{\_ \mapsto \_\}\_$	$: \mathbf{N} \times \mathbf{X} \times \mathbf{X} \to \mathbf{X}$
open-term	$\{i \mapsto t_2\}t_1$	$[\_ \mapsto \_]\ \_$	$: \mathbf{A} \times \mathbf{X} \times \mathbf{X} \to \mathbf{X}$

*We shall write t^x for $\{0 \to x\}t$, t/x for $\{0 \leftarrow x\}t$ and t^u for $\{0 \mapsto u\}t$.

Challenges for Adopting LNR

open_term_lc:	$lc\ t_1$	$\Rightarrow t_1 = t_1^{t_2}$
lc_open_term:	$(\forall x \notin L, lc\ t_1^x) \wedge lc\ t_2$	$\Rightarrow lc\ t_1^{t_2}$
subst_lc:	$lc\ t_1 \wedge lc\ t_2$	$\Rightarrow lc\ ([x \mapsto t_1]t_2)$
subst_open_var:	$x \neq y \wedge lc\ u$	$\Rightarrow [x \mapsto u](t^y) = ([x \mapsto u]t)^y$
subst_intro:	$x \notin fv(t) \wedge lc\ u$	$\Rightarrow t^u = [x \mapsto u]t^x$
subst_as_close_open:	$lc\ t_1 \wedge lc\ t_2$	$\Rightarrow [x \mapsto t_1]t_2 = (t_2/x)^{t_1}$

– Charguéraud, A.: The Locally Nameless Representation.

Challenges for Adopting LNR

open_term_lc:	$lc\ t_1$	$\Rightarrow t_1 = t_1^{t_2}$
lc_open_term:	$(\forall x \notin L, lc\ t_1^x) \wedge lc\ t_2$	$\Rightarrow lc\ t_1^{t_2}$
subst_lc:	$lc\ t_1 \wedge lc\ t_2$	$\Rightarrow lc\ ([x \mapsto t_1]t_2)$
subst_open_var:	$x \neq y \wedge lc\ u$	$\Rightarrow [x \mapsto u](t^y) = ([x \mapsto u]t)^y$
subst_intro:	$x \notin fv(t) \wedge lc\ u$	$\Rightarrow t^u = [x \mapsto u]t^x$
subst_as_close_open:	$lc\ t_1 \wedge lc\ t_2$	$\Rightarrow [x \mapsto t_1]t_2 = (t_2/x)^{t_1}$

– Charguéraud, A.: The Locally Nameless Representation.

Challenges for Adopting LNR

$$lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$$

Challenges for Adopting LNR

$$lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$$

In STLC:

$$\frac{}{lc\ x}\ \text{LC-VAR} \qquad \frac{lc\ t_1 \quad lc\ t_2}{lc\ (t_1\ t_2)}\ \text{LC-APP} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \lambda.t}\ \text{LC-ABS}$$

Challenges for Adopting LNR

$$lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$$

In PCF:

$$\frac{}{lc\ x}\ \text{LC-VAR} \qquad \frac{lc\ t_1 \quad lc\ t_2}{lc\ (t_1\ t_2)}\ \text{LC-APP} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \lambda.t}\ \text{LC-ABS}$$

$$\frac{}{lc\ z}\ \text{LC-Z} \qquad \frac{lc\ t}{lc\ s(t)}\ \text{LC-S} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \mathsf{fix}.t}\ \text{LC-FIX}$$

$$\frac{lc\ t \quad lc\ t_0 \quad \boxed{\forall x \notin L,}\ lc\ (t_1^x)}{lc\ (ifz\ t\ t_0\ t_1)}\ \text{LC-IFZ}$$

Challenges for Adopting LNR

$$lc\ t_1 \land lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$$

In PCF:

$$\frac{}{lc\ x}\ \text{LC-VAR} \qquad \frac{lc\ t_1 \quad lc\ t_2}{lc\ (t_1\ t_2)}\ \text{LC-APP} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \lambda.t}\ \text{LC-ABS}$$

$$\frac{}{lc\ z}\ \text{LC-Z} \qquad \frac{lc\ t}{lc\ s(t)}\ \text{LC-S} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \text{fix}.t}\ \text{LC-FIX}$$

$$\frac{lc\ t \quad lc\ t_0 \quad \boxed{\forall x \notin L,}\ lc\ (t_1^x)}{lc\ (ifz\ t\ t_0\ t_1)}\ \text{LC-IFZ}$$

Challenges for Adopting LNR

$$lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$$

In PCF:

$$\frac{}{lc\ x}\ \text{LC-VAR} \qquad \frac{lc\ t_1 \quad lc\ t_2}{lc\ (t_1\ t_2)}\ \text{LC-APP} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \lambda.t}\ \text{LC-ABS}$$

$$\frac{}{lc\ z}\ \text{LC-Z} \qquad \frac{lc\ t}{lc\ s(t)}\ \text{LC-S} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \text{fix}.t}\ \text{LC-FIX}$$

$$\frac{lc\ t \quad lc\ t_0 \quad \boxed{\forall x \notin L,}\ lc\ (t_1^x)}{lc\ (ifz\ t\ t_0\ t_1)}\ \text{LC-IFZ}$$

Challenges for Adopting LNR

$$lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$$

In PCF:

$$\frac{}{lc\ x}\ \text{LC-VAR} \qquad \frac{lc\ t_1 \quad lc\ t_2}{lc\ (t_1\ t_2)}\ \text{LC-APP} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \lambda.t}\ \text{LC-ABS}$$

$$\frac{}{lc\ z}\ \text{LC-Z} \qquad \frac{lc\ t}{lc\ s(t)}\ \text{LC-S} \qquad \frac{\boxed{\forall x \notin L,}\ lc\ (t^x)}{lc\ \text{fix}.t}\ \text{LC-FIX}$$

$$\frac{lc\ t \quad lc\ t_0 \quad \boxed{\forall x \notin L,}\ lc\ (t_1^x)}{lc\ (ifz\ t\ t_0\ t_1)}\ \text{LC-IFZ}$$

Challenges for Adopting LNR

More constructs, more proofs!

