GAME SEMANTICS FOR λ-CALCULI

Pietro Di Gianantonio

Università di Udine

Results in **building** and **analyzing** models for the lambda calculus (typed and the untyped) using the paradigm of games semantics.

Results obtained in Udine.

Game semantics

- has solved a long standing problem, to find a **fully-abstract model** for PCF (Abramsky Jagadeesan Malacaria, Hyland Ong, Nickau)

- and many other languages.

How does it work?

- It introduces some **operational behavior** in the **denotational semantics** .

- The execution of a program is described as the interaction of two Players:
  - the **Proponent**: the program itself,
  - the **Opponent**: the environment.

  This interaction is called a **Play**, and consists in a sequence of **Moves** made by the two Players.

What are Moves?

The action of:
- **posing question**
- **giving answer**

The semantics of a program term is defined as a **strategy**, ie a description of how the program **reacts** to every possible **behaviour** of the environments.

Games form a **category**

- **objects**: games (set of rules)
- **morphisms** from A to B : strategies in $A \to B$.

Examples

Addition $\lambda xy\ .\ (x + y)$

- O: what is the **output**?
  - P: what is the **first input**?
  - O: the first input is **3**
  - P: what is the **second input**?
  - O: the second input is **2**
- P: the output is **5**

Example in the lazy λ-calculus

$[\![M = \lambda y.\Omega]\!]$ is the **strategy**:

- O: is M a λ-abstraction?
- P: yes!
- O: is Mx a λ abstraction?
- P: . . . (wait, I'm normalizing a term)

A possible **play** in $[\![M = \lambda y.yM_1]\!]$ is:

- O: is M a λ-abstraction?
- P: yes!
- O: is Mx_1 a λ-abstraction?
  - P: is x_1 a λ-abstraction?
  - O: yes!
  - P: is x_1M_1 a λ-abstraction?

  - O: yes!

- P: yes!

- O: is Mx_1x_2 a λ-abstraction?
  - P: is $x_1M_1X_2$ a λ-abstraction?

Böhm trees

In general, given a **programming language** L and a **fully abstract game model** G for L, there is a strict connection between the

- **Bohm trees** on L
- **strategies** on G

A strategy can be seen as a different description of a Böhm tree.

In a play the environment enquires about the form of the nodes of Böhm tree, and the program reacts accordingly.

This correspondence gives a **universality resul**:
the finite strategies are definable by a finite Böhm tree (or the corresponding terms).

A **key ingredient** in the proofs of full abstraction.

The untyped λ-calculus

A **correspondence** between Böhm trees and strategies exists for arbitrary games model (not just for the fully-abstract ones).

For any M, the strategy $[\![M]\!]$ is a description of the Böhm (Levy-Longo) tree of M.

Any **model** G for the untyped lambda calculus, ie **reflexive** object (in the category of AJM games)

$$[G \to G] \lhd G$$

induces one of three possible theories:

- the theory $\mathcal{H}^*$ induced by D^∞,

- the theory $\mathcal{B}$ induced by Böhm trees,

- the theory $\mathcal{L}$ induced by Lévy-Longo trees.

In particular:

- If $G = [G \to G]$ then $\mathcal{H}^*$

- If $\psi \circ \bot_{[G \Rightarrow G]} = \bot_G$ then $\mathcal{B}$
  where $\psi : [G \to G] \to G$

- $\mathcal{L}$ otherwise

[D Franco Honsell]

Full completion for the (lazy) untyped λ-calculus

Standard games **do not genenerate fully-abstract** models for the pure lazy untyped λ-calculus.

Reasons: not all stategies correspond Levy-Longo trees.

A term replays to the **main questions** of the environment interrogating **only one** argument (the head variable). This behaviour is not forced on strategies.

Fully-abstract models only for untyped calculi **extended** with test constants.

Full abstraction for AJM games

A fully abstract model for the untyped calculus is built by adding a

- an order on moves,
- a motonocity condition on strategies.

[D]

Full definability for innocent games

Innocent games do not need an extra order relations. Insted one can exploiting the justification relations moves, and add a **Persistency** (dual to the Well Bracketing).

This technique, together with new extra condition on strategies gives a fully complete model for the pure lazy λ calculus.

[Ong, D]

Linear realizability

An approach alternative to **Game Semantics** in providing **fully complete**/**fully abstract** models.

The **linear realizability** technique amounts to constructing a category of **Partial Equivalence Relations** (**PERs**) over a **Linear Combinatory Algebra** (**LCA**).

When applied with "**particle style**" linear combinatory algebra yields models "similar" to ones of game semantics

- the combinatory algebra has a simpler and cleaner construction.
- PERs construction yield models with **extensionality** properties.

Linear Combinatory Algebra [Abramsky96]

A **Linear Combinatory Algebra** is an applicative structure $(A, \bullet)$ with a unary (injective) operation **!**, and **combinators B**,**C**,**I**,**K**,**W**,**D**,δ,**F** such that

B xyz = x(yz)	**Cut**
C xyz = (xz)y	**Exchange**
I x = x	**Identity**
K x !y = x	**Weakening**
W x !y = x !y !y	**Contraction**
D !x = x	**Derecliction**
δ !x = !!x	**Comultiplication**
F !x !y = !(xy)	**Closed Functoriality**

LCAs correspond to **Hilbert style** axiomatization of $\multimap$, **!** fragment of **LL**.

