Game semantics for first-order logic

Olivier LAURENT
Preuves Programmes Systèmes
CNRS – Université Paris VII
UMR 7126 – Case 7014
75205 Paris Cedex 13 – FRANCE
Olivier.Laurent@ens-lyon.fr

October 20, 2010

Abstract

We refine HO/N game semantics with an additional notion of pointer (mu-pointers) and extend it to first-order classical logic with completeness results. We use a Church style extension of Parigot's lambda-mu-calculus to represent proofs of first-order classical logic.

We present some relations with Krivine's classical realizability and applications to type isomorphisms.

Keywords: game semantics; first-order logic; classical logic; intuitionistic logic; lambda-calculus; lambda-mu-calculus; full completeness; type isomorphisms; realizability; control category.

ACM: F.3.2 Semantics of Programming Languages — Denotational semantics; F.4.1 Mathematical Logic — Lambda calculus and related systems; F.3.3 Studies of Program Constructs — Control primitives, Functional constructs.

Game interpretations of logic and programming languages have been initially developed on the logic side (for example [Lor60] for intuitionistic logic). From the beginning of the 90s, most of the attention has been turned to programming languages with the introduction of *game semantics* [AJM00, HO00, Nic94, McC96, Har99].

Our goal is to develop a game model of first-order classical logic based on the HO/N model (more precisely its "classical" version presented in [Lai97] which relaxes the bracketing condition). We have to work between the model of the λ-calculus presented in [DHR96] (which is fully complete for the λ-calculus and thus not general enough to allow for the interpretation of classical features) and the model of [Lai97] where the use of answers might allow for too general behaviours (from the logical point of view). The difference between these two models can be figured out by looking at the interpretation of atoms: [DHR96] uses a *one-move* game, while [Lai97] uses a *two-moves* game. The key ingredient will be the introduction of additional *μ-pointers* in the model of [DHR96] (together with the usual justification pointers, or λ-pointers). The extension of the one-move model allows for the interpretation of classical logic (and not only the λ-calculus).

Game models provide accurate interpretations of logical systems and programming languages as given by *full completeness* results (any element of the model acting on the interpretation of

a formula/type is the interpretation of a proof/term). A natural companion property is *faithful completeness* (two different syntactic objects have different interpretations). When both are satisfied, the distinction between syntax and semantics becomes almost irrelevant as suggested by Girard [Gir99].

To be slightly more precise, we say that a denotational model is *equivalence complete* if it is fully and faithfully complete, and if each object of the model is isomorphic to the interpretation of a formula/type. That is, if the interpretation functor into the model defines an *equivalence of categories* between the syntactic category and the model.

We will show how the appropriate notion of canonical form for a Church style first-order extension of Parigot's $\lambda\mu$-calculus [Par92] together with our first-order game model give an equivalence completeness result. This is the main theorem of this paper.

Starting from the first-order case, we derive a few (known and new) models for different subsystems. We also study the relation between our one-move model with μ-pointers and the two-moves model (in the setting of propositional logic). Here is a summary of the main game models considered in the paper:

atomic games	simply typed/propositional λ-calculus (1 atom)	propositional NJ/λ-calculus	propositional $\mathsf{NK}/\lambda\mu$-calculus	first-order logic
1 move	[DHR96]	Section 3.2	Section 3.3	Section 2
2 moves	[HO00, Nic94]	Section 3.3	[Lai97]	end of Section 3.3
structures used here	λ-pointers	atomic labels	μ-pointers	first-order labels instantiations

There are equivalence completeness results underlying all these models.

Following the method developed in [Lau05a], we apply our game model to the purely syntactic problem of characterizing the type isomorphisms of call-by-name first-order classical logic. This is a new result in the topic of type isomorphisms.

We end the paper with the presentation of a close relation between game semantics and Krivine's classical realizability [Kri06]. This is important for two reasons. First, the idea of introducing μ-pointers in the one-move game models came from an analysis of the interpretation of proofs through Krivine's realizability. Second, game semantics and realizability are two of the most important tools developed along the Curry-Howard correspondence to relate logic and computer science and to derive computational interpretations of proofs. Being able to conciliate these two approaches is a very pleasant thing.

In this paper we focus on the logical aspects of game semantics. However games are also a crucial tool in the study of the semantics of programming languages. The $\lambda\mu$-calculus appears as a natural bridge since it is known to provide both a term syntax for proofs in classical logic and a foundation for functional programming languages with control operators. The link between games and realizability, which is presented here, offers another bridge between games and the theory of programming languages.

Related works. The game setting developed in [Lor60, Fel85] is quite similar to our proposal concerning the notion of play (and view). However it is done in an intuitionistic setting and without any particular interest for the composition of strategies which is at the core of HO/N games. In this line of work, Coquand [Coq95] has explicitly worked on composition and in a classical setting

but in relation with a quite different syntactic system: Novikoff's calculus. Finally Herbelin [Her97] (following Coquand) and Laird [Lai97] (following HO/N) arrived to a meeting point by giving a fully complete game model for a classical extension (*à la λμ-calculus*) of PCF, that is without propositional variables or quantification.

A key ingredient which is new with respect to those works is the notion of μ-pointer. It happens that they appear to be a particular case of the contingency pointers introduced by Laird for local exceptions [Lai01].

A more algebraic approach (by means of generators and relations) to game semantics for first-order quantification is developed in [Mim09]. The underlying logic is very basic: linear and without propositional connectives.

Contents

1 Notations and used languages

1.1 First-order logic

In the whole paper, we consider a fixed first-order language $\mathcal{L}$, that is a countable set of function symbols (with given arities), denoted by f, g, ... and a countable set of relation symbols (with given arities), denoted by X, Y, ... (arities are natural numbers). We assume given a countable set of first-order variables $\mathcal{V}$, denoted x, y, ...

To clarify the different uses we will have of first-order variables and of first-order terms, we consider the set of variables $\mathcal{V}$ as the disjoint union of three countable sets of variables: $\mathcal{A}$-variables, $\mathcal{O}$-variables and $\mathcal{P}$-variables. And we assume given an enumeration $(o_i)_{i\in\mathbb{N}}$ of $\mathcal{O}$-variables.

First-order terms are defined as:

$$t \quad ::= \quad x \mid f\vec{t}$$

where function application respects the arity of symbols.

As sub-classes, we will use $\mathcal{AP}$-terms (first-order terms built from $\mathcal{A}$-variables and $\mathcal{P}$-variables only) and $\mathcal{OP}$-terms (first-order terms built from $\mathcal{O}$-variables and $\mathcal{P}$-variables only). This will be done in the spirit of Barendregt's convention [Bar84, 2.1.13 page 26]: different names are used for different purposes (see in particular Section 2). $\mathcal{A}$-variables will be used for bound occurrences in types/formulas and arenas, $\mathcal{O}$-variables for bound occurrences in $\lambda\mu$-terms and strategies and $\mathcal{P}$-variables for free occurrences.