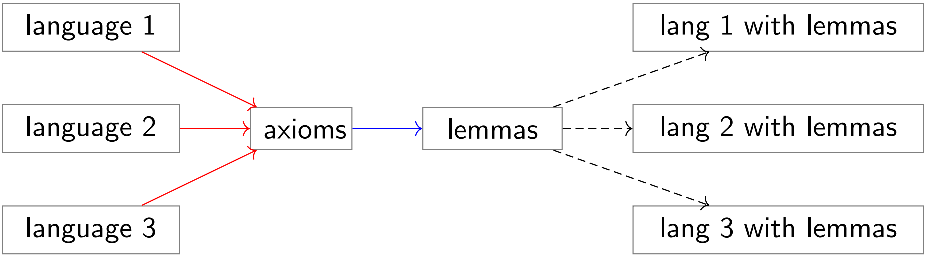
Challenges for Adopting LNR

More constructs, more proofs!
System-F or ML? double proofs!

Challenges for Adopting LNR

More constructs, more proofs!
System-F or ML? double proofs!
Can we avoid inductive reasoning?

Part I



⟶ : derive lemmas from axioms

Equational Reasoning

To derive **open_term_lc** : $lc\ t_1 \Rightarrow t_1^{t_2} = t_1$

$$\dfrac{}{lc\ t_1 \Rightarrow t_1^{t_2} = t_1}$$

Equational Reasoning

To derive **open_term_lc** : $lc\ t_1 \Rightarrow t_1^{t_2} = t_1$

$$\cfrac{\overline{i \succ t_1 \Rightarrow \{i \mapsto t_2\} t_1 = t_1}}{lc\ t_1 \Rightarrow t_1^{t_2} = t_1}$$

Equational Reasoning

To derive **open_term_lc** : $lc\ t_1 \Rightarrow t_1^{t_2} = t_1$

$$\cfrac{\cfrac{\cfrac{\cfrac{[i \succ t_1]}{\quad}}{\{i \mapsto t_2\}t_1 = t_1}}{i \succ t_1 \Rightarrow \{i \mapsto t_2\}t_1 = t_1}}{lc\ t_1 \Rightarrow t_1^{t_2} = t_1}$$

Equational Reasoning

To derive **open_term_lc** : $lc\ t_1 \Rightarrow t_1^{t_2} = t_1$

$$\dfrac{\dfrac{\dfrac{\dfrac{[i \succ t_1]}{\{i \rightarrow a\} t_1 = t_1}}{\{i \mapsto t_2\} t_1 = t_1}}{i \succ t_1 \Rightarrow \{i \mapsto t_2\} t_1 = t_1}}{lc\ t_1 \Rightarrow t_1^{t_2} = t_1}$$

To derive **open_term_lc** : $lc\ t_1 \Rightarrow t_1^{t_2} = t_1$

$$\cfrac{\cfrac{\cfrac{\cfrac{[i \succ t_1]}{\{i \rightarrow a\}t_1 = t_1} \quad < \mathbf{OT_1} >}{\{i \mapsto t_2\}t_1 = t_1}}{i \succ t_1 \Rightarrow \{i \mapsto t_2\}t_1 = t_1}}{lc\ t_1 \Rightarrow t_1^{t_2} = t_1}$$

$$\mathbf{OT_1} : \{i \mapsto t_2\}\{i \rightarrow a\}t_1 = \{i \rightarrow a\}t_1$$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\frac{\frac{\frac{\frac{}{}}{}}{}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\dfrac{\overline{\hspace{6cm}}}{\overline{\hspace{8cm}}}$$

$$\dfrac{\overline{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\qquad}{\qquad}}{i \succ [x \mapsto t_1]t_2}}{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{}}{}\ \mathit{saco}}{i \succ [x \mapsto t_1]t_2}}{\dfrac{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}}
$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\dfrac{\dfrac{\dfrac{\overline{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ [x \mapsto t_1]t_2}\ saco}{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\cfrac{\cfrac{\cfrac{\cfrac{\overline{\forall j \geq i, \{j \to y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ [x \mapsto t_1]t_2}\ \mathit{saco}}{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{j = i, \{i \to y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2}\ saco}{i \succ [x \mapsto t_1]t_2}}{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{[i \succ t_1] \quad < \mathbf{OT_2} >}{j = i, \{i \rightarrow y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2} \; saco}{i \succ [x \mapsto t_1]t_2}}{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$
$\mathbf{OT_2}$: $i \succ t_1 \Rightarrow \{i \rightarrow y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}t$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{\forall j > i, \{j \to y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ [x \mapsto t_1]t_2}\ saco}{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$

$\mathbf{OT_2}$: $i \succ t_1 \Rightarrow \{i \to y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}t$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\cfrac{\cfrac{\cfrac{\cfrac{< \mathbf{OT_3} >}{\forall j > i, \{j \to y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2} \; saco}{i \succ [x \mapsto t_1]t_2}}{\cfrac{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}}$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$