A "particle style" LCA [Abramsky97]

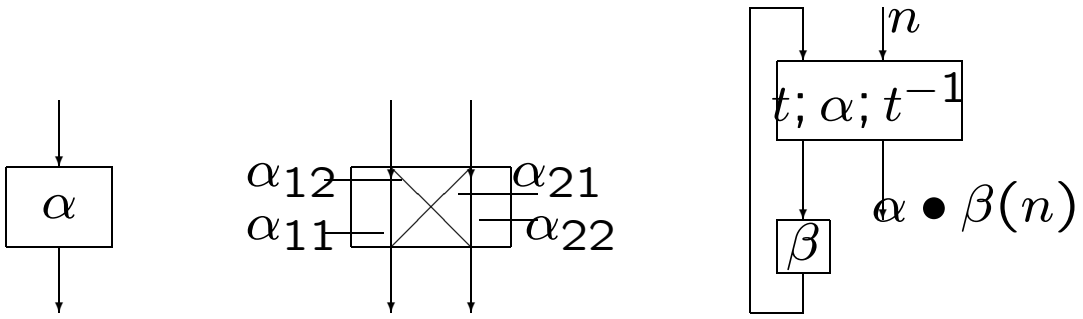
Define $\mathcal{A}_{Pfn} = ([\mathsf{Nat} \rightharpoonup \mathsf{Nat}], \bullet, !)$ as follows:

Fix two injective **codings**: $t : \mathsf{Nat}+\mathsf{Nat} \rightharpoonup \mathsf{Nat}$, $p : \mathsf{Nat}\times\mathsf{Nat} \rightharpoonup \mathsf{Nat}$.

Geometrical description of linear application

In the application $\alpha \bullet \beta$, for $\alpha, \beta \in [\mathrm{Nat} \rightharpoonup \mathrm{Nat}]$, we regard α as a two-input/two-output function via the coding t.

The composition is obtained by an token exchange between α and β



An algebra of reversible computations

$\mathcal{A}_{\mathtt{PInv}} \;=\; ([N \multimap_{Inv} N], \bullet, !)$

$\alpha : N \multimap N$ is an **involution** **iff** graph(f) is **symmetric**.

- $\mathcal{A}_{\mathtt{PInv}}$ is an **highly constraint** algebra, in which all **computations** are **reversible** [Abramsky01].

- Partial involutions can be regarded as generalized **copy-cat strategies**.

- We can think of partial involutions as **abstract families** of **axiom links** as in the proof-nets of LL.

PERs over an **LCA** $\mathcal{A} = (A, \bullet, !)$ [Abramsky-Lenisa 99]

- A **PER** R on $\mathcal{A}$ is a **partial equivalence relation** on A.

PERs over $\mathcal{A}$ form a **linear category**:

- α (**R** $\multimap S$) β **iff** $\forall \gamma\ R\ \gamma'.\ \alpha \bullet \gamma\ S\ \beta \bullet \gamma'$.
- **Pairing** is **tensor** $\otimes$ (up-to transitive closure)

$$(\mathbf{P}\alpha\beta)\ (\mathbf{R} \otimes \mathbf{S})\ (\mathbf{P}\alpha'\beta') \quad \textbf{iff} \quad \alpha R \alpha' \wedge \beta S \beta' ,$$

where $\mathbf{P} = \lambda xyz.zxy$.

If $\mathcal{A}$ is **affine**, then the transitive closure is not needed.

- **Comonad** ! $(!\alpha)\ !\mathbf{R}\ (!\beta)$ **iff** $\alpha\ R\ \beta$.

Fully complete/fully abstract models for:

- The fragment of **System F** consisting of **ML types**;

- The **maximal theory** on the **simply typed λ-calculus**;

- An **infinitary** version of the simply typed λ-calculus;

- A simply typed λ-calculus with k ground constants;

- **Unary PCF**;

- **...**

[Abramsky-Lenisa]

A general pattern for full completeness proofs

A **Decomposition Theorem**:
any **semantical element decomposes** in such a way to allow us to recover the **top-level structure** of the **typed Böhm tree** corresponding to it. The proof of the Decomposition Theorem is **factorized** in various lemmata which express properties of the **underlying linear category**.

Infinitary Böhm trees do not arise.

An example of decomposition theorem

Forall $f \in \mathsf{Hom}_{\mathbf{G}(\vec{\mathcal{U}})}(\overset{n}{\underset{i=1}{\times}} [\![\vec{X} \vdash \sigma_i]\!], [\![\vec{X} \vdash X_k]\!])$,
where $\sigma_i = \tau_{i1} \to \ldots \to \tau_{iq_i} \to X_i$
there exists $i \in \{1, \ldots, n\}$ and $g_1 \ldots g_{q_i}$, $\quad g_j \in \mathsf{Hom}_{\mathbf{G}(\vec{\mathcal{U}})}([\![\vec{X} \vdash \vec{\sigma}]\!], [\![\vec{X} \vdash \tau_{ij}]\!]$

such that

$$\mathbf{f} = [\![\vec{X};\ \vec{x} : \vec{\sigma} \vdash x_i : \sigma_i]\!] \bullet g_1 \ldots \bullet g_{q_i}\ .$$

I.e.:

$$\mathbf{f} = [\![\vec{X};\ \vec{x} : \vec{\sigma} \vdash x_i : \vec{\tau_i} \to X_i]\!]$$

$$g_1 \quad \ldots \quad g_{q_i}$$

What are fully abstract/ fully complete models useful for?

- The **infinitary** λ-calculus.

Exploiting the **algorithmic** content of Games:

- Applications to **model checking** [McCusker-Ghica00].

- **Semantical proofs** of **decidability**:

We exploit the **extra dimension** offered by **interactions**, **strategies**, **involutions**, and the fact that **atomic** values are **structured**.