Formulas are defined as:

$$A \quad ::= \quad \top \mid \bot \mid X\vec{t} \mid A \to A \mid A \wedge A \mid \forall x A$$

where relation application respects the arity of symbols, x is an $\mathcal{A}$-variable and $\vec{t}$ are $\mathcal{AP}$-terms.

An *atomic formula* is $Xt_1 \dots t_k$, $\top$ or $\bot$, denoted R, S, ... If it is neither $\top$ nor $\bot$ it is a *non-constant* atomic formula.

The (now quite popular [Kri01, Sel01, Lau04]) restriction of the set of connectives to the so-called "negative" ones is what makes the framework much easier to manage. Note that the other connectives are easy to define from their negative dual by means of negation (for example $\exists x A \equiv (\forall x(A \to \bot)) \to \bot$).

1.2 Church style $\lambda\mu$-calculus for first-order logic

In order to describe proofs in first-order classical logic, we use (according to the Curry-Howard correspondence) a Church style extension of Parigot's $\lambda\mu$-calculus [Par92] with abstraction and application for first-order universal quantification. First-order formulas are used as types.

Given two disjoint countable sets of variables (λ-variables, denoted a, b, ... and μ-variables, denoted α, β, ...), the corresponding $\lambda\mu$-terms are:

$$M \quad ::= \quad a \mid \lambda a.M \mid (M)M \mid \langle M, M\rangle \mid \pi_1 M \mid \pi_2 M \mid \star \mid [\alpha]M \mid \mu\alpha.M \mid \Lambda x.M \mid M\{t\}$$

where x is an $\mathcal{O}$-variable and t is an $\mathcal{OP}$-term. We use the simplified notation $\mu\alpha[\beta]M$ instead of $\mu\alpha.[\beta]M$ when these two constructions come together.

$\lambda\mu$-terms are considered up to α-equivalence for λ-variables bound by λ, μ-variables bound by μ and $\mathcal{O}$-variables bound by Λ. We consider only $\lambda\mu$-terms without free $\mathcal{O}$-variables (Barendregt's

$$\frac{}{\Gamma, a : A \vdash a : A \mid \Delta} \qquad \frac{\Gamma, a : A \vdash M : B \mid \Delta}{\Gamma \vdash \lambda a.M : A \to B \mid \Delta} \qquad \frac{\Gamma \vdash M : A \to B \mid \Delta \qquad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash (M)N : B \mid \Delta}$$

$$\frac{}{\Gamma \vdash \star : \top \mid \Delta} \qquad \frac{\Gamma \vdash M : A \mid \Delta \qquad \Gamma \vdash N : B \mid \Delta}{\Gamma \vdash \langle M, N \rangle : A \wedge B \mid \Delta} \qquad \frac{\Gamma \vdash M : A_1 \wedge A_2 \mid \Delta}{\Gamma \vdash \pi_i M : A_i \mid \Delta}$$

$$\frac{\Gamma \vdash M : A \mid \Delta, \alpha : A}{\Gamma \vdash [\alpha]M : \bot \mid \Delta, \alpha : A} \qquad \frac{\Gamma \vdash M : \bot \mid \Delta, \alpha : A}{\Gamma \vdash \mu\alpha.M : A \mid \Delta}$$

$$\frac{\Gamma \vdash M[{}^{y}/_{z}] : A[{}^{y}/_{x}] \mid \Delta}{\Gamma \vdash \Lambda z.M : \forall x A \mid \Delta}\ y \notin \Gamma, M, A, \Delta \qquad \frac{\Gamma \vdash M : \forall x A \mid \Delta}{\Gamma \vdash M\{t\} : A[{}^{t}/_{x}] \mid \Delta}$$

x is an $\mathcal{A}$-variable, y is a $\mathcal{P}$-variable and z is an $\mathcal{O}$-variable.

Table 1: Typing rules for the first-order $\lambda\mu$-calculus

convention). A $\lambda\mu$-term is *closed* if it contains neither free λ-variables nor free μ-variables (it may contain free $\mathcal{P}$-variables).

Typing judgments are of the shape $\Gamma \vdash M : A \mid \Delta$ where Γ is a set of typing declarations for distinct λ-variables (*i.e.* pairs $a : A$) and Δ is a set of typing declarations for distinct μ-variables (*i.e.* pairs $\alpha : A$). The derivation rules for this system are given in Table 1.

Through the Curry-Howard correspondence, type inhabitance corresponds to provability.

Proposition 1 (Provability)
The formula A is provable in first-order classical logic if and only if there exists a closed $\lambda\mu$-term M such that $\vdash M : A \mid$ is derivable.

The equality between proofs is the congruence generated by the equational theory $\beta\eta\mu\rho\theta$ on *typed* $\lambda\mu$-terms given in Table 2.

1.3 The syntactic category

The *syntactic category* $\mathcal{S}$ has objects given by types and morphisms from A to B obtained by quotienting the set of closed $\lambda\mu$-terms of type $A \to B$ by the congruence generated by $\beta\eta\mu\rho\theta$. The identity morphism is the equivalence class of the $\lambda\mu$-term $\lambda a.a$ of type $A \to A$. The composition of two equivalence classes containing $M : A \to B$ and $N : B \to C$ is the class of $\lambda a.(M)(N)a : A \to C$ ($a \notin M$, $a \notin N$).

In order to simplify our work in the rest of the paper, we are going to move from the syntactic category to an equivalent one.

Concerning formulas, we first define $\to$-canonical forms (non-terminal $\mathtt{Q}$ in Table 4):

$$\forall \vec{x}(Q_1 \to \cdots \to Q_k \to R)$$

with R atomic but different from $\top$ (called the *final atom* of the formula) and the Q_js in $\to$-canonical form. Then *canonical forms* are:

$$\bigwedge_{1 \leq i \leq n} \forall \vec{x}(Q_1^i \to \cdots \to Q_{k_i}^i \to R^i)$$