$\mathbf{OT_2}$: $i \succ t_1 \Rightarrow \{i \to y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}t$

$\mathbf{OT_3}$: $j \neq i \wedge j \succ t_1 \Rightarrow \{j \to y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}\{j \to y\}t$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{i \mapsto t_1\}\{j \to y\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}\ <\mathbf{OT_3}>}{\forall j > i, \{j \to y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2}\ saco}{i \succ [x \mapsto t_1]t_2}}{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$
OT₂ : $i \succ t_1 \Rightarrow \{i \to y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}t$
OT₃ : $j \neq i \wedge j \succ t_1 \Rightarrow \{j \to y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}\{j \to y\}t$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{<\mathbf{OC_7}>}{\{i \mapsto t_1\}\{j \to y\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2} \quad <\mathbf{OT_3}>}{\forall j > i, \{j \to y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2} \quad saco}{i \succ [x \mapsto t_1]t_2}}{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$
$\mathbf{OT_2}$: $i \succ t_1 \Rightarrow \{i \to y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}t$
$\mathbf{OT_3}$: $j \neq i \wedge j \succ t_1 \Rightarrow \{j \to y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}\{j \to y\}t$
$\mathbf{OC_7}$: $j \neq i \wedge y \neq x \Rightarrow \{j \to y\}\{i \leftarrow x\}t = \{i \leftarrow x\}\{j \to y\}t$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\{i \mapsto t_1\}\{i \leftarrow x\}\{j \rightarrow y\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2} < \mathbf{OC_7} >}{\{i \mapsto t_1\}\{j \rightarrow y\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2} < \mathbf{OT_3} >}{\forall j > i, \{j \rightarrow y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2}}{i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2} \quad s}{i \succ [x \mapsto t_1]t_2}}{\cfrac{i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2}{lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)}}$$

$subst\_as\_close\_open : i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$
$\mathbf{OT_2} : i \succ t_1 \Rightarrow \{i \rightarrow y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}t$
$\mathbf{OT_3} : j \neq i \wedge j \succ t_1 \Rightarrow \{j \rightarrow y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}\{j \rightarrow y\}t$
$\mathbf{OC_7} : j \neq i \wedge y \neq x \Rightarrow \{j \rightarrow y\}\{i \leftarrow x\}t = \{i \leftarrow x\}\{j \rightarrow y\}t$

Equational Reasoning 2

To derive **subst_lc** : $lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)$

$$
\begin{array}{c}
[i \succ t_2] \quad [j > i] \\
\hline
j \succ t_2 \\
\hline
\{i \mapsto t_1\}\{i \leftarrow x\}\{j \rightarrow y\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2 \quad <\mathbf{OC_7}> \\
\hline
\{i \mapsto t_1\}\{j \rightarrow y\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2 \quad <\mathbf{OT_3}> \\
\hline
\forall j > i, \{j \rightarrow y\}\{i \mapsto t_1\}\{i \leftarrow x\}t_2 = \{i \mapsto t_1\}\{i \leftarrow x\}t_2 \\
\hline
i \succ \{i \mapsto t_1\}\{i \leftarrow x\}t_2 \quad s \\
\hline
i \succ [x \mapsto t_1]t_2 \\
\hline
i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ [x \mapsto t_1]t_2 \\
\hline
lc\ t_1 \wedge lc\ t_2 \Rightarrow lc\ ([x \mapsto t_1]t_2)
\end{array}
$$

subst_as_close_open : $i \succ t_1 \wedge i \succ t_2 \Rightarrow [x \mapsto t_1]t_2 = \{i \mapsto t_2\}\{i \leftarrow x\}t_1$

OT₂ : $i \succ t_1 \Rightarrow \{i \rightarrow y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}t$

OT₃ : $j \neq i \wedge j \succ t_1 \Rightarrow \{j \rightarrow y\}\{i \mapsto t_1\}t = \{i \mapsto t_1\}\{j \rightarrow y\}t$

OC₇ : $j \neq i \wedge y \neq x \Rightarrow \{j \rightarrow y\}\{i \leftarrow x\}t = \{i \leftarrow x\}\{j \rightarrow y\}t$

