A Mechanized Bisimulation for the Nu-Calculus

Vasileios Koutavas
Trinity College Dublin
(joint work with Nick Benton)

Motivation

- We would like to understand and **formally reason** about the behavior of our programs
  - Contextual equivalence
- This **depends on the language**: can be tricky when the programming language has **side-effects** (e.g. assignment) and **higher-order features** (e.g. callbacks)

Generative names

- Entities that are created **fresh** at runtime and can be tested for identity
  - Fresh generation is a **side-effect**

Generative names

- Entities that are created **fresh** at runtime and can be tested for identity
  - Fresh generation is a **side-effect**
- They appear in numerous languages
  - **Imperative languages**: locations of objects in the heap
  - **Concurrent languages**: network channels/ports
  - **Languages with cryptographic primitives**: keys

Generative names

- Entities that are created **fresh** at runtime and can be tested for identity
  - Fresh generation is a **side-effect**
- They appear in numerous languages
  - **Imperative languages**: locations of objects in the heap
  - **Concurrent languages**: network channels/ports
  - **Languages with cryptographic primitives**: keys
- Pitts and Stark developed the **nu-calculus** to study generative names

The nu-calculus [Pitts-Stark]

e ::= **x | λx:T.e | e e** S.T. λ-calculus

T ::= **T → T**

The nu-calculus [Pitts-Stark]

e ::= x | λx:T.e | e e

| **true | false**

| **if e then e else e**

S.T. λ-calculus

w/ booleans

T ::= T → T | **o**

The nu-calculus [Pitts-Stark]

e ::= x | λx:T.e | e e	S.T. λ-calculus
| true | false	w/ booleans
| if e then e else e	
| n | new | (e = e)	& names
T ::= T → T | o **| ν**	

The nu-calculus [Pitts-Stark]

e ::= x | λx:T.e | e e — S.T. λ-calculus

| true | false — w/ booleans

| if e then e else e

| n | **new** | (e = e) — & names

T ::= T → T | o | ν

The only side-effect

The nu-calculus [Pitts-Stark]

Big-step operational semantics:

$$s; e \Downarrow (t)\, V$$

The nu-calculus [Pitts-Stark]

Big-step operational semantics:

$s; e \Downarrow (t)\ V$

Set of names

The nu-calculus [Pitts-Stark]

Big-step operational semantics:

$$s;\ e \Downarrow (t)\ V$$

Set of **fresh** names

The nu-calculus [Pitts-Stark]

Big-step operational semantics:

$$\frac{n \notin s}{s;\ \mathsf{new} \Downarrow (\{n\})\ n}$$

The nu-calculus [Pitts-Stark]

Big-step operational semantics:

$$\frac{s;\ e_1 \Downarrow (t_1)\ \lambda x.\ e_3 \qquad s \oplus t_1;\ e_2 \Downarrow (t_2)\ V_2 \qquad s \oplus t_1 \oplus t_2;\ e_3[V_2/x] \Downarrow (t_3)\ V}{s;\ e_1\ e_2 \Downarrow (t_1 \oplus t_2 \oplus t_3)\ V}$$

Behavioral Reasoning

- **Contextual Equivalence** ($\equiv$)
  - Two expressions are indistinguishable by all contexts

Behavioral Reasoning

- **Contextual Equivalence** (≡)
  - Two expressions are indistinguishable by all contexts

$e_1 \equiv e_2$ iff

$\forall$ s, C,

s; C[**e_1**] $\Downarrow$ (_) true iff s; C[**e_2**] $\Downarrow$ (_) true

Examples

let x=new in M $\quad\quad$ M

$x \notin fv(M)$

Examples

let x=new in M $\equiv$ M

$x \notin fv(M)$

Examples

let **x**=new in
let **y**=new in M

let **y**=new in
let **x**=new in M

Examples

let **x**=new in
let **y**=new in M

≡

let **y**=new in
let **x**=new in M

Examples

let y=new
in λx:ν.(x=y)

λx:ν. false

Examples

let y=new
in λx:ν.(x=y)

≡

λx:ν. false

Examples

let x_1=new in

 λf:ν→o.

  let x_2=new in

   (f x_1 = f x_2)