$(\lambda a.M)N$	$=_\beta$	$M[^N/_a]$	$: A$	
$\lambda a.(M)a$	$=_\eta$	M	$: A \to B$	$a \notin M$
$\pi_1\langle M, N\rangle$	$=_\beta$	M	$: A$	
$\pi_2\langle M, N\rangle$	$=_\beta$	N	$: A$	
$\langle \pi_1 M, \pi_2 M\rangle$	$=_\eta$	M	$: A \wedge B$	
$\star$	$=_\eta$	M	$: \top$	
$(\Lambda x.M)\{t\}$	$=_\beta$	$M[^t/_x]$	$: A$	
$\Lambda x.M\{x\}$	$=_\eta$	M	$: \forall x A$	$x \notin M$
$(\mu\alpha.M)N$	$=_\mu$	$\mu\alpha.M[^{[\alpha](L)N}/_{[\alpha]L}]$	$: A$	
$\pi_1\mu\alpha.M$	$=_\mu$	$\mu\alpha.M[^{[\alpha]\pi_1 L}/_{[\alpha]L}]$	$: A$	
$\pi_2\mu\alpha.M$	$=_\mu$	$\mu\alpha.M[^{[\alpha]\pi_2 L}/_{[\alpha]L}]$	$: A$	
$(\mu\alpha.M)\{t\}$	$=_\mu$	$\mu\alpha.M[^{[\alpha]L\{t\}}/_{[\alpha]L}]$	$: A$	
$[\beta]\mu\alpha.M$	$=_\rho$	$M[^\beta/_\alpha]$	$: \bot$	
$\mu\alpha[\alpha]M$	$=_\theta$	M	$: A$	$\alpha \notin M$
$[\alpha]M$	$=_\rho$	M	$: \bot$	

where $M[^{\mathcal{C}[L]}/_{[\alpha]L}]$ is obtained by substituting any sub-term of M of the shape $[\alpha]L$ by $\mathcal{C}[L]$.

Table 2: Equalities between $\lambda\mu$-terms

with $n \geq 0$ (where $\bigwedge_{1\leq i\leq 0} Q_i = \top$, $\bigwedge_{1\leq i\leq 1} Q_i = Q_1$ and $\bigwedge_{1\leq i\leq n+1} Q_i = (\bigwedge_{1\leq i\leq n} Q_i) \wedge Q_{n+1}$), with the $\forall\vec{x}(Q_1^i \to \cdots \to Q_{k_i}^i \to R^i)$s in $\to$-canonical form. This corresponds to the non-terminal C in the grammar of Table 4.

Proposition 2 (Canonical forms for formulas)
If we consider formulas up to the equations of Table 3[1] *(except the last two), any formula can be written in canonical form.*

Proof: We consider the equations of Table 3 (except the last two) as rewriting rules from left to right.

We define the two functions ϕ and ψ from formulas to integers greater or equal to 2:

$$\begin{aligned}
\phi(\top) = \phi(\bot) = \phi(R) = \psi(\top) = \psi(\bot) = \psi(R) &= 2\\
\phi(A \wedge B) &= 2(\phi(A) + 1)\phi(B)\\
\psi(A \wedge B) &= 2(\psi(A) + 1)\psi(B)\\
\phi(A \to B) &= \phi(B)^{\phi(A)}\\
\psi(A \to B) &= \psi(B)^{\psi(A)}\\
\phi(\forall x A) &= \phi(A)^2\\
\psi(\forall x A) &= 2\psi(A)
\end{aligned}$$

We can easily check that for each rewriting rule $A \mapsto B$, $(\phi(A), \psi(A)) > (\phi(B), \psi(B))$ (with respect to the lexicographic order). Finally, if A is a formula such that no rewriting rule applies to it, then A is in canonical form. □

[1] These equations are validated by syntactic isomorphisms, see Proposition 11 page 46.

$$
\begin{aligned}
A \wedge (B \wedge C) &= (A \wedge B) \wedge C \\
A \wedge \top &= A \\
\top \wedge A &= A \\
\\
(A \wedge B) \to C &= A \to (B \to C) \\
\top \to A &= A \\
\\
A \to (B \wedge C) &= (A \to B) \wedge (A \to C) \\
A \to \top &= \top \\
\\
\forall x(A \wedge B) &= \forall x A \wedge \forall x B \\
\forall x \top &= \top \\
A \to \forall x B &= \forall x(A \to B) && x \notin A \\
\\
A \wedge B &= B \wedge A \\
\forall x \forall y A &= \forall y \forall x A
\end{aligned}
$$

Table 3: Type isomorphisms

$$
\begin{aligned}
\mathsf{R} &::= X\vec{t} \mid \bot \\
\mathsf{A} &::= \mathsf{R} \mid \mathsf{Q} \to \mathsf{A} \\
\mathsf{Q} &::= \mathsf{A} \mid \forall x \mathsf{Q} \\
\mathsf{B} &::= \mathsf{Q} \mid \mathsf{B} \wedge \mathsf{Q} \\
\mathsf{C} &::= \mathsf{B} \mid \top
\end{aligned}
$$

Table 4: Canonical forms for formulas

Up to the $\beta\eta\mu\rho\theta$ equational theory, any closed $\lambda\mu$-term whose type is a canonical form can be written as a *canonical normal form* which is either $\star$ or a tuple of terms of the shape:

$$\Lambda\vec{x}.\lambda\vec{a}.\underline{\mu[\,]}((b)\{\vec{t}\})\vec{M}$$

where $\underline{\mu[\,]}$ is of the shape $\mu\alpha[\beta]$ except that $[\beta]$ disappears if $((b)\{\vec{t}\})\vec{M}$ has type $\perp$ and that $\mu\alpha$ disappears if $\underline{\mu[\,]}((b)\{\vec{t}\})\vec{M}$ has type $\perp$ (see Appendix A.1 for a proof of this result).

The *syntactic category* $\mathcal{S}^c$ is the category in which objects are types in canonical form and morphisms are closed $\lambda\mu$-terms in canonical normal form quotiented by $\beta\eta\mu\rho\theta$[2]. According to the previous remarks, this is a category equivalent to $\mathcal{S}$.

2 A game model of first-order logic

2.1 Arenas

The notions of *forest* and *tree* will occur at different places in this work. Sometimes enriched with some additional structure (such as labels or pointers) and sometimes not. Here we consider forests and trees as finite objects defined by mutual induction:

- a finite list of trees is a forest,
- a node together with a forest is a tree, the node is called a *root* and the roots of the trees of the forest are the *sons* of this root and the trees of the forest are the *immediate sons* of this tree.

This definition is well founded by using the case of an empty list of trees as a forest. Notice that a tree is never empty while a forest could perfectly be empty.

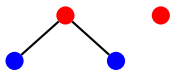
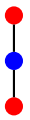
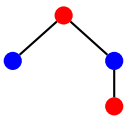
The root of a tree is considered as the *top element* of the tree, so that we can speak about a node *above* or *below* another in a tree/forest. The *polarity* of a node is the parity of the length of the path from a root to this node (in particular the polarity of roots is even).