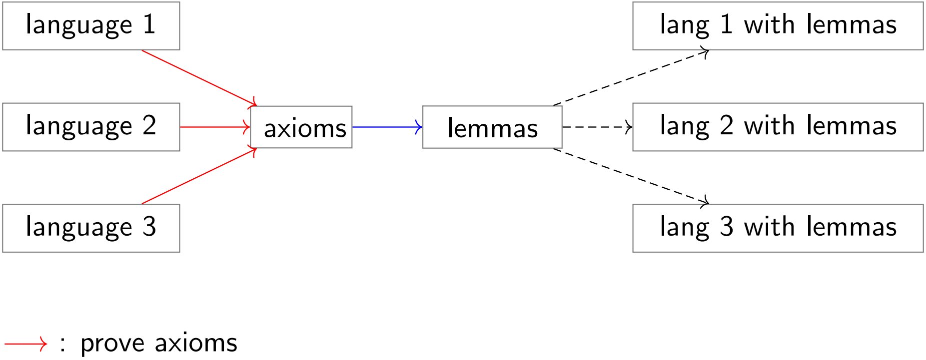
Axioms and Lemmas

$\mathbf{OC_{1-7}}$		axioms in locally nameless sets
$\mathbf{OT_1}$		$\Rightarrow \{i \mapsto u\}\{i \to a\}t = \{i \to a\}t$
$\mathbf{OT_2}$	$i \succ u$	$\Rightarrow \{i \to a\}\{i \mapsto u\}t = \{i \mapsto u\}t$
$\mathbf{OT_3}$	$i \neq j \wedge i \succ u$	$\Rightarrow \{i \to a\}\{j \mapsto u\}t = \{j \mapsto u\}\{i \to a\}t$
$\mathbf{OT_4}$:		$\Rightarrow \{i \mapsto \mathit{fvar}\ a\}t = \{i \to a\}t$
$\mathbf{S_1}$:	$x \# t_2$	$\Rightarrow [x \mapsto t_1]t_2 = t_2$
$\mathbf{S_2}$:	$i \succ t_1$	$\Rightarrow [x \mapsto t_1]\{i \mapsto t_3\}t_2 =$ $\{i \mapsto [x \mapsto t_1]t_3\}([x \mapsto t_1]t_2)$
$\mathbf{S_3}$:		$\Rightarrow [a \mapsto t]\,(\mathit{fvar}\ a) = t$
$\mathbf{S_4}$:	$b \neq a$	$\Rightarrow [a \mapsto t](\mathit{fvar}\ b) = \mathit{fvar}\ b$

Axioms and Lemmas

open_term_lc:	$i \succ t$	$\Rightarrow t = \{i \mapsto y\}t$
lc_open_term:	$(i+1) \succ t \wedge i \succ y$	$\Rightarrow i \succ \{i \mapsto y\}t$
subst_intro:	$x \# t \wedge i \succ u$	$\Rightarrow \{i \mapsto u\}t =$ $[x \mapsto u](\{i \to x\}t)$
subst_lc:	$i \succ t \wedge i \succ u$	$\Rightarrow i \succ [x \mapsto u]t$
subst_as_close_open:	$i \succ u \wedge i \succ t$	$\Rightarrow [x \mapsto u]t = \{i \mapsto u\}\{i \leftarrow x\}$
subst_open_var:	$y \neq x \wedge 0 \succ u$	$\Rightarrow ([x \mapsto u]t)^y = [x \mapsto u](t^y)$
	$\ldots$	

Part II


language 1
language 2
language 3
axioms
lemmas
lang 1 with lemmas
lang 2 with lemmas
lang 3 with lemmas
: prove axioms

New Problem

To prove

$$\mathbf{S_1} : x \# t \Rightarrow [x \mapsto u]t = t$$

Proof. Induction on the structure of t

Case App : $t = (t_1\ t_2)$

New Problem

To prove

$$\mathbf{S_1} : x \# t \Rightarrow [x \mapsto u]t = t$$

Proof. Induction on the structure of t

Case App : $t = (t_1\ t_2)$

$$x \# (t_1\ t_2) \Rightarrow x \# t_1 \wedge x \# t_2?$$

New Problem

In generic setting,

$$x \# (t_1\ t_2) \quad \Rightarrow \quad x \# t_1 \wedge x \# t_2?$$

New Problem

In generic setting,

$$
\begin{array}{lcl}
x \mathrel{\#} (t_1\ t_2) & \Rightarrow & x \mathrel{\#} t_1 \wedge x \mathrel{\#} t_2? \\
x \mathrel{\#} \lambda.t & \Rightarrow & x \mathrel{\#} t?
\end{array}
$$

New Problem

In generic setting,

$$\begin{array}{lcl} x \# (t_1\ t_2) & \Rightarrow & x \# t_1 \wedge x \# t_2? \\ x \# \lambda.t & \Rightarrow & x \# t? \end{array}$$

In traditional LNR,

$$fv := i \mapsto \emptyset \mid x \mapsto \{x\} \mid (t_1\ t_2) \mapsto fv(t_1) \cup fv(t_2) \mid \lambda.t \mapsto fv(t)$$

$$\begin{array}{lcl} x \notin fv(t_1\ t_2) & \Rightarrow & x \notin fv(t_1) \wedge x \notin fv(t_2) \\ x \notin fv(\lambda.t) & \Rightarrow & x \notin fv(t) \end{array}$$

New Problem

In generic setting,

$$\begin{array}{lcl} x \# (t_1\ t_2) & \Rightarrow & x \# t_1 \wedge x \# t_2? \\ x \# \lambda.t & \Rightarrow & x \# t? \end{array}$$

In traditional LNR,

$$fv := i \mapsto \emptyset \mid x \mapsto \{x\} \mid (t_1\ t_2) \mapsto fv(t_1) \cup fv(t_2) \mid \lambda.t \mapsto fv(t)$$

$$\begin{array}{lcl} x \notin fv(t_1\ t_2) & \Rightarrow & x \notin fv(t_1) \wedge x \notin fv(t_2) \\ x \notin fv(\lambda.t) & \Rightarrow & x \notin fv(t) \end{array}$$

New Problem

In generic setting,

$$\begin{array}{lcl} x \# (t_1\ t_2) & \Rightarrow & x \# t_1 \wedge x \# t_2? \\ x \# \lambda.t & \Rightarrow & x \# t? \end{array}$$

In traditional LNR,

$$fv := i \mapsto \emptyset \mid x \mapsto \{x\} \mid (t_1\ t_2) \mapsto fv(t_1) \cup fv(t_2) \mid \lambda.t \mapsto fv(t)$$

$$\begin{array}{lcl} x \notin fv(t_1\ t_2) & \Rightarrow & x \notin fv(t_1) \wedge x \notin fv(t_2) \\ x \notin fv(\lambda.t) & \Rightarrow & x \notin fv(t) \end{array}$$