λf:ν→o. true

Examples

let x_1=new in
 λf:ν→o.
  let x_2=new in
   (f x_1 = f x_2)

≠

λf:ν→o. true

let F = [ ] in
 F (λx:ν.
  F (λy:ν. x=y))

Examples

let x_1=new in

let x_2=new in

$\lambda f{:}\nu \rightarrow o.$

$(f\ x_1 = f\ x_2)$

$\equiv$

$\lambda f{:}\nu \rightarrow o.$ true

Previous Approaches

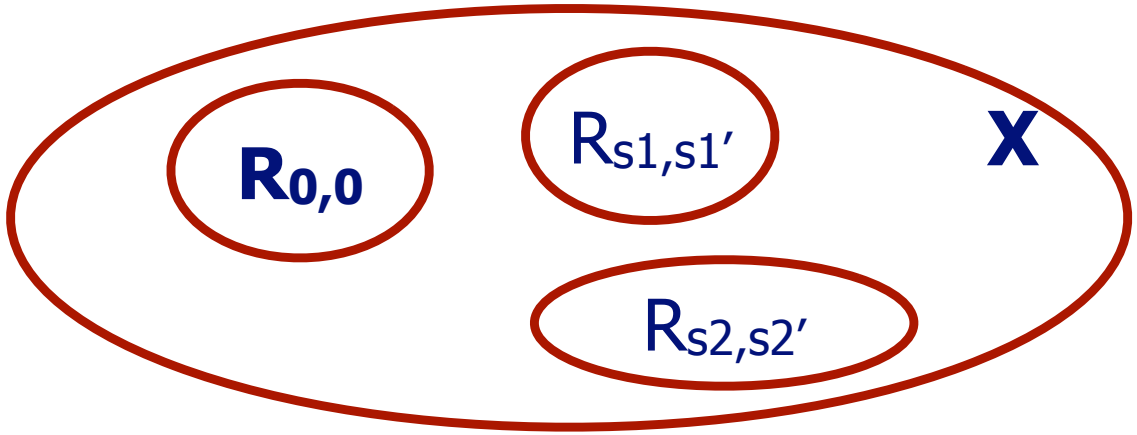
- Reasoning for such equivalences is hard, especially the last one
  - [Pitts&Stark]: operational logical relation which proves these equivalences with the exception of the last one
  - [Stark]: proved the last example with a logical relation specifically tailored to fit that equivalence
  - [Abramsky et al.]: rather complex game-semantics model of the nu-calculus which provides a proof for the last equivalence

Contributions

- **Elementary theory of bisimulations**
  - Theoretically **sound** and **complete** w.r.t. (≡)
  - Gives **simple proofs** for all equivalences of [Stark]
    - The proof of the "hard" equivalence is done by a **double use** of our method.
  - Soundness and proofs of equivalence are **mechanically verified** in the Coq theorem prover
  - Is **generally applicable** to many higher-order languages with side-effects [Koutavas&Wand][3]

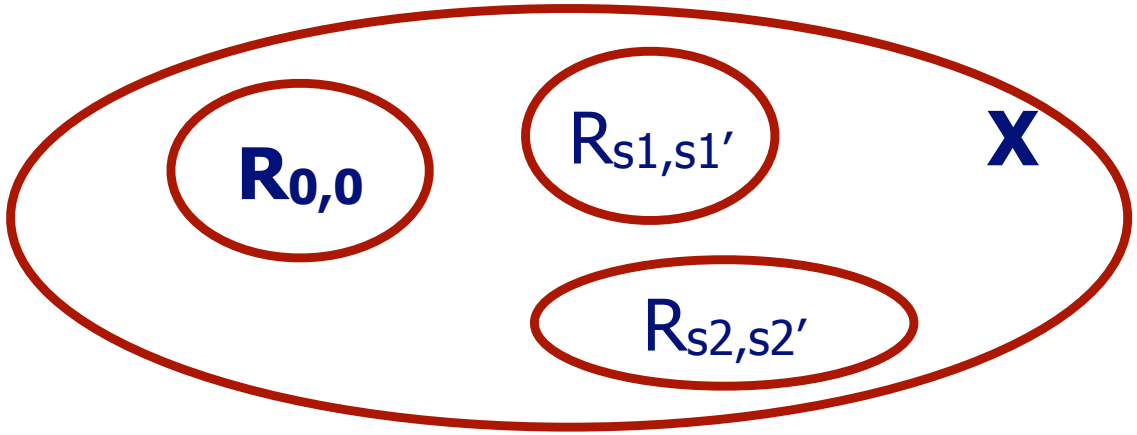
Bisimulations for the nu-calculus

- We consider **sets** of relations **annotated** by the namesets under which they hold [Sumii&Pierce]