If $\mathcal{F}$ is a forest and $\mathcal{T}$ is a tree, the *graft* of $\mathcal{F}$ on $\mathcal{T}$ is the tree which has the same root as $\mathcal{T}$ and with immediate sons obtained by concatenating $\mathcal{F}$ (on the left) to the list of the immediate sons of $\mathcal{T}$. If $\mathcal{F}'$ is a forest, the graft of $\mathcal{F}$ on $\mathcal{F}'$ is obtained by grafting $\mathcal{F}$ on each tree of $\mathcal{F}'$ (this may entail duplications of $\mathcal{F}$, and if $\mathcal{F}$ comes with some additional structure, this structure is also duplicated).

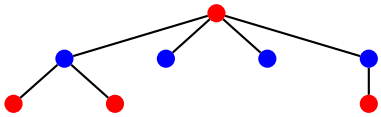
If $\mathcal{T}_1$ and $\mathcal{T}_2$ are two trees, the *merging* of $\mathcal{T}_1$ and $\mathcal{T}_2$ is the tree obtained by grafting the list of immediate sons of $\mathcal{T}_1$ (which is a forest) on $\mathcal{T}_2$. This means that the two roots are identified and the two lists of immediate sons are concatenated. If $\mathcal{F}_1 = [\mathcal{T}_1, \ldots, \mathcal{T}_p]$ and $\mathcal{F}_2 = [\mathcal{T}'_1, \ldots, \mathcal{T}'_q]$ are two forests, the *merging* of $\mathcal{F}_1$ and $\mathcal{F}_2$ is the forest $[\mathcal{T}_1^1, \ldots, \mathcal{T}_1^q, \mathcal{T}_2^1, \ldots, \mathcal{T}_2^q, \ldots, \mathcal{T}_p^1, \ldots, \mathcal{T}_p^q]$ where the tree $\mathcal{T}_i^j$ is the merging of $\mathcal{T}_i$ and $\mathcal{T}'_j$.

Example 1
If we consider the forest $\mathcal{F}$ and the two trees $\mathcal{T}_1$ and $\mathcal{T}_2$:

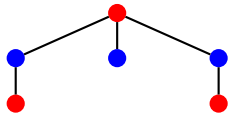
[2] We will see in fact in Corollary 2.1 that there is no quotient involved here since two different canonical normal forms cannot be equalized through $\beta\eta\mu\rho\theta$.

$\mathcal{F} =$  $\mathcal{T}_1 =$  $\mathcal{T}_2 =$ 

The graft of $\mathcal{F}$ on $\mathcal{T}_2$ is the following tree:



The merging of $\mathcal{T}_1$ and $\mathcal{T}_2$ is:



We define a notion of arena adapted to the presence of first-order quantification (in the spirit of polymorphic arenas [Hug97] developed for second-order quantification).

Definition 1 (Arena)
An *arena* is a forest with nodes labelled with:

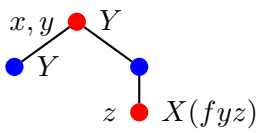
- a list of first-order $\mathcal{A}$-variables, called the *first-order label* of the node;
- a list of non-constant atomic formulas (using only $\mathcal{AP}$-terms), called the *atomic label* of the node (in such a way that $\mathcal{A}$-variables appearing in an $\mathcal{AP}$-term already appear in the first-order label of the node or of a node above it).

The nodes of the forest are called *moves*. Concerning the *polarity*, we also use O for even and P for odd.

If the move $\mathtt{m}$ is the son of the move $\mathtt{n}$ in the arena A, we say that $\mathtt{n}$ *enables* $\mathtt{m}$ (denoted by $\mathtt{n} \vdash_A \mathtt{m}$). Roots are also called *initial moves* denoted by $\vdash_A \mathtt{m}$.

In this paper, we have to deal with a bunch of binding structures. For each of them we can use binding through names and α-renaming, de Bruijn indexes, pointers, ... We decide to use explicit names for first-order variables in arenas. If an $\mathcal{A}$-variable x appears in an $\mathcal{AP}$-term of the atomic label of a move $\mathtt{m}$ and also in the first-order label of a move $\mathtt{n}$ above $\mathtt{m}$ (or of $\mathtt{m}$ itself), x has to be considered as bound in the arena. We will not explicitly deal with arenas up to α-conversion of these bound $\mathcal{A}$-variables. However we will assume all the elements of the first-order labels of an arena to be different. This could require implicit renaming in the arena constructions.

Example 2
If we represent first-order labels on the left-hand side and atomic labels on the right-hand side of each move, here is an arena:



where X has arity 1, Y has arity 0, and f has arity 2.

Remember that, in the general case, the atomic label of a move may contain more than one element.

For the following examples, we name the root as $\mathtt{a}_0$, its sons as $\mathtt{a}_1$ and $\mathtt{a}_2$ and the son of $\mathtt{a}_2$ is named $\mathtt{a}_3$.

Definition 2 (Arrow arena)
Let A and B be two arenas, the *arrow arena* $A \to B$ is the graft of A on B.

2.2 Sequences of moves

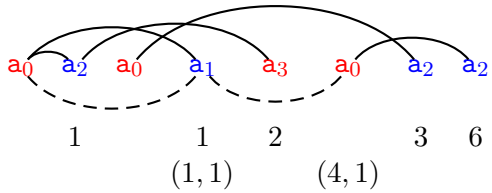
Game semantics usually deals with sequences of moves equipped with some additional structure. Due to the presence of first-order quantification, we will use some even richer structure.

We first introduce pointers on sequences of moves. Two kinds of pointers are required in our setting.

Definition 3 (Justified sequence)
A *justified sequence* $\mathtt{s}$ on the arena A is a sequence of moves of A together with:

- for each occurrence of a non-initial move $\mathtt{m}$, we give a *justification pointer* (or λ-*pointer*) to an earlier occurrence of move in $\mathtt{s}$ (that corresponds to giving an integer smaller than the index of $\mathtt{m}$ in $\mathtt{s}$) which enables $\mathtt{m}$ in A;
- for each occurrence of a move $\mathtt{m}$ with atomic label l in A, we give, for each element of l, at most one μ-*pointer* to an element of the atomic label of an earlier occurrence of move $\mathtt{n}$ of opposite polarity (this can be represented as, for each element of l, an integer corresponding to the index of $\mathtt{n}$ and then an integer giving the chosen element in the atomic label l' of $\mathtt{n}$).

Example 3
Here is a justified sequence on the arena A of Example 2:



λ-pointers are represented as plain lines and μ-pointers as dashed lines.

The second line gives λ-pointers as integers. The third line gives μ-pointers as pairs of integers (the second index is always 1 since the length of the atomic labels of A is at most 1, and there is at most one pair for each move for the same reason).

We now introduce first-order instantiations on sequences of moves (independently of the pointer structure). An $\mathcal{O}$-*instantiation* of a move $\mathtt{m}$ of an arena A, which has a first-order label of length n, is a list of n $\mathcal{O}$-variables. A $\mathcal{P}$-*instantiation* (which is *not* dual to $\mathcal{O}$-instantiation) of a move $\mathtt{m}$ of an arena A, which has a first-order label of length n, is a list of n $\mathcal{OP}$-terms.