We lost lemmas generated by *fv* and *lc*

Generalization of Morphisms

Restore *lc* and *fv* in the generic setting

- construction – e.g., $i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ (\textbf{app}\ t_1\ t_2)$
- destruction – e.g., $x \# (\textbf{app}\ t_1\ t_2) \Rightarrow x \# t_1 \wedge x \# t_2$

Generalization of Morphisms

Restore *lc* and *fv* in the generic setting

- construction – e.g., $i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ (\mathbf{app}\ t_1\ t_2)$
- destruction – e.g., $x \# (\mathbf{app}\ t_1\ t_2) \Rightarrow x \# t_1 \wedge x \# t_2$

LNS: the constructs of the language are instances of morphisms
Morphism $f : \mathbf{X} \to \mathbf{X}$

$$\begin{array}{lcl} f(\{i \to a\}t) = \{i \to a\}(f\ t) & \xrightarrow{imply} & a \# t \Rightarrow a \# f\ t \\ f(\{i \leftarrow a\}t) = \{i \leftarrow a\}(f\ t) & & i \succ t \Rightarrow i \succ f\ t \end{array}$$

Generalization of Morphisms

Restore *lc* and *fv* in the generic setting

- construction – e.g., $i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ (\textbf{app}\ t_1\ t_2)$
- destruction – e.g., $x \# (\textbf{app}\ t_1\ t_2) \Rightarrow x \# t_1 \wedge x \# t_2$

LNS: the constructs of the language are instances of morphisms

Binary Morphism $g : \mathbf{X} \times \mathbf{X} \rightarrow \mathbf{X}$

$$\begin{array}{l} g(\{i \rightarrow a\}t_1)(\{i \rightarrow a\}t_2) = \{i \rightarrow a\}(g\ t_1\ t_2) \\ g(\{i \leftarrow a\}t_1)(\{i \leftarrow a\}t_2) = \{i \leftarrow a\}(g\ t_1\ t_2) \end{array} \xrightarrow{imply} \begin{array}{l} a \# t_1 \wedge a \# t_2 \Rightarrow a \# g\ t_1\ t_2 \\ i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ g\ t_1\ t_2 \end{array}$$

Generalization of Morphisms

Restore *lc* and *fv* in the generic setting

- construction – e.g., $i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ (\textbf{app}\ t_1\ t_2)$
- destruction – e.g., $x \# (\textbf{app}\ t_1\ t_2) \Rightarrow x \# t_1 \wedge x \# t_2$

LNS: the constructs of the language are instances of morphisms

Binary Morphism $g : \mathbf{X} \times \mathbf{X} \rightarrow \mathbf{X}$

$$\begin{array}{lcl} g(\{i \rightarrow a\}t_1)(\{i \rightarrow a\}t_2) = \{i \rightarrow a\}(g\ t_1\ t_2) & & a \# t_1 \wedge a \# t_2 \Rightarrow a \# g\ t_1\ t_2 \\ g(\{i \leftarrow a\}t_1)(\{i \leftarrow a\}t_2) = \{i \leftarrow a\}(g\ t_1\ t_2) & \xrightarrow{imply} & i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ g\ t_1\ t_2 \end{array}$$

Generalization of Morphisms

Restore *lc* and *fv* in the generic setting

- construction – e.g., $i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ (\textbf{app}\ t_1\ t_2)$
- destruction – e.g., $x \# (\textbf{app}\ t_1\ t_2) \Rightarrow x \# t_1 \wedge x \# t_2$

LNS: the constructs of the language are instances of morphisms

Injective Binary Morphism $g : \mathbf{X} \times \mathbf{X} \to \mathbf{X}$

$$
\begin{array}{l}
g(\{i \to a\}t_1)(\{i \to a\}t_2) = \{i \to a\}(g\ t_1\ t_2) \\
g(\{i \leftarrow a\}t_1)(\{i \leftarrow a\}t_2) = \{i \leftarrow a\}(g\ t_1\ t_2) \\
g\ t_1\ t_2 = g\ t_3\ t_4 \Rightarrow t_1 = t_3 \wedge t_2 = t_4
\end{array}
\xrightarrow{imply}
\begin{array}{l}
a \# t_1 \wedge a \# t_2 \Rightarrow a \# g\ t_1\ t_2 \\
i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ g\ t_1\ t_2 \\
a \# t_1 \wedge a \# t_2 \Leftarrow a \# g\ t_1\ t_2 \\
i \succ t_1 \wedge i \succ t_2 \Leftarrow i \succ g\ t_1\ t_2
\end{array}
$$

Generalization of Morphisms

Restore *lc* and *fv* in the generic setting

- construction – e.g., $i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ (\textbf{app}\ t_1\ t_2)$
- destruction – e.g., $x \# (\textbf{app}\ t_1\ t_2) \Rightarrow x \# t_1 \wedge x \# t_2$

LNS: the constructs of the language are instances of morphisms

Injective Binary Morphism $g : \mathbf{X} \times \mathbf{X} \to \mathbf{X}$

$$\begin{array}{l} g(\{i \to a\}t_1)(\{i \to a\}t_2) = \{i \to a\}(g\ t_1\ t_2) \\ g(\{i \leftarrow a\}t_1)(\{i \leftarrow a\}t_2) = \{i \leftarrow a\}(g\ t_1\ t_2) \\ g\ t_1\ t_2 = g\ t_3\ t_4 \Rightarrow t_1 = t_3 \wedge t_2 = t_4 \end{array} \xrightarrow{imply} \begin{array}{l} a \# t_1 \wedge a \# t_2 \Rightarrow a \# g\ t_1\ t_2 \\ i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ g\ t_1\ t_2 \\ a \# t_1 \wedge a \# t_2 \Leftarrow a \# g\ t_1\ t_2 \\ i \succ t_1 \wedge i \succ t_2 \Leftarrow i \succ g\ t_1\ t_2 \end{array}$$

