Definability and full abstraction in lambda-calculi

Antonio Bucciarelli

Laboratoire Preuves, Programmes et Systèmes
Université Paris Diderot
Outline

1. Introduction
2. The full abstraction problem for PCF
3. Quantitative models
4. The resource calculus
5. Conclusion
Terminology

**Language**

A typed or untyped $\lambda$-calculus endowed with an *operational semantics*, defined via a notion of *reduction* $\rightarrow$, and with a notion of *observational equivalence* $\equiv_{\text{obs}}$. The observational equivalence is *contextual*: two terms $M$ and $N$ are equivalent if for any context $C[ ]$, $C[M]$ and $C[N]$ are observably indistinguishable.

**Examples:**

<table>
<thead>
<tr>
<th>Language</th>
<th>Reduction</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>untyped $\lambda$-calculus</td>
<td>$\beta$-reduction</td>
<td>head normal forms</td>
</tr>
<tr>
<td>PCF</td>
<td>$\beta$-$\delta$-$Y$-reduction</td>
<td>ground constants (integer and booleans)</td>
</tr>
</tbody>
</table>

Hence, in PCF, $M \equiv_{\text{obs}} N$ if for all context $C[ ]$ of ground type, $C[M] \rightarrow c$ iff $C[N] \rightarrow c$, $c$ being a ground constant.

In the untyped $\lambda$-calculus $M \equiv_{\text{obs}} N$ if for all context $C[ ]$, $C[M]$ has a head normal form iff $C[N]$ has a head normal form.
### Terminology

**Model**

A Cartesian closed category, where types of are interpreted by objects, and terms by morphisms. In the untyped case, a model is a reflexive object of the ccc. Convertible terms get the same interpretation: $M \leadsto N \Rightarrow [M] = [M]$.

Examples for PCF:

<table>
<thead>
<tr>
<th>Model</th>
<th>Objects</th>
<th>Morphisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott model</td>
<td>Scott domains</td>
<td>Scott-continuous functions</td>
</tr>
<tr>
<td>Stable model</td>
<td>coherence spaces</td>
<td>stable functions</td>
</tr>
</tbody>
</table>

Examples for the untyped $\lambda$-calculus:

- Graph models, Scott’s $D_\infty$.

Semantic brackets $[\ ]$ (possibly with superscript: $[\ ]^{\text{Scott}}, [\ ]^{\text{stab}}$) denote the interpretation of types and terms. For instance, in the Scott’s model of PCF:

- $[\text{bool}] = (\perp, \text{true}, \text{false}), \perp < \text{true}, \text{false}$
- $[\text{fun} (x : \text{bool}) \rightarrow x] = (\perp, \perp), (\text{true}, \text{true}), (\text{false}, \text{false})$
Full abstraction and definability

$L$ a language, $\mathcal{M}$ one of its models:

**Adequacy**
- $\mathcal{M}$ is *adequate* for $L$ if, for all terms $M, N$, $\llbracket M \rrbracket^\mathcal{M} = \llbracket N \rrbracket^\mathcal{M} \Rightarrow M \equiv_{\text{obs}} N$.
- $\mathcal{M}$ is *fully abstract* for $L$ if, for all terms $M, N$, $\llbracket M \rrbracket^\mathcal{M} = \llbracket N \rrbracket^\mathcal{M} \Leftrightarrow M \equiv_{\text{obs}} N$.

**Definability**
- A morphism $f$ of $\mathcal{M}$ is *$L$-definable* if there is a closed $L$-term $M$ such that $\llbracket M \rrbracket = f$.
- If all the (finite) elements of $\mathcal{M}$ are $L$-definable, then (under some reasonable hypothesis) $\mathcal{M}$ is fully abstract for $L$. 
Historical digression

- The λ-calculus, paradigm of the untyped functional languages, was defined by Alonzo Church around 1930. Its first model was found by D. Scott some 40 years later.

- For PCF, paradigm of typed functional languages, the definition of the canonical Scott model, i.e. of the category of Scott domains and Scott-continuous functions, came some years before the precise definition of the language and of its operational semantics (due to Plotkin, around 1975).
Outline