Definition 4 (Instantiated sequence)
An *instantiated sequence* $\mathtt{s}$ on the arena A is a sequence of moves of A together with:

- an $\mathcal{O}$-instantiation for each O-move
- a $\mathcal{P}$-instantiation for each P-move

such that all the $\mathcal{O}$-variables appearing in the $\mathcal{O}$-instantiations are different.

We consider the possibility of modifying the $\mathcal{O}$-variables: an *$\mathcal{O}$-renaming* is an injection from the set of $\mathcal{O}$-variables to itself. If $\mathtt{s}$ is an instantiated sequence and if ς is an $\mathcal{O}$-renaming, $\mathtt{s}\varsigma$ is the instantiated sequence obtained by substituting o by $\varsigma(o)$ in any instantiation of $\mathtt{s}$.

More generally, an *$\mathcal{O}$-substitution* is a function from $\mathcal{O}$-variables to $\mathcal{OP}$-terms. If $\mathtt{s}$ is an instantiated sequence and if ϑ is an $\mathcal{O}$-substitution, $\mathtt{s}\vartheta$ is obtained by substituting o by $\vartheta(o)$ in any $\mathcal{P}$-instantiation of $\mathtt{s}$.

The combination of pointers and instantiations is required for interaction sequences and plays.

Definition 5 (Interaction sequence)
Let A, B and C be three arenas, an *interaction sequence* $\mathtt{u}$ on A, B and C is a justified sequence on $(A \to B) \to C$ (without any μ-pointer between a move of A and a move of C) together with:

- for each O-move played in C and for each P-move played in A, an $\mathcal{O}$-instantiation;
- for each P-move played in C and for each O-move played in A, a $\mathcal{P}$-instantiation;
- for each move played in B, a pair of an $\mathcal{O}$-instantiation and of a $\mathcal{P}$-instantiation;

such that all the $\mathcal{O}$-variables appearing in the $\mathcal{O}$-instantiations are different. This turns $\mathtt{u}$ into an instantiated sequence on $A \to (B \to C)$ by forgetting: pointers, $\mathcal{O}$-instantiations for O-moves played in B and $\mathcal{P}$-instantiations for P-moves played in B.

The set of all interaction sequences on A, B and C is noted $\mathtt{int}(A, B, C)$.

An instantiated justified sequence $\mathtt{s}$ on an arena A generates substitutions of the $\mathcal{A}$-variables appearing in the first-order labels of A by the $\mathcal{OP}$-terms appearing in the instantiations. Let $\mathtt{m}$ be an occurrence of move in $\mathtt{s}$ with instantiation $[t_1, \ldots, t_k]$ and let $[x_1, \ldots, x_k]$ be the first-order label of $\mathtt{m}$ in A, we define the substitution $\theta_{\mathtt{m}}$ as $\{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$ if $\mathtt{m}$ is an initial move, and $\theta_{\mathtt{n}} \cup \{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$ where $\mathtt{n}$ is the occurrence of move justifying $\mathtt{m}$ in $\mathtt{s}$ otherwise.

Definition 6 (Play)
A *play* on the arena A is an instantiated justified sequence on A such that:

- polarities of moves are alternating;
- there are no μ-pointers from Opponent moves;
- there is exactly one μ-pointer for each element of the atomic label of each Player move;
- for each μ-pointer going from a formula $Xt_1 \ldots t_k$ labelling (in A) an occurrence of move $\mathtt{m}$ to a formula $Yu_1 \ldots u_p$ labelling (in A) an occurrence of move $\mathtt{n}$, we have $X = Y$, $k = p$, $t_1\theta_{\mathtt{m}} = u_1\theta_{\mathtt{n}}$, ..., $t_k\theta_{\mathtt{m}} = u_k\theta_{\mathtt{n}}$;
- all the $\mathcal{O}$-variables appearing in a $\mathcal{P}$-instantiation have appeared in a previous $\mathcal{O}$-instantiation.

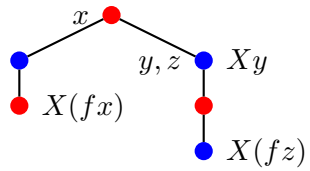
The set of all plays on A is noted $\mathcal{P}_A$. The set of even length plays on A is noted $\mathcal{P}_A^P$. The prefix order on plays is noted $\leq$ and we use the notation $\mathtt{s} \leq^P \mathtt{t}$ ($\mathtt{s}$ is *P-prefix* of $\mathtt{t}$) for $\mathtt{s} \leq \mathtt{t} \wedge \mathtt{s} \in \mathcal{P}_A^P$.

We can summarize the structure put on moves of a play:

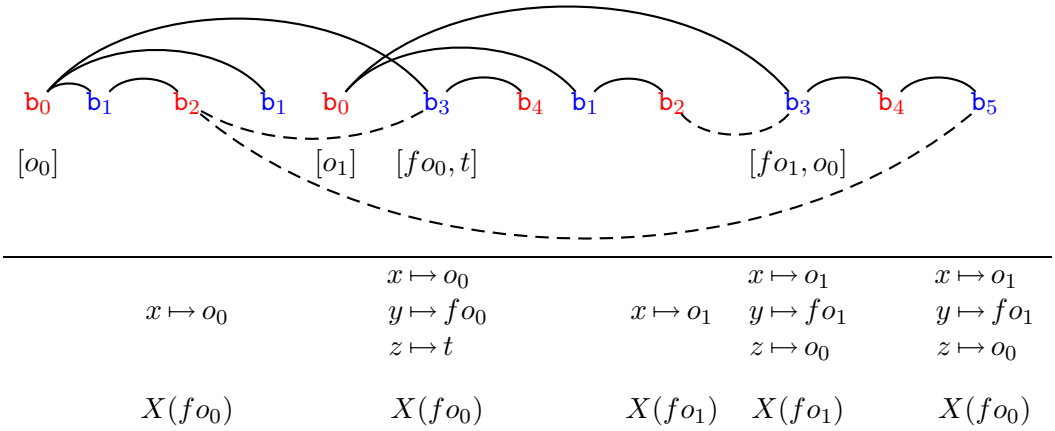
- an Opponent move is equipped with a justification pointer and with an $\mathcal{O}$-instantiation;
- a Player move is equipped with a justification pointer, with a list of μ-pointers and with a $\mathcal{P}$-instantiation.

$\mathcal{O}$-variables are introduced by Opponent and then used by Player.

