Full Abstraction for Reduced ML

Nikos Tzevelekos
Oxford University Computing Laboratory

Joint work with Andrzej Murawski

FoSSaCS, York, March 2009

What this talk is about

Denotational semantics of Reduced ML (Reduced ML).

- Reduced ML (Stark '94) is a functional language with *nominal* integer references.
- Its denotational modelling had not been addressed before in a satisfactory way.
- We provide such a model in *Nominal Games*.

Nominal references

Names in programming languages are *atomic* constructs which can be:

- created fresh locally,
- compared for equality,
- passed around via function application.

…*Locality, Distinguishability, Mobility.*

These are the basic specifications, describing a very 'simple' nominal language, the ν-calculus (Pitts & Stark 93).

Other uses for names: π-calculus, Java, ML, etc.

Nominal references

Names in programming languages are *atomic* constructs which can be:

- created fresh locally,
- compared for equality,
- passed around via function application.

…*Locality, Distinguishability, Mobility.*

These are the basic specifications, describing a very 'simple' nominal language, the ν-calculus (Pitts & Stark 93).

Other uses for names: π-calculus, Java, ML, etc.

Nominal references can also be:

- dereferenced and updated.

Reduced ML: syntax

Simply-typed λ-calculus with integers and references.

Assume a countably infinite set of names, $\mathbb{A}$, with elements $a, b, \ldots$.

- Types:

$$\theta ::= \ \mathsf{unit} \mid \mathsf{int} \mid \mathsf{int\,ref} \mid \theta \rightarrow \theta$$

- Terms:

$$\begin{aligned} M ::= \ & x \mid \lambda x.M \mid M\,M \mid () \mid \Omega \\ & \mid n \mid M \odot M \mid \mathsf{if}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M \\ & \mid a \mid \mathsf{ref}\,M \mid\ !\,M \mid M := M \end{aligned}$$

$$V ::= \ n \mid () \mid a \mid x \mid \lambda x.M$$

- Typing judgments have the shape

$$u, \Gamma \vdash M : \theta$$

where u is a finite subset of $\mathbb{A}$.

$$\frac{}{u, \Gamma \vdash () : \mathsf{unit}} \qquad \frac{}{u, \Gamma \vdash \Omega : \mathsf{unit}} \qquad \frac{i \in \mathbb{Z}}{u, \Gamma \vdash i : \mathsf{int}}$$

$$\frac{(x : \theta) \in \Gamma}{u, \Gamma \vdash x : \theta} \qquad \frac{u, \Gamma \vdash M : \theta \to \theta' \quad u, \Gamma \vdash N : \theta}{u, \Gamma \vdash MN : \theta'} \qquad \frac{u, \Gamma \oplus \{x : \theta\} \vdash M : \theta'}{u, \Gamma \vdash \lambda x^{\theta}.M : \theta \to \theta'}$$

$$\frac{u, \Gamma \vdash M_1 : \mathsf{int} \qquad u, \Gamma \vdash M_2 : \mathsf{int}}{u, \Gamma \vdash M_1 \odot M_2 : \mathsf{int}} \qquad \frac{u, \Gamma \vdash M : \mathsf{int} \qquad u, \Gamma \vdash N_0, N_1 : \theta}{u, \Gamma \vdash \mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0 : \theta}$$

- Typing judgments have the shape

$$u, \Gamma \vdash M : \theta$$

where u is a finite subset of $\mathbb{A}$.

$$\frac{a \in u}{u, \Gamma \vdash a : \mathsf{int\,ref}} \qquad \frac{u, \Gamma \vdash M : \mathsf{int}}{u, \Gamma \vdash \mathsf{ref}\ M : \mathsf{int\,ref}}$$

$$\frac{u, \Gamma \vdash M : \mathsf{int\,ref}}{u, \Gamma \vdash\ !M : \mathsf{int}}$$

$$\frac{u, \Gamma \vdash M : \mathsf{int\,ref} \quad u, \Gamma \vdash N : \mathsf{int}}{u, \Gamma \vdash M := N : \mathsf{unit}}$$

Reduced ML: reduction rules

Reduction rules have the shape

$$s, M \longrightarrow s', M'$$

where $s, s' : \mathbb{A} \to_{\mathrm{fin}} \mathbb{Z}$.