Bisimulations for the nu-calculus

- We consider **sets** of relations **annotated** by the namesets under which they hold [Sumii&Pierce]



- We **derive** the conditions for an X to be a bisimulation similarly to [Koutavas&Wand]

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

1. $R_{s,s'}$ is the **identity on booleans**

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

1. $R_{s,s'}$ is the **identity on booleans**
2. $R_{s,s'}$ is a **bijection on names**

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

1. $R_{s,s'}$ is the **identity on booleans**
2. $R_{s,s'}$ is a **bijection on names**
3. X is **closed under allocation of fresh names**: for any fresh n, n′,

$$(R \cup \{(n,n')\})_{s \oplus \{n\}, s' \oplus \{n'\}} \in X$$

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If $(f, f') \in R_{s,s'}$, $(v, v') \in (R_{s,s'})^{CXT}$, $IH_X(k\text{-}1)$,
   and $s; f\ v \Downarrow^k (t)\ w$ then

$$s'; f'\ v' \Downarrow (t')\ w'$$

$$(w, w') \in (Q_{s \oplus t, s' \oplus t'})^{CXT}$$

$$Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}$$

$$Q_{s \oplus t, s' \oplus t'} \in X$$

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If **$(f, f') \in R_{s,s'}$** , $(v, v') \in (R_{s,s'})^{CXT}$, $IH_X(k\text{-}1)$,
   and $s; f\ v \Downarrow^k (t)\ w$ then

$$s'; f'\ v' \Downarrow (t')\ w'$$

$$(w, w') \in (Q_{s \oplus t, s' \oplus t'})^{CXT}$$

$$Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}$$

$$Q_{s \oplus t, s' \oplus t'} \in X$$

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If $(f, f') \in R_{s,s'}$, **$(v, v') \in (R_{s,s'})^{CXT}$**, $IH_X(k\text{-}1)$,
   and $s; f\ v \Downarrow^k (t)\ w$ then
   $s'; f'\ v' \Downarrow (t')\ w'$
   $(w, w') \in (Q_{s \oplus t, s' \oplus t'})^{CX}$
   $Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}$
   $Q_{s \oplus t, s' \oplus t'} \in X$

$v = C[v_1, \ldots]$
$v' = C[v_1', \ldots]$
$(v_1, v_1'), \ldots \in R_{s,s'}$

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If $(f, f') \in R_{s,s'}$, $(v, v') \in (R_{s,s'})^{CXT}$, $IH_X(k\text{-}1)$,
   and **$s; f\ v \Downarrow^k (t)\ w$** then

$$s'; f'\ v' \Downarrow (t')\ w'$$

$$(w, w') \in (Q_{s \oplus t, s' \oplus t'})^{CXT}$$

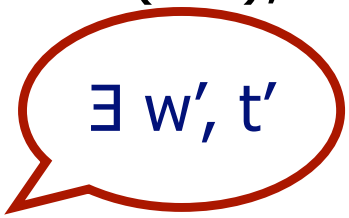
$$Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}$$

$$Q_{s \oplus t, s' \oplus t'} \in X$$

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If $(f, f') \in R_{s,s'}$, $(v, v') \in (R_{s,s'})^{CXT}$, $IH_X(k\text{-}1)$,
   and $s; f\ v \Downarrow^k (t)\ w$ then



$$\mathbf{s'; f'\ v' \Downarrow (t')\ w'}$$

$$(w, w') \in (Q_{s \oplus t, s' \oplus t'})^{CXT}$$

$$Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}$$

$$Q_{s \oplus t, s' \oplus t'} \in X$$

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If $(f, f') \in R_{s,s'}$, $(v, v') \in (R_{s,s'})^{CXT}$, $IH_X(k\text{-}1)$,
   and $s; f\ v \Downarrow^k (t)\ w$ then