Example 4
On the arena:



if we name the root as $\mathtt{b}_0$, then we name the moves along the first branch $\mathtt{b}_1$ and $\mathtt{b}_2$ and along the second branch $\mathtt{b}_3$, $\mathtt{b}_4$ and $\mathtt{b}_5$, we have the following play (above the line):



For each occurrence of move $\mathtt{m}$ with a non-empty atomic label $[R]$ (there is no atomic label of greater length in the considered arena), we have indicated (below the line) the corresponding substitution $\theta_{\mathtt{m}}$ and the associated formula $R\theta_{\mathtt{m}}$.

It is thus easy to check that μ-pointers validate the condition on atomic labels given in the definition of play. The formula $R\theta_{\mathtt{m}}$ gives some dynamic content of the move which depends on the position in the play (thus in a proof on the syntactic side). A comment on the logical meaning of this is given in the beginning of Section 6.

We define various notions of projections of sequences of moves.

If $\mathtt{s}$ is an instantiated justified sequence on $A \to B$, $\mathtt{s} \upharpoonright_A$ (resp. $\mathtt{s} \upharpoonright_B$) is the subsequence (with some pointers and some instantiations) of $\mathtt{s}$ containing the moves belonging to A (resp. B), with their justification pointers (except for initial moves of A which do not have justification pointers anymore), with their μ-pointers going to moves in A (resp. B) (the others disappear), and with their instantiations. It is an instantiated justified sequence.

If $\mathtt{u}$ is an interaction sequence on A, B and C, we define the following sequences (with some pointers and some instantiations):

- $\mathtt{u} \upharpoonright_{A \to B}$ is the subsequence of $\mathtt{u}$ containing moves in A and moves in B with their pointers (if they arrive to a move in A or B and are not μ-pointers starting from a Player move of B in $\mathtt{u}$) and with their instantiation for moves in A, and their $\mathcal{O}$-instantiation for P-moves in $\mathtt{u}$ played in B and their $\mathcal{P}$-instantiation for O-moves in $\mathtt{u}$ played in B.

- $\mathtt{u} \upharpoonright_{B \to C}$ is the subsequence of $\mathtt{u}$ containing moves in B and moves in C with their pointers (if they arrive to a move in B or C and are not μ-pointers starting from an Opponent move of B in $\mathtt{u}$) and with their instantiation for moves in C, their $\mathcal{O}$-instantiation for O-moves in $\mathtt{u}$ played in B and their $\mathcal{P}$-instantiation for P-moves in $\mathtt{u}$ played in B.

- $\mathtt{u} \upharpoonright_{A \to C}$ is the subsequence of $\mathtt{u}$ containing moves in A and moves in C with their justification pointer if it arrives to a move in A or C.

  For any initial move $\mathtt{m}$ in A, whose justifier must be an initial move $\mathtt{m}'$ in B itself justified by an initial move $\mathtt{m}''$ in C, we put $\mathtt{m}''$ as justifier of $\mathtt{m}$.

  The μ-pointers of this justified sequence are given by: we put a μ-pointer from the formula R associated with the occurrence of move $\mathtt{m}$ to the formula S associated with the occurrence of move $\mathtt{n}$, if there exists a sequence of μ-pointers $p_1, \ldots, p_n$ $(n > 0)$ in $\mathtt{u}$ such that:

  - the source of p_1 is R associated with the occurrence of move $\mathtt{m}$
  - the target of p_n is S associated with the occurrence of move $\mathtt{n}$
  - the source of p_i is the target of p_{i-1} $(2 \leq i \leq n)$
  - the target of p_i is in B $(1 \leq i \leq n-1)$

  This means that we find a path of μ-pointers from R to S going only through labels of moves in B (if the path contains only one edge, it has not to go through B).

  Since with any move of $\mathtt{u}$ in B are associated both an $\mathcal{O}$-instantiation and a $\mathcal{P}$-instantiation (of the same length), we can define an $\mathcal{O}$-substitution ϑ: the $\mathcal{O}$-variable x is substituted by t if x appears in kth position in the $\mathcal{O}$-instantiation of an occurrence of move $\mathtt{m}$ of $\mathtt{u}$ in B and t is the kth element of the $\mathcal{P}$-instantiation of $\mathtt{m}$. The instantiations in $\mathtt{u} \upharpoonright_{A \to C}$ are obtained from the instantiations in $\mathtt{u}$ by applying ϑ.

The objective of these projections of interaction sequences is to extract candidate plays for $A \to B$, $B \to C$ and $A \to C$, as given in the definition of the composition of strategies below.

Definition 7 (Strategy)
A *strategy* σ on the arena A, denoted $\sigma : A$, is a non-empty set of even length plays which is closed under even length prefixes, and:

- *deterministic*: if $\mathtt{sm} \in \sigma$ and $\mathtt{sn} \in \sigma$ then $\mathtt{sm} = \mathtt{sn}$;

- *uniform*: if $\mathtt{s} \in \sigma$ and ς is an $\mathcal{O}$-renaming then $\mathtt{s}\varsigma \in \sigma$.

A particular kind of strategy playing μ-pointers and instantiations in a very constrained way is useful. A play is *μ-rigid* if:

- the atomic label of a Player move always has the same length as the atomic label of the previous move,

- a μ-pointer is always going to the corresponding element of the atomic label of the previous move,
- the instantiation of a Player move is always the same as the instantiation of the previous move.

A strategy is *μ-rigid* if all its plays are.

In order to define a category, we consider the following identities and composition.

If A is an arena, the *identity* $\mathtt{id}_A$ on $A \to A$ is given by:

$$\mathtt{id}_A = \{\mathtt{s} \in \mathcal{P}^P_{A_1 \to A_2} \mid \forall \mathtt{t} \leq^P \mathtt{s}, \mathtt{t} \upharpoonright_{A_1} = \mathtt{t} \upharpoonright_{A_2} \\ \wedge\ \mu\text{-pointers are going to the corresponding element of the previous move}\}$$

It contains only μ-rigid plays.

If $\sigma : A \to B$ and $\tau : B \to C$ are two strategies, the *composition* of σ and τ is given on $A \to C$ by:

$$\sigma\,;\tau = \{\mathtt{u} \upharpoonright_{A \to C} \in \mathcal{P}^P_{A \to C} \mid \mathtt{u} \in \mathtt{int}(A, B, C) \wedge \mathtt{u} \upharpoonright_{A \to B} \in \sigma \wedge \mathtt{u} \upharpoonright_{B \to C} \in \tau\}$$

These two constructions give rise to strategies and we obtain a category of arenas and strategies (see Appendix B).

2.3 Innocence

In order to restrict the set of strategies to those corresponding to proofs in first-order logic, we introduce the notions of view and innocence.