- $s, (\lambda x.M)V \longrightarrow M[V/x]$
- $s, \mathsf{if}\ 0\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0 \longrightarrow s, N_0$
- $s, \mathsf{if}\ 1\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0 \longrightarrow s, N_1$
- $\dots$
- $s, \mathsf{ref}\ i \longrightarrow s(a \mapsto i), a$ with $a \notin \mathsf{dom}(s)$
- $s, a := i \longrightarrow s(a \mapsto i), ()$
- $s, !a \longrightarrow s, i$ with $s(a) = i$.

Notation:

- $M; N \ \equiv\ (\lambda x.N)M$, some x not free in N.
- new (x, M) in $N \ \equiv\ (\lambda x^{\mathsf{int\,ref}}.N)(\mathsf{ref}\ M)$

$$s, \mathsf{new}\,(x, M)\,\mathsf{in}\,N \ \longrightarrow\!\!\!\!\rightarrow\ s', \mathsf{new}\,(x, i)\,\mathsf{in}\,N \ \longrightarrow\!\!\!\!\rightarrow\ s'(a \mapsto i), N[a/x]$$

Notation:

- $M; N \;\equiv\; (\lambda x.N)M$, some x not free in N.
- new (x, M) in $N \;\equiv\; (\lambda x^{\mathsf{int\,ref}}.N)(\mathsf{ref}\ M)$

Examples

- Scope extrusion. E.g: $\lambda x.\,\mathsf{ref}\,x \;:\; \mathsf{int} \rightarrow \mathsf{int\,ref}$.

Notation:

- $M; N \;\equiv\; (\lambda x.N)M$, some x not free in N.
- $\mathsf{new}\,(x, M)\,\mathsf{in}\,N \;\equiv\; (\lambda x^{\mathsf{int\,ref}}.N)(\mathsf{ref}\,M)$

Examples

- Scope extrusion. E.g: $\lambda x.\,\mathsf{ref}\,x \;:\; \mathsf{int} \to \mathsf{int\,ref}$.
- Functions with local state. E.g. define $\mathsf{Counter} : \mathsf{unit} \to \mathsf{int}$ as:

$$\mathsf{new}\,(z, 0)\,\mathsf{in}\,(\lambda x^{\mathsf{unit}}.\,z := \,!\,z + 1\,;\,!\,z)$$

Notation:

- $M; N \;\equiv\; (\lambda x.N)M$, some x not free in N.
- $\mathsf{new}\,(x, M)\,\mathsf{in}\,N \;\equiv\; (\lambda x^{\mathsf{int\,ref}}.N)(\mathsf{ref}\,M)$

Examples

- Scope extrusion. E.g: $\lambda x.\,\mathsf{ref}\,x \;:\; \mathsf{int} \to \mathsf{int\,ref}$.
- Functions with local state. E.g. define $\mathsf{Counter} : \mathsf{unit} \to \mathsf{int}$ as:

$$\mathsf{new}\,(z, 0)\,\mathsf{in}\,(\lambda x^{\mathsf{unit}}.\, z := \,!\,z + 1\,;\,!\,z)$$

- Name equality. Define $\mathsf{Test} : \mathsf{int\,ref} \to \mathsf{int\,ref} \to \mathsf{int}$ as:

$$\begin{aligned}\lambda x^{\mathsf{int\,ref}}.\lambda y^{\mathsf{int\,ref}}.\mathsf{new}\,(z, 0), (x', !\,x), (y', !\,y)\,\mathsf{in}&\\ x := 0\,; y := 1\,;(\mathsf{if}\;!\,x\,\mathsf{then}\,z := 1\,\mathsf{else}\,())\,;&\\ x := \,!\,x'\,; y := \,!\,y'\,;!\,z&\end{aligned}$$

Given $\vdash M : \text{unit}$ we write $M \Downarrow$ if, for some s,

$$\emptyset, M \longrightarrow\!\!\!\!\rightarrow s, ()$$

Definition

Terms-in-context $\emptyset, \Gamma \vdash M_1 : \theta$ and $\emptyset, \Gamma \vdash M_2 : \theta$ are **equivalent** (written $\Gamma \vdash M_1 \cong M_2$) if

$$C[M_1] \Downarrow \quad \Longleftrightarrow \quad C[M_2] \Downarrow$$

for any context $C[-]$ such that $\vdash C[M_1], C[M_2] : \text{unit}$.