$$s'; f'\ v' \Downarrow (t')\ w'$$

$$\mathbf{(w, w') \in (Q_{s \oplus t, s' \oplus t'})^{CXT}}$$

$$\mathbf{Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}}$$

$$\mathbf{Q_{s \oplus t, s' \oplus t'} \in X}$$

∃ Q

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If $(f, f') \in R_{s,s'}$, $(v, v') \in (R_{s,s'})^{CXT}$, **IHx(k-1)**,
   and $s; f\ v \Downarrow^k (t)\ w$ then

$$s'; f'\ v' \Downarrow (t')\ w'$$

$$(w, w') \in (Q_{s \oplus t, s' \oplus t'})^{CXT}$$

$$Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}$$

$$Q_{s \oplus t, s' \oplus t'} \in X$$

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If $(f, f') \in R_{s,s'}$, $(v, v') \in (R_{s,s'})^{CXT}$, **$IH_X(k-1)$**,
   and $s; f\ v \Downarrow^k (t)\ w$ then
   $s'; f'\ v' \Downarrow (t')\ w'$
   $(w, w') \in \mathbf{(Q_{s \oplus t, s' \oplus t'})^{CXT}}$
   $Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}$
   $Q_{s \oplus t, s' \oplus t'} \in X$

Help to deal with inf. quantif. over arguments and H.O. functions

Bisimulations for the nu-calculus

X is a bisimulation iff for all $R_{s,s'} \in X$

4. If $(f, f') \in R_{s,s'}$, $(v, v') \in (R_{s,s'})^{CXT}$, $IH_X(k\text{-}1)$,
   and $s; f\ v \Downarrow^k (t)\ w$ then

$$s'; f'\ v' \Downarrow (t')\ w'$$

$$(w, w') \in (Q_{s \oplus t, s' \oplus t'})^{CXT}$$

$$Q_{s \oplus t, s' \oplus t'} \supseteq R_{s,s'}$$

$$Q_{s \oplus t, s' \oplus t'} \in X$$

and vice versa...

The "hard" equivalence

let x_1=new in
 let x_2=new in
 $\lambda f{:}\nu\to o.$
 (f x_1 = f x_2)

$\equiv$

$\lambda f{:}\nu\to o.$ true

The "hard" equivalence

M

let x_1=new in
let x_2=new in
λf:ν→o.
(f x_1 = f x_2)

≡

M′

λf:ν→o. true

The “hard” equivalence

$([], [], \{(M, M')\}) \in X$

The "hard" equivalence

[]; M

$\Downarrow$

$(\{n_1, n_2\})$

$\lambda f{:}\nu \to o.$

$(f\ n_1 = f\ n_2)$

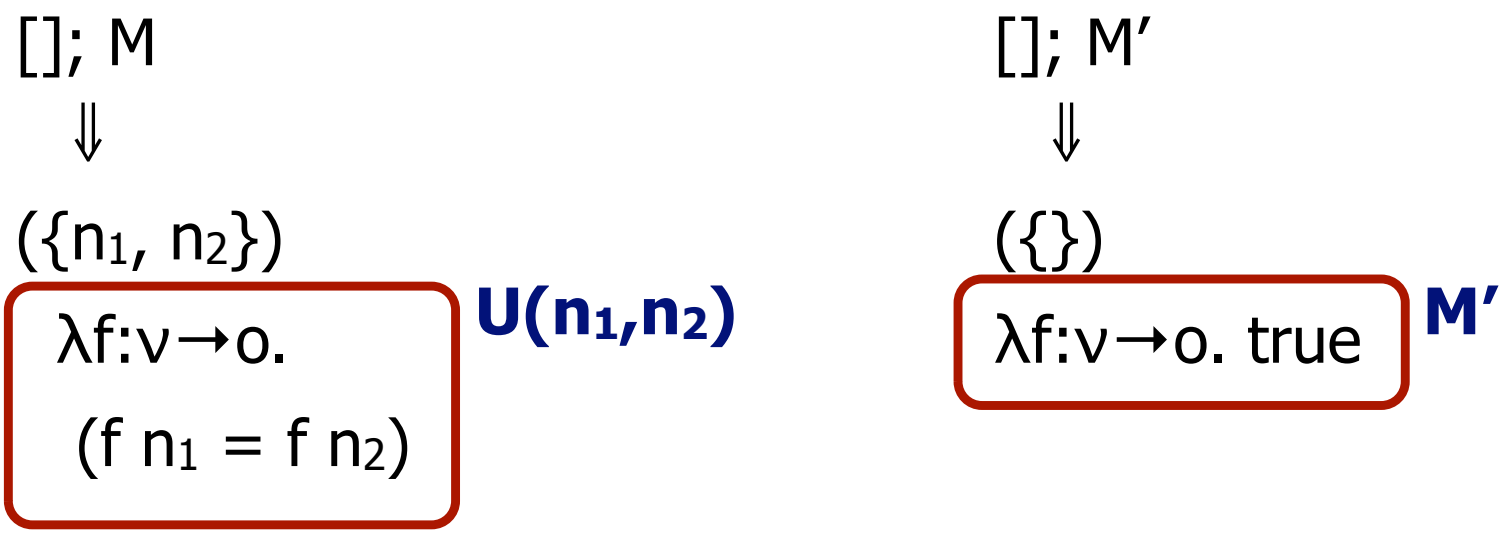
[]; M'