Generalization of Morphisms

Restore *lc* and *fv* in the generic setting

- construction – e.g., $i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ (\mathbf{app}\ t_1\ t_2)$
- destruction – e.g., $x \# (\mathbf{app}\ t_1\ t_2) \Rightarrow x \# t_1 \wedge x \# t_2$

LNS: the constructs of the language are instances of morphisms

Injective Binary Morphism $g : \mathbf{X} \times \mathbf{X} \rightarrow \mathbf{X}$

$$\begin{array}{l} g(\{i \rightarrow a\}t_1)(\{i \rightarrow a\}t_2) = \{i \rightarrow a\}(g\ t_1\ t_2) \\ g(\{i \leftarrow a\}t_1)(\{i \leftarrow a\}t_2) = \{i \leftarrow a\}(g\ t_1\ t_2) \\ g\ t_1\ t_2 = g\ t_3\ t_4 \Rightarrow t_1 = t_3 \wedge t_2 = t_4 \end{array} \xrightarrow{imply} \begin{array}{l} a \# t_1 \wedge a \# t_2 \Rightarrow a \# g\ t_1\ t_2 \\ i \succ t_1 \wedge i \succ t_2 \Rightarrow i \succ g\ t_1\ t_2 \\ a \# t_1 \wedge a \# t_2 \Leftarrow a \# g\ t_1\ t_2 \\ i \succ t_1 \wedge i \succ t_2 \Leftarrow i \succ g\ t_1\ t_2 \end{array}$$

Injective Shift Morphism $h : \mathbf{X} \rightarrow \mathbf{X}$

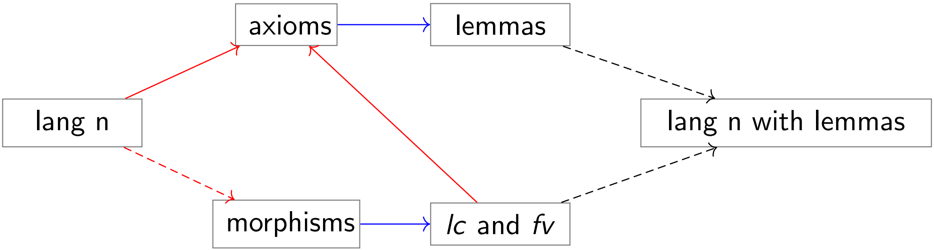
$$\begin{array}{l} h(\{(i+1) \rightarrow a\}t) = \{i \rightarrow a\}(h\ t) \\ h(\{(i+1) \leftarrow a\}t) = \{i \leftarrow a\}(h\ t) \\ g\ t_1 = g\ t_2 \Rightarrow t_1 = t_2 \end{array} \xrightarrow{imply} \begin{array}{l} a \# t \Rightarrow a \# h\ t \\ (i+1) \succ t \Rightarrow i \succ h\ t \\ a \# t \Leftarrow a \# h\ t \\ (i+1) \succ t \Leftarrow i \succ h\ t \end{array}$$

Generalization of Morphisms

Morphisms for substitution and open-term?

$$f([x \mapsto u]t) = [x \mapsto u](f\ t) \quad \xrightarrow{imply} \quad f\ (\widehat{\gamma}(t)) = \widehat{\gamma}(f\ t)$$

Extended Generic Reasoning



: prove morphisms, automatically

New Problem, Revisited

To prove

$$\mathbf{S_1} : x \,\#\, t \Rightarrow [x \mapsto u]t = t$$

Proof. Induction on the structure of t

Case App : $t = (t_1\ t_2)$

$$x \,\#\, (t_1\ t_2) \Rightarrow x \,\#\, t_1 \wedge x \,\#\, t_2$$

Embed Predicates into Equations

$$\overline{x \mathrel{\#} t \Rightarrow [x \mapsto u]t = t}$$

Embed Predicates into Equations

$$\frac{\dfrac{[x \mathrel{\#} t]}{\quad}}{\dfrac{[x \mapsto u]t = t}{x \mathrel{\#} t \Rightarrow [x \mapsto u]t = t}}$$

Embed Predicates into Equations

$$\cfrac{\cfrac{\cfrac{[x \mathrel{\#} t]}{\{i \leftarrow x\}t = t}}{[x \mapsto u]t = t}}{x \mathrel{\#} t \Rightarrow [x \mapsto u]t = t}$$

Embed Predicates into Equations

$$\dfrac{\dfrac{[x \# t]}{\{i \leftarrow x\}t = t} \quad < \mathbf{S_1'} >}{\dfrac{[x \mapsto u]t = t}{x \# t \Rightarrow [x \mapsto u]t = t}}$$

$$\mathbf{S_1'} : [x \mapsto u]\{i \leftarrow x\}t = \{i \leftarrow x\}t$$

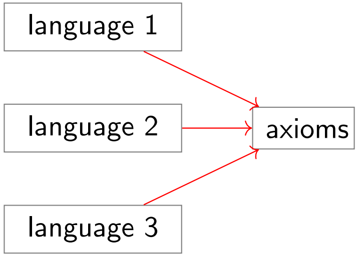
Embed Predicates into Equations

$$\dfrac{\dfrac{\dfrac{[x \# t]}{\{i \leftarrow x\}t = t} \quad < \mathbf{S_1'} >}{[x \mapsto u]t = t}}{x \# t \Rightarrow [x \mapsto u]t = t}$$

$$\mathbf{S_1'} : [x \mapsto u]\{i \leftarrow x\}t = \{i \leftarrow x\}t$$

proved by induction, no $x \# t$ appears

Tactic Programming is Generic Reasoning



How to automate axioms? By tactics.
Tactic programming is generic reasoning

Tactic Programming is Generic Reasoning

$$\mathbf{S_1'} : [x \mapsto u]\{i \leftarrow x\}t = \{i \leftarrow x\}t$$

Proof. Induction on t.