Definition 8 (View)
A *view* on the arena A is a play $\mathtt{s}$ on A such that:

- Opponent moves in $\mathtt{s}$ are all λ-justified by the preceding move;
- the list of $\mathcal{O}$-variables played by Opponent (obtained by concatenating the $\mathcal{O}$-instantiations in $\mathtt{s}$ according to the order in which they appear in $\mathtt{s}$) is a prefix of the enumeration $(o_i)_{i \in \mathbb{N}}$.

The condition on $\mathcal{O}$-variables is related with the notion of *skeleton* in [Fel85].

If $\mathtt{s}$ is an instantiated justified sequence, the *pre-view* $\underline{\ulcorner \mathtt{s} \urcorner}$ of $\mathtt{s}$ is defined by: $\underline{\ulcorner \varepsilon \urcorner} = \varepsilon$, $\underline{\ulcorner \mathtt{sm} \urcorner} = \underline{\ulcorner \mathtt{s} \urcorner}\mathtt{m}$ if $\mathtt{m}$ is a Player move, $\underline{\ulcorner \mathtt{sm} \urcorner} = \mathtt{m}$ if $\mathtt{m}$ is an initial Opponent move, $\underline{\ulcorner \mathtt{smtn} \urcorner} = \underline{\ulcorner \mathtt{sm} \urcorner}\mathtt{n}$ if $\mathtt{n}$ is an Opponent move justified by $\mathtt{m}$.

The *view* $\ulcorner \mathtt{s} \urcorner$ of $\mathtt{s}$ is obtained from its pre-view by applying the $\mathcal{O}$-renaming required to respect the naming condition of views. If $[x_0, x_1, \ldots, x_n]$ is the list of $\mathcal{O}$-variables played by Opponent in $\underline{\ulcorner \mathtt{s} \urcorner}$ (obtained by concatenating the $\mathcal{O}$-instantiations in $\underline{\ulcorner \mathtt{s} \urcorner}$ according to the order in which they appear), we consider an $\mathcal{O}$-renaming ς satisfying $\varsigma(x_i) = o_i$ for $0 \leq i \leq n$ (we call it a *canonical renaming* induced by $\underline{\ulcorner \mathtt{s} \urcorner}$) and we define $\ulcorner \mathtt{s} \urcorner = \underline{\ulcorner \mathtt{s} \urcorner}\varsigma$ (the value of ς outside $\{x_0, \ldots, x_n\}$ has no impact).

If $\mathtt{s}$ is a play then $\ulcorner \mathtt{s} \urcorner$ is a view and if $\mathtt{s}$ is a view then $\ulcorner \mathtt{s} \urcorner = \mathtt{s}$.

We choose the presentation of *innocent strategies* based on their underlying view functions.

Definition 9 (View function)
A *view function* on the arena A is a non-empty set of even length views on A which is closed under even length prefixes and deterministic: if $\mathtt{sm} \in \sigma$ and $\mathtt{sn} \in \sigma$ then $\mathtt{sm} = \mathtt{sn}$.

Note that a view function is not a strategy since it violates the uniformity condition.

Let σ be a view function on A, its *view closure* $\mathtt{VC}(\sigma)$ is given by $\varepsilon \in \mathtt{VC}(\sigma)$, and if $\mathtt{s} \in \mathtt{VC}(\sigma)$, $\mathtt{smn} \in \mathcal{P}_A$ and $\ulcorner \mathtt{smn} \urcorner \in \sigma$ then $\mathtt{smn} \in \mathtt{VC}(\sigma)$.

Lemma 1 (View closure)
If σ is a view function then $\mathtt{VC}(\sigma)$ is a strategy.

Proof: By definition, $\mathtt{VC}(\sigma)$ is a non-empty P-prefix closed set of even-length plays. If $\mathtt{sm} \in \mathtt{VC}(\sigma)$ and $\mathtt{sn} \in \mathtt{VC}(\sigma)$ then $\ulcorner \mathtt{sm} \urcorner = \ulcorner \mathtt{s} \urcorner \mathtt{m}_0 \in \sigma$ and $\ulcorner \mathtt{sn} \urcorner = \ulcorner \mathtt{s} \urcorner \mathtt{n}_0 \in \sigma$ (where $\mathtt{m}_0$ and $\mathtt{n}_0$ are obtained from $\mathtt{m}$ and $\mathtt{n}$ by applying a canonical $\mathcal{O}$-renaming induced by $\ulcorner \mathtt{s} \urcorner$) thus $\ulcorner \mathtt{sm} \urcorner = \ulcorner \mathtt{sn} \urcorner$ by determinism of σ and finally $\mathtt{sm} = \mathtt{sn}$.

By induction on the length of $\mathtt{s}$, we can show that $\mathtt{s} \in \mathtt{VC}(\sigma)$ implies $\mathtt{s}\varsigma \in \mathtt{VC}(\sigma)$ for any $\mathcal{O}$-renaming ς. This is an easy consequence of $\ulcorner \mathtt{s} \urcorner = \ulcorner \mathtt{s}\varsigma \urcorner$. □

Composition of view functions is given by: $\sigma \,;\, \tau = \{\ulcorner \mathtt{s} \urcorner \mid \mathtt{s} \in \mathtt{VC}(\sigma) \,;\, \mathtt{VC}(\tau)\}$, and the identity view function is $\ulcorner \mathtt{id} \urcorner$.

An *innocent strategy* is a strategy obtained as the view closure of a view function. We will now consider only innocent strategies and just say "strategy". Moreover we will mainly say "strategy" for the underlying view function.

Proposition 3 (Category of innocent games)
Arenas and view functions give a category $\mathcal{G}$.

Proof: All the technical results on strategies corresponding to the categorical structure are developed in Appendix B. □

2.4 Constructions

The notion of arrow arena was already required to define morphisms between arenas. We now turn to other constructions on arenas and strategies to describe the richer structure of the category of games: a control category [Sel01]. We first start with the propositional constructions.

Arena constructions. Let A and B be two arenas:

Empty. The *empty arena* $\top$ is the empty forest.

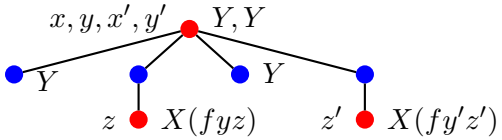
Unit. The *unit arena* $\bot$ is the forest with only one tree with only one node $*$ (empty labels).

Atom. If R is a non-constant atomic formula, the corresponding *atomic arena* R is the unit arena with $[R]$ as atomic label for its unique node (empty first-order label).

Sum. The *sum* $A + B$ of A and B is the concatenation of A and B.

Product. The *product* $A \times B$ of A and B is the merging of A and B. The labels of roots are obtained by concatenation from the labels of the corresponding roots in A and B (the first-order labels of A and B are supposed to be disjoint). A move in $A \times B$ is represented as a pair of moves $(\mathtt{m}, \mathtt{n})$ of A and B such that at least one is initial.