Examples

$$\vdash \ \mathsf{new}\,(x,0),(y,0)\,\mathsf{in}\,M \ \cong \ \mathsf{new}\,(y,0),(x,0)\,\mathsf{in}\,M$$

$$\vdash \ \mathsf{new}\,(y,0)\,\mathsf{in}\,(\lambda x^{\mathsf{int\,ref}}.\,\mathsf{Test}(x,y)) \ \cong \ \lambda x^{\mathsf{int\,ref}}.\,0$$

$$f : \mathsf{int\,ref} \to \mathsf{unit} \ \vdash \ \mathsf{new}\,(x,0)\,\mathsf{in}\,(fx\,;x:=0\,;fx)$$
$$\cong \ \mathsf{new}\,(x,0),(y,0)\,\mathsf{in}\,(fx\,;fy)$$

A denotational model for Reduced ML is **fully abstract** if

$$\Gamma \vdash M_1 \cong M_2 : \theta \iff [\![\Gamma \vdash M_1 : \theta]\!] = [\![\Gamma \vdash M_2 : \theta]\!]$$

A denotational model for Reduced ML is **fully abstract** if

$$\Gamma \vdash M_1 \cong M_2 : \theta \iff [\![\Gamma \vdash M_1 : \theta]\!] = [\![\Gamma \vdash M_2 : \theta]\!]$$

This had not been solved satisfactorily.

- Traditional denotational models:
  global/fixed scope (no locality/mobility), soundness only.
- Game semantics (Abramsky & McCusker '97):
  "bad" variables.
- Nominal game semantics (Laird '04, Laird '08):
  references to names, not integers.
- Nominal game semantics (AGMOS '04, Tz. '07):
  FA via semantical quotienting.

Nominal Games

- Two participants: O (environment) and P (program).
- Games specified by plays, i.e *justified* sequences of *moves*. Moves can contain names, i.e. elements of $\mathbb{A}$.
- Moves selected from *arenas*. Examples:

$$[\![\mathsf{unit}]\!] = \{*_P\} \qquad [\![\mathsf{int}]\!] = \{i_P \mid i \in \mathbb{Z}\}$$

$$[\![\mathsf{int\,ref}]\!] = \{a_P \mid a \in \mathbb{A}\}$$

$$[\![\mathsf{int\,ref} \to \mathsf{unit}]\!] = \quad *_P \quad a_O \quad *_P$$

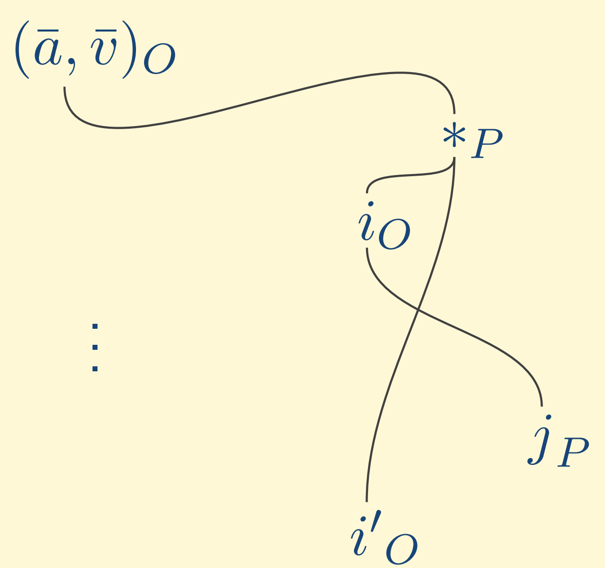
- Founded on *(linear) nominal sets* (Gabbay & Pitts '99).

Strategies

- Programs are interpreted as *strategies* for P.
  - These are *deterministic* up to choice of fresh names,
  - saturated wrt name-permutations.