Case ABS: $t = \textbf{abs}\ t_1$, $\mathbf{S_1'}$ holds for t_1.

Goal:

$$[x \mapsto u]\{i \leftarrow x\}\textbf{abs}\ t_1 = \{i \leftarrow x\}\textbf{abs}\ t_1$$

$$\mathbf{S_1'} : [x \mapsto u]\{i \leftarrow x\}t = \{i \leftarrow x\}t$$

Proof. Induction on t.

Case ABS: $t = \mathbf{abs}\ t_1$, $\mathbf{S_1'}$ holds for t_1.

Goal:

$$[x \mapsto u]\{i \leftarrow x\}\mathbf{abs}\ t_1 = \{i \leftarrow x\}\mathbf{abs}\ t_1$$

simpl:

$$\mathbf{abs}\ ([x \mapsto u]\{(i+1) \leftarrow x\}t_1) = \mathbf{abs}\ (\{(i+1) \leftarrow x\}t_1)$$

Tactic Programming is Generic Reasoning

$$\mathbf{S_1'} : [x \mapsto u]\{i \leftarrow x\}t = \{i \leftarrow x\}t$$

Proof. Induction on t.

Case ABS: $t = \textbf{abs}\ t_1$, $\mathbf{S_1'}$ holds for t_1.

Goal:

$$[x \mapsto u]\{i \leftarrow x\}\textbf{abs}\ t_1 = \{i \leftarrow x\}\textbf{abs}\ t_1$$

simpl:

$$\textbf{abs}\ ([x \mapsto u]\{(i+1) \leftarrow x\}t_1) = \textbf{abs}\ (\{(i+1) \leftarrow x\}t_1)$$

f_equal:

$$[x \mapsto u]\{(i+1) \leftarrow x\}t_1 = \{(i+1) \leftarrow x\}t_1$$

Tactic Programming is Generic Reasoning

Abstract the previous reasoning:

Tactic Programming is Generic Reasoning

Abstract the previous reasoning:
Define $< \_ \oplus \_ >$, which is one of $\{\_ \rightarrow \_\}, \{\_ \leftarrow \_\}, \{\_ \mapsto \_\}, [\_ \mapsto \_]$.

Tactic Programming is Generic Reasoning

Abstract the previous reasoning:
Define $< \_ \oplus \_ >$, which is one of $\{\_ \to \_\}, \{\_ \leftarrow \_\}, \{\_ \mapsto \_\}, [\_ \mapsto \_]$.

$$\textbf{Axiom}:\ < \_ \oplus \_ >^n\ t = < \_ \oplus \_ >^m\ t.$$

Tactic Programming is Generic Reasoning

Abstract the previous reasoning:
Define $< \_ \oplus \_ >$, which is one of $\{\_ \rightarrow \_\}, \{\_ \leftarrow \_\}, \{\_ \mapsto \_\}, [\_ \mapsto \_]$.

$$\textbf{Axiom}:\ < \_ \oplus \_ >^n\ t = < \_ \oplus \_ >^m\ t.$$

<u>Proof.</u> Induction on t.

Case ABS: $t = \textbf{abs}\ t_1$, **Axiom** holds for t_1.

Goal:

$$< \_ \oplus \_ >^n \textbf{abs}\ t_1 =< \_ \oplus \_ >^m \textbf{abs}\ t_1$$

Tactic Programming is Generic Reasoning

Abstract the previous reasoning:
Define $< \_ \oplus \_ >$, which is one of $\{\_ \rightarrow \_\}, \{\_ \leftarrow \_\}, \{\_ \mapsto \_\}, [\_ \mapsto \_]$.

$$\textbf{Axiom}: < \_ \oplus \_ >^n t = < \_ \oplus \_ >^m t.$$

Proof. Induction on t.

Case ABS: $t = \textbf{abs}\ t_1$, **Axiom** holds for t_1.

Goal:

$$< \_ \oplus \_ >^n \textbf{abs}\ t_1 =< \_ \oplus \_ >^m \textbf{abs}\ t_1$$

definition of morphism:

$$\textbf{abs}\ (< \_ \oplus \_ >^{n'}\ t_1) = \textbf{abs}\ (< \_ \oplus \_ >^{m'}\ t_1)$$

Tactic Programming is Generic Reasoning

Abstract the previous reasoning:
Define $< \_ \oplus \_ >$, which is one of $\{\_ \rightarrow \_\}, \{\_ \leftarrow \_\}, \{\_ \mapsto \_\}, [\_ \mapsto \_]$.

$$\textbf{Axiom}:\ < \_ \oplus \_ >^n\ t = < \_ \oplus \_ >^m\ t.$$

Proof. Induction on t.

Case ABS: $t = \textbf{abs}\ t_1$, **Axiom** holds for t_1.