1 Introduction

2 The full abstraction problem for PCF

3 Quantitative models

4 The resource calculus

5 Conclusion
Plotkin’s terms

```ocaml
let rec omega = fun () -> (omega (): bool);;
(* omega() denotes the undefined boolean value *)

let p = fun (f:bool->bool->bool)->
    if f (omega()) true then
        if f true (omega()) then
            if not(f false false) then true
            else omega()
        else omega()
    else omega();;

let q = fun (f:bool->bool->bool)->
    if f (omega()) true then
        if f true (omega()) then
            if not(f false false) then false
            else omega()
        else omega()
    else omega();;

Is there a context allowing to make a difference between \( p \) and \( q \) ?
The *parallel or* function

\[
\text{por } x \ y = \begin{cases} 
  \text{true} & \text{if } x = \text{true} \text{ or } y = \text{true} \\
  \text{false} & \text{if } x = \text{false} \text{ and } y = \text{false} \\
  \bot & \text{otherwise}
\end{cases}
\]

**Fact**

*por* is a Scott-continuous function.

\[
[p]^{\text{Scott}} \neq [q]^{\text{Scott}} \quad \text{since} \\
[p]^{\text{Scott}} \text{por} = \text{true} \quad \text{and} \\
[q]^{\text{Scott}} \text{por} = \text{false}
\]

**Theorem (Plotkin)**

- The parallel or function is not PCF-definable.
- The terms *p* and *q* (the “parallel or testers”) above are observationally equivalent.
- If PCF is endowed with a new constant computing the parallel or function, then all the finite elements of the Scott model become definable, and the model itself become fully abstract.
A property shared by all PCF-definable functions, not respected by \textit{por}, is \textit{stability}:

A Scott-continuous function $f$ is stable if for all $x, y : x \uparrow y \Rightarrow f(x \land y) = f(x) \land f(y)$

where $x \uparrow y$ means $\exists z \ x, y \leq z$.

### Stable model (Berry-Girard)

- Objects: coherence spaces.
- Morphisms: stable functions.

In this model, $\llbracket p \rrbracket = \llbracket q \rrbracket = \bot (\text{bool} \rightarrow \text{bool} \rightarrow \text{bool}) \rightarrow \text{bool}$

Nevertheless, the theory of the stable model is not closer to the observational equivalence than the one of the Scott model (they are actually incomparable).
A higher-order example

```ocaml
let left_or = fun x y -> if x then true else y;;

let right_or = fun x y -> if y then true else x;;