$\Downarrow$

$(\{\})$

$\lambda f{:}\nu \to o.$ true

The "hard" equivalence



The "hard" equivalence

$([], [], \{(M, M')\}) \in X$

$$\frac{(s, s', R) \in X \quad n_1, n_2 \text{ fresh}}{(s \oplus \{n_1, n_2\}, s', R \cup \{(U(n_1, n_2), M')\}) \in X}$$

The "hard" equivalence

$s \oplus \{n_1, n_2\};$

$U(n_1, n_2)\ v$

$\downarrow$

$s \oplus \{n_1, n_2\};$

$v\ n_1 = v\ n_2$

$\Downarrow$

(t) b

$s'; M'\ v'$

$\Downarrow$

({}) true

$(v,v') \in R^{CXT}$

The "hard" equivalence

$s \oplus \{n_1, n_2\};$

$U(n_1, n_2)\ v$

$\downarrow$

$s \oplus \{n_1, n_2\};$

$v\ n_1 = v\ n_2$

$\Downarrow$

(t) b

b=true ?

$s'; M'\ v'$

$\Downarrow$

({}) true

$(v,v') \in R^{CXT}$

The "hard" equivalence

$s \oplus \{n_1, n_2\}; v\ n_1 \quad \equiv \quad s \oplus \{n_1, n_2\}; v\ n_2$

$(v, v') \in R^{CXT}$

The "hard" equivalence

$s \oplus \{n_1, n_2\}; v\ n_1 \quad \equiv \quad s \oplus \{n_1, n_2\}; v\ n_2$

We construct a bisimulation Y s.t.

$(s \oplus \{n_1, n_2\}, s \oplus \{n_1, n_2\}, Q) \in Y$

$(v\ n_1, v\ n_2) \in Q$

$(v,v') \in R^{CXT}$

The "hard" equivalence

Y is easily constructed from X:

$(s, s', R) \in X \quad n_1, n_2$ fresh

$(s \oplus \{n_1, n_2\}, s', R \cup \{(U(n_1, n_2), M')\}) \in X$

$(s, s, Q) \in Y \quad n_1, n_2$ fresh

$(s \oplus \{n_1, n_2\}, s \oplus \{n_1, n_2\},$

$Q \cup \{(U(n_1, n_2), U(n_1, n_2)), \mathbf{(n_1, n_2), (n_2, n_1)}\}) \in Y$

Formalization in Coq

- We used the **locally nameless library** [Aydemir et al.] (3500 loc)
- Relations were straightforwardly encoded in Coq (avoiding use of coinductive types)
- Developed infrastructure about parallel substitution and R^{CXT} (3000 loc)
- Proved soundness of the theory (2800 loc)
- Proof of simple equivalence (1000 loc)
- Proof of hard equivalence (3000 loc)

Conclusions

- A theory of bisimulations for the nu calculus
- Sound and complete
- Deals with H.O. functions
- Simple proofs of equivalence
- Mechanized in Coq

Thank you