Example: how does $[\![u, \Gamma \vdash \lambda x.M : \mathsf{int} \to \mathsf{int}]\!]$ look like?

$$[\![u,\Gamma]\!] \longrightarrow [\![\mathsf{int} \to \mathsf{int}]\!]$$

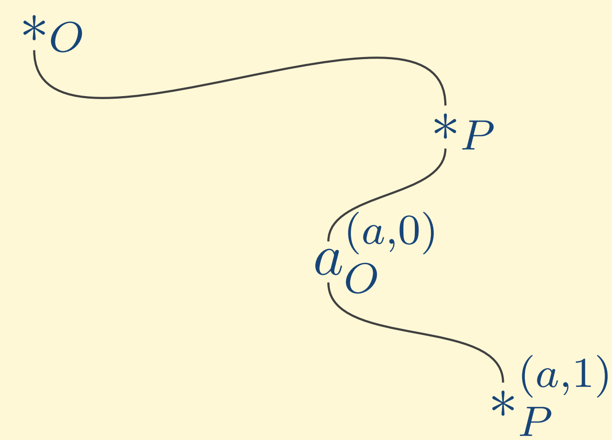


etc.

Local state

- We use moves attached with store:

$$[\![\emptyset,\emptyset]\!] \longrightarrow [\![\mathsf{int\,ref} \to \mathsf{unit}]\!]$$



Notation: $\overset{\frown}{\underset{O}{*}\ \underset{P}{*}}\ \underset{O}{a^{(a,0)}}\ \underset{P}{*^{(a,1)}}$

- Stores should not introduce fresh names (*frugality*).

Considerations:

- Which names are available to P?
- Which names should appear in the store?

Examples

- Consider a function of type int $\rightarrow$ int.

$$u, \Gamma \vdash \lambda x^{\mathsf{int}}.\, M$$

  Can it ‘remember’ x’s from different function calls?

- Now consider a function of type int ref $\rightarrow$ int.

$$u, \Gamma \vdash \lambda x^{\mathsf{int\,ref}}.\, N$$

  Can it ‘remember’ x’s from different function calls?

- Therefore, P cannot have full information of the history of a play.
- In particular, he can only see his own names and those of the current *closure*.
  Put differently, P is *purely functional* namewise.
- But he can see everything else..

Name innocence

- Therefore, P cannot have full information of the history of a play.
- In particular, he can only see his own names and those of the current *closure*.
  Put differently, P is *purely functional* namewise.
- But he can see everything else..

In game semantics:

current closure $\longmapsto$ P-*view*

pure behaviour $\longmapsto$ *Innocence*

General innocence:

- Introduced by (Hyland & Ong '94).
- Captures PCF.

Restrictions on plays

The P-view of a play s (written $\ulcorner s \urcorner$) is the subplay corresponding to the computation that has been performed under (and up to) the current closure.

We define the set of a play's P-**available** names:

$$\mathsf{Av}_P(s) = P(s) \cup \nu(\ulcorner s \urcorner)$$

P-**availability** If P plays an O-name then it must have been P-available.

$$s' p^S \text{ a play and } a \in (\nu(p) \cap O(s')) \implies a \in \mathsf{Av}_P(s')$$

P-**storage** Only P-available names appear in the store.

$$(s' m^S \text{ a play}) \implies \mathsf{dom}(S) = \mathsf{Av}_P(s' m^S)$$

Restrictions on strategies

Conditions on plays:

- P cannot play unavailable names.
- P cannot change/use the values of unavailable names.

But he can still *see* them..

E.g. can a strategy σ_{bad} contain both the following plays?

$$\begin{array}{cccccc} * & * & a^{(a,0)} & *^{(a,1)} & a^{(a,0)} & *^{(a,1)} \\ O & P & O & P & O & P \end{array}$$

$$\begin{array}{cccccc} * & * & a^{(a,0)} & *^{(a,1)} & b^{(b,0)} & *^{(b,0)} \\ O & P & O & P & O & P \end{array}$$

- We can define a notion of **renamings** invisible to P. (Note these are not just name-permutations..)
- This induces an equivalence relation $\overset{\mathrm{r}}{\sim}$ on plays.