Goal:

$$< \_ \oplus \_ >^n\ \textbf{abs}\ t_1 =< \_ \oplus \_ >^m\ \textbf{abs}\ t_1$$

definition of morphism:

$$\textbf{abs}\ (< \_ \oplus \_ >^{n'}\ t_1) = \textbf{abs}\ (< \_ \oplus \_ >^{m'}\ t_1)$$

injectivity:

$$< \_ \oplus \_ >^{n'}\ t_1 =< \_ \oplus \_ >^{m'}\ t_1$$

Tactic Programming is Generic Reasoning

Abstract the previous reasoning:
Define $< \_ \oplus \_ >$, which is one of $\{\_ \rightarrow \_\}, \{\_ \leftarrow \_\}, \{\_ \mapsto \_\}, [\_ \mapsto \_]$.

$$\textbf{Axiom}\text{: } < \_ \oplus \_ >^n t = < \_ \oplus \_ >^m t.$$

Proof. Induction on t.

Case ABS: $t = \textbf{abs } t_1$, **Axiom** holds for t_1.
Goal:

$$< \_ \oplus \_ >^n \textbf{abs } t_1 = < \_ \oplus \_ >^m \textbf{abs } t_1$$

definition of morphism:

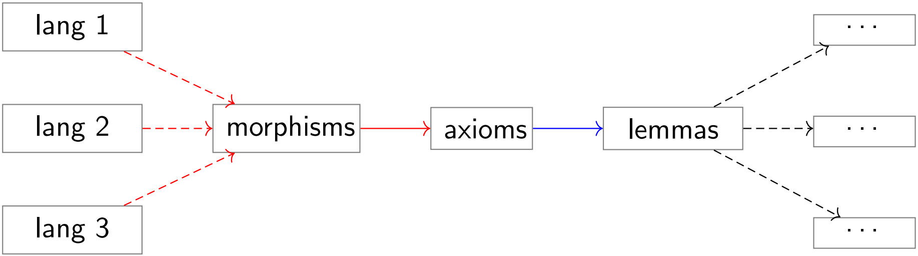
$$\textbf{abs } (< \_ \oplus \_ >^{n'} t_1) = \textbf{abs } (< \_ \oplus \_ >^{m'} t_1)$$

injectivity:

$$< \_ \oplus \_ >^{n'} t_1 = < \_ \oplus \_ >^{m'} t_1$$

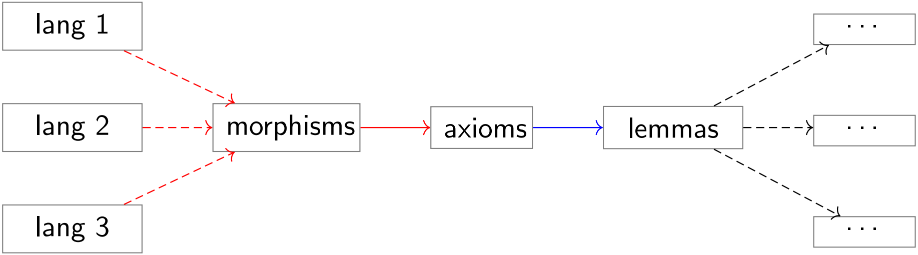
simpl: applying axioms of morphisms
f_equal: applying injectivity of morphisms

Complete Version of Generic Reasoning



- - → : prove morphisms
⟶ : prove axioms by general tactic
⟶ : derive lemmas from axioms
- - → : instantiated with the language

Part III



--→ : instantiate with the language, automatically

Application and Evaluation

- Equivalence properties of STLC
  - Normalization theorem
  - Fundamental theorem
  - Observational equivalence $\Leftrightarrow$ Logical equivalence
- Compactness theorem for PCF
  - Soundness and completeness of stack machine
  - Generalized Compactness theorem

Table: Statistics of Our Development

Components	LOC	LOC about LNR	Percentage
Base	1k	0	0
Generic Reasoning Library	1.1k	1.1k	100%
Program Equivalence for STLC	2.9k	176	6.1%
Compactness for PCF	2.9k	307	10.6%
Total (Without library)	5.8k	483	8.3%
Total	7.9k	1.6k	20.2%

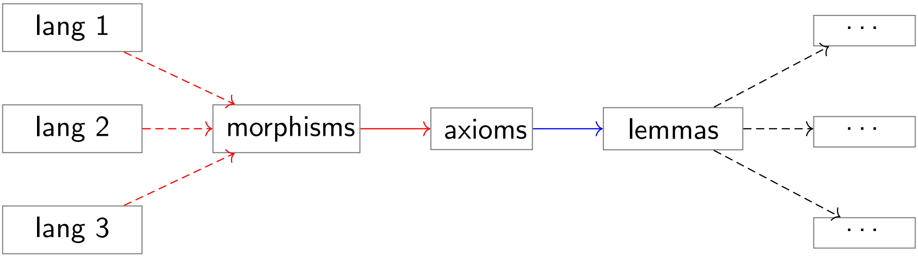
Bonus

Fix a mistake in Practical Foundations for Programming Languages

Theorem (Fixed Point Induction)

Suppose that $x : \tau \vdash e : \tau$*. If* $(\forall m \geq 0) \mathit{fix}^m x : \tau.e \sim_\tau \mathit{fix}^m x : \tau.e'$*, then* $\mathit{fix}\ x : \tau.e \sim_\tau \mathit{fix}\ x : \tau.e'$*.*

Conclusion



- A generic reasoning approach of LNR
- A formalized library in Coq and applications
  - equivalence properties of STLC
  - compactness theorem for PCF

Future work: soundness ($\dashrightarrow$) and completeness ($\longrightarrow$) of our theory