let or_tester = fun (f: (bool-> bool -> bool) -> bool) -> bool) 
  if f left_or then
    if not(f right_or) then true
    else omega()
  else omega();;
```

In the Scott model, the interpretations of `left_or` and `right_or` are upper bounded by the parallel or. Hence, no functional can yield `true` on the former and `false` on the latter.

As a consequence

\[
\llbracket \text{or}\_\text{tester} \rrbracket^{\text{Scott}} = \llbracket \text{fun}(f: (\text{bool} \rightarrow \text{bool} \rightarrow \text{bool}) \rightarrow \text{bool}) \rightarrow \omega() \rrbracket^{\text{Scott}} = \bot
\]

On the other hand

\[
\llbracket \text{or}\_\text{tester} \rrbracket^{\text{stab}} F = \text{true}
\]

if \( F[\text{left}\_\text{or}]^{\text{stab}} = \text{true} \) and \( F[\text{right}\_\text{or}]^{\text{stab}} = \text{false} \), and such a functional \( F \) does exist in the stable model.

Hence \( \llbracket \text{or}\_\text{tester} \rrbracket^{\text{stab}} \neq \llbracket \text{fun } f \rightarrow \omega() \rrbracket^{\text{stab}} \)
Stability is not enough to characterise the definable functions in a purely functional, sequential language like PCF. Further developments:

- Model of sequential algorithms (Berry-Curien).
- Strongly stable model (B.-Ehrhard).
- Game models (Abramsky-Jagadeesan-Malacaria, Hyland-Ong) (first solutions to the full abstraction problem of PCF).
<table>
<thead>
<tr>
<th>Language</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCF + por</td>
<td>Scott</td>
</tr>
<tr>
<td>PCF stable</td>
<td>stab</td>
</tr>
<tr>
<td>PCF</td>
<td>Games and innocent strategies</td>
</tr>
<tr>
<td>PCF + H</td>
<td>Hypercoherences and strongly stable functions</td>
</tr>
<tr>
<td>PCF + references (Idealised Algol)</td>
<td>Games and well balanced strategies</td>
</tr>
<tr>
<td>PCF + catch (SPCF)</td>
<td>Concrete Data Structures and sequential algorithms</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. The full abstraction problem for PCF
3. Quantitative models
4. The resource calculus
5. Conclusion
The redundant identity

let id = fun (x:bool) -> x;;
let r_id = fun x -> if x then x else x;;

\[ [id] = [r_id] \] in Scott and stable models.
(hence, \textit{a fortiori}, \( id \equiv_{obs} r_id \)).

It is natural to distinguish between these two terms, in order to take into account the usage of resources by a program (intuitively \( r_id \) uses its argument twice, whereas \( id \) uses it once.

This boils down to move from \textit{qualitative} models to \textit{quantitative} ones, like the \textit{relational model}. 
The category $MRel$

- Objects: sets
- Morphisms: $MRel(A, B) = \mathcal{P}(\mathcal{M}_{\text{fin}}(A) \times B)$
  where $\mathcal{M}_{\text{fin}}(A)$ denotes the set of finite multi-sets over $A$, and $\mathcal{P}(A)$ the set of subsets of $A$.
- Identities: $id_A = \{([\alpha], \alpha) | \alpha \in A\}$
- Composition: $f \in MRel(A, B), g \in MRel(B, C)$ $g \circ f =$
  $\{(m_1 \uplus \ldots \uplus m_k, \gamma) | \exists \beta_1, \ldots \beta_k \in B, (m_i, \beta_i) \in f, 1 \leq i \leq k, ([\beta_1, \ldots, \beta_k], \gamma) \in g\}$
- Terminal object: $\emptyset$
- Cartesian product: disjoint union
- Function spaces: $B^A = \mathcal{M}_{\text{fin}}(A) \times B$

**Fact**

$MRel$ is Cartesian closed.
The quantitative flavour of \( MRel \)

Let \([ \_ ]^{\text{rel}}\) denote the interpretation of PCF term in \( MRel \). Then:

\[
\begin{align*}
\llbracket \text{id} \rrbracket^{\text{rel}} &= \{ ([true], true), ([false], false) \} \\
\llbracket r\_id \rrbracket^{\text{rel}} &= \{ ([true, true], true), ([false, false], false), ([true, false], true), ([true, false], false) \}
\end{align*}
\]
A reflexive object in $MRel$

**The model $M_{\infty}$**

- $M_0 = \emptyset$
- $M_{n+1} = (M_{\text{fin}}(D_n))^{<\omega}$
- $M_{\infty} = \bigcup_{n \in \omega} D_n$

In particular $M_1 = \{( [], [], \ldots, [], \ldots )\}$, call $\star$ the unique element of $M_1$.

The isomorphism $M_{\infty} \leftrightarrow M_{\infty}^{M_{\infty}}$ is trivial:

$(m_0, m_1, \ldots, m_k, \ldots) \leftrightarrow (m_0, (m_1, \ldots, m_k, \ldots))$.

The interpretation of a closed $\lambda$-term in $M_{\infty}$ coincides with the set of its non-idempotent intersection types.

**Full abstraction (without definability)**

- $M_{\infty}$ is fully abstract for the untyped $\lambda$-calculus, that is, its theory is the maximal semi-sensible $\lambda$-theory $H\star$.
- Nevertheless, $\star$ is not definable, that is, no closed $\lambda$-term is typable with $\star$. 
Outline

1. Introduction
2. The full abstraction problem for PCF
3. Quantitative models
4. The resource calculus
5. Conclusion
Resource calculi are intended to take into account, from an operational point of view, the linear/non linear use of resources (arguments).

Key idea: **linear substitution**

\[ t \langle t'/x \rangle \] denotes the term \( t \) in which exactly one occurrence of \( x \) is replaced by \( t' \).

Example:

\[ xx \langle \lambda z.z/x \rangle = (\lambda z.z)x + x(\lambda z.z) \]

Linear substitution \( \Rightarrow \) Non determinism.

The **resource (or differential) \( \lambda \)-calculus** (Ehrhard-Regnier) is an extension of both typed and untyped \( \lambda \)-calculi, featuring linear and classical substitutions.
The untyped resource calculus

Syntax

- Terms
  \[ t ::= x \mid \lambda x.t \mid t[b] \]
- Bags
  \[ b ::= [t_1, \ldots, t_k, t^!] \]

Reduction

\[(\lambda x.t)[t_1, \ldots, t_k, t^!] \rightsquigarrow t\langle t_1/x\rangle \ldots \langle t_k/x\rangle\{t/x\}\]

Observational equivalence

A term is in outer normal form, if it has no redexes but under a !; two terms \( t, t' \) are observationally equivalent if for all context \( C[\ ] \), \( C[t] \) reduces to an outer normal form if and only if \( C[t'] \) reduces to an outer normal form.

As for \( \lambda \)-calculus, the interpretation of terms of the resource calculus in \( M_\infty \) may be given via a suitable typing system.
Adequacy

$M_\infty$ is an adequate model of the resource calculus.

Full abstraction

- $M_\infty$ is not fully abstract for the resource calculus (Breuvart, 2013).
- $M_\infty$ is fully abstract for an extension of the resource calculus: the resource calculus with tests. (B., Carraro, Ehrhard, Manzonetto 2011).

Test elimination

A test elimination procedure allows to give an alternative proof of the full abstraction of $M_\infty$ w.r.t. the untyped $\lambda$-calculus, and an original proof of the full abstraction of $M_\infty$ w.r.t. the $!$-free fragment of the resource calculus.
Outline

1 Introduction
2 The full abstraction problem for PCF
3 Quantitative models
4 The resource calculus
5 Conclusion
Some open problems

- Full abstraction for the resource calculus.
- Full abstraction for the non deterministic $\lambda$-calculus.
- Definability and full abstraction for probabilistic PCF.
- Dual problems: given a model, provide an operational characterisation of the theory it induces.
  For instance: provide an operational characterisation of the theory of $M_\infty$ in the resource calculus.
References


• Flavien Breuvart: The resource lambda calculus is short-sighted in its relational model. To appear (TLCA 2013).