Definition

A strategy σ is **blind** if $s \in \sigma$ implies $s' \in \sigma$ for all $s' \overset{\mathrm{r}}{\sim} s$.

Name blindness

- We can define a notion of **renamings** invisible to P. (Note these are not just name-permutations..)
- This induces an equivalence relation $\overset{\mathrm{r}}{\sim}$ on plays.

Definition
A strategy σ is **blind** if $s \in \sigma$ implies $s' \in \sigma$ for all $s' \overset{\mathrm{r}}{\sim} s$.

What about σ_{bad}?

$$* \; * \; a^{(a,0)} \; *^{(a,1)} \; a^{(a,0)} \; *^{(a,1)}$$

$$* \; * \; a^{(a,0)} \; *^{(a,1)} \; b^{(b,0)} \; *^{(b,0)}$$

- We can define a notion of **renamings** invisible to P. (Note these are not just name-permutations..)
- This induces an equivalence relation $\overset{\mathrm{r}}{\sim}$ on plays.

Definition
A strategy σ is **blind** if $s \in \sigma$ implies $s' \in \sigma$ for all $s' \overset{\mathrm{r}}{\sim} s$.

What about σ_{bad}?

$$* \; * \; a^{(a,0)} \; *^{(a,1)} \; a^{(a,0)} \; *^{(a,1)} \quad \overset{\mathrm{r}}{\sim} \quad * \; * \; a^{(a,0)} \; *^{(a,1)} \; b^{(b,0)} \; *^{(b,1)}$$

$$* \; * \; a^{(a,0)} \; *^{(a,1)} \; b^{(b,0)} \; *^{(b,0)}$$

Name blindness

- We can define a notion of **renamings** invisible to P. (Note these are not just name-permutations..)
- This induces an equivalence relation $\stackrel{\mathrm{r}}{\sim}$ on plays.

Definition
A strategy σ is **blind** if $s \in \sigma$ implies $s' \in \sigma$ for all $s' \stackrel{\mathrm{r}}{\sim} s$.

What about σ_{bad}?

$$* \; * \; a^{(a,0)} \; *^{(a,1)} \; b^{(b,0)} \; *^{(b,1)}$$

$$* \; * \; a^{(a,0)} \; *^{(a,1)} \; b^{(b,0)} \; *^{(b,0)}$$

Name blindness

- We can define a notion of **renamings** invisible to P. (Note these are not just name-permutations..)
- This induces an equivalence relation $\stackrel{\mathrm{r}}{\sim}$ on plays.

Definition
A strategy σ is **blind** if $s \in \sigma$ implies $s' \in \sigma$ for all $s' \stackrel{\mathrm{r}}{\sim} s$.

What about σ_{bad}?

breaks determinacy!

$$* \; * \; a^{(a,0)} \; *^{(a,1)} \; b^{(b,0)} \; *^{(b,1)}$$

$$* \; * \; a^{(a,0)} \; *^{(a,1)} \; b^{(b,0)} \; *^{(b,0)}$$

Proposition
There is a category $\mathcal{G}$ having arenas as objects and blind strategies as arrows.

Proposition
$\mathcal{G}$ is a sound model of Reduced ML.

Theorem
Finitary blind strategies are definable.

Theorem
Two Reduced ML terms are equivalent iff they generate the same sets of *complete symmetric* plays.

Conclusion

Notes:

- The fully abstract model of Reduced ML is effectively presentable.
- Nevertheless, Reduced ML is undecidable (Murawski '03).

Conclusion

Notes:

- The fully abstract model of Reduced ML is effectively presentable.
- Nevertheless, Reduced ML is undecidable (Murawski '03).

What has been achieved?

- We have a handle on program equivalence in Reduced ML.
- We have new notions and intuitions in nominal games (name-availability, blindness).

Conclusion

Notes:

- The fully abstract model of Reduced ML is effectively presentable.
- Nevertheless, Reduced ML is undecidable (Murawski '03).

What has been achieved?

- We have a handle on program equivalence in Reduced ML.
- We have new notions and intuitions in nominal games (name-availability, blindness).

THANKS!