Example 5
Starting from the arena A of Example 2, the product $A \times A$ (where we put "primes" on the second copy) is the arena:



with root named $(\mathtt{a}_0, \mathtt{a}'_0)$, its sons named $(\mathtt{a}_1, \mathtt{a}'_0)$, $(\mathtt{a}_2, \mathtt{a}'_0)$, $(\mathtt{a}_0, \mathtt{a}'_1)$ and $(\mathtt{a}_0, \mathtt{a}'_2)$, the son of $(\mathtt{a}_2, \mathtt{a}'_0)$ is $(\mathtt{a}_3, \mathtt{a}'_0)$ and the son of $(\mathtt{a}_0, \mathtt{a}'_2)$ is $(\mathtt{a}_0, \mathtt{a}'_3)$.

Strategy constructions.

Definition 10 (Linear strategy)
Let σ be a view function on $A \to B$, σ is *linear* if:

- for each initial move $\mathtt{m}$ in B, there is a play $\mathtt{mn}$ in σ with $\mathtt{n}$ in A
- for each view $\mathtt{mns}$ in σ, $\mathtt{n}$ is the unique move in A justified by $\mathtt{m}$

Let $\sigma : A \to C$ and $\tau : B \to D$ be two view functions:

Sum. The view function $\sigma + \tau$ is obtained by the union of the view functions. Its view closure is $\{\mathtt{s} \in \mathcal{P}^P_{A+B \to C+D} \mid \mathtt{s} \restriction_{A \to C} \in \mathtt{VC}(\sigma) \wedge \mathtt{s} \restriction_{B \to D} \in \mathtt{VC}(\tau)\}$. If both σ and τ are linear then $\sigma + \tau$ is linear.

Product. Assume $\sigma : A \to C$ is linear. For each initial move $\mathtt{c}_0$ in C, there is a unique move $\mathtt{a}_0$ in A such that $\mathtt{c}_0\mathtt{a}_0$ belongs to σ. A view $\mathtt{s}$ in $A \times B \to C \times D$ *respects* σ if, for $(\mathtt{c}_0,\mathtt{d}_0)$ its initial move, any move $(\mathtt{a},\mathtt{b})$ in $\mathtt{s}$ with $\mathtt{a}$ initial satisfies $\mathtt{a} = \mathtt{a}_0$ (with $\mathtt{c}_0\mathtt{a}_0 \in \sigma$ and $\mathtt{a}_0$ justified by $\mathtt{c}_0$) and any move $(\mathtt{c},\mathtt{d})$ in $\mathtt{s}$ with $\mathtt{c}$ initial satisfies $\mathtt{c} = \mathtt{c}_0$. If $\mathtt{s}$ respects σ, $\mathtt{s}\wr_\tau$ is obtained by replacing any move $(\mathtt{a}_0,\mathtt{b})$ by $\mathtt{b}$ and any move $(\mathtt{c}_0,\mathtt{d})$ by $\mathtt{d}$ (with the appropriate pointers and instantiations) and by removing the other moves. If $\mathtt{s}$ respects σ, we consider $\mathtt{s}_0$ obtained by replacing any move $(\mathtt{a},\mathtt{b}_0)$ with $\mathtt{a}$ non initial in A and $\mathtt{b}_0$ initial in B by $\mathtt{a}$ and any move $(\mathtt{c},\mathtt{d}_0)$ with $\mathtt{c}$ non initial in C and $\mathtt{d}_0$ initial in D by $\mathtt{c}$ (with the appropriate pointers and instantiations). We define $\varepsilon\wr_\sigma = \varepsilon$ and, if $\mathtt{s}$ is not empty, $\mathtt{s}\wr_\sigma$ is $\mathtt{c}_0\mathtt{a}_0\mathtt{s}_0$ in which $\mathtt{a}_0$ is justified by $\mathtt{c}_0$ and the moves of $\mathtt{s}_0$ enabled by $\mathtt{a}_0$ in A are justified by $\mathtt{a}_0$. We define $\sigma \times \tau = \{\mathtt{s} \in \mathcal{P}^P_{A \times B \to C \times D} \mid \forall \mathtt{t} \leq^P \mathtt{s}, \mathtt{t}$ is a view respecting $\sigma \wedge \mathtt{t}\wr_\sigma \in \sigma \wedge \mathtt{t}\wr_\tau \in \tau\}$.

If τ is linear (but σ is not) we can proceed in a symmetric way for defining $\sigma \times \tau$. If both σ and τ are linear, the two definitions coincide and $\sigma \times \tau$ is linear.

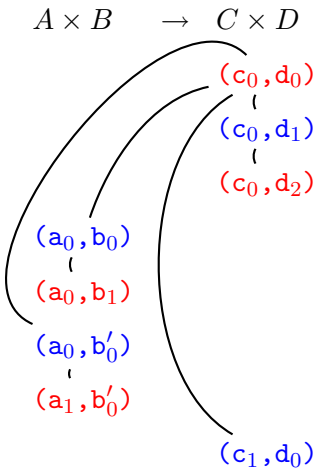
Projections. The linear view function $\ulcorner \mathtt{id}_A \urcorner : A \to A$ is also a linear view function on $A + B \to A$ and on $B + A \to A$.

Diagonal. We can consider moves in $A \to B$ as moves in $A \to B + B$ by identifying the original B with either the left one or the right one. In this way we can see $\ulcorner \mathtt{id}_A \urcorner$ as a set of plays $\mathtt{id}^1_A$ in $A \to A + A$ by considering the left embedding and also as a set of plays $\mathtt{id}^2_A$ in $A \to A + A$ by considering the right embedding. The linear view function Δ_A on $A \to A + A$ is the union of $\mathtt{id}^1_A$ and $\mathtt{id}^2_A$.

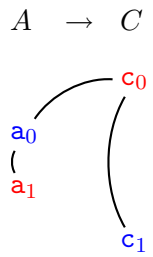
Weakening. The linear view function $\mathtt{wk}_A$ on $\bot \to A$ is $\{\varepsilon\} \cup \{\mathtt{m}* \mid \mathtt{m} \text{ initial in } A\}$ (this means that $\mathtt{m}$ comes with the unique possible $\mathcal{O}$-instantiation leading to a view and that $*$ is justified by $\mathtt{m}$).

Contraction. We consider a play $\mathtt{s}$ in $A \times A \to A$ (we add indexes: $A_1 \times A_2 \to A_0$ to make things clearer). An occurrence of move in A_0 is called a left move if the previous move in $A_1 \times A_2$ was in A_1 (and the same with "right move" and A_2). If only moves from A_0 were played before, we consider it both as a left move and as a right move. We define $\mathtt{s}_l$ as the subsequence of $\mathtt{s}$ containing $\mathtt{m}$: if $(\mathtt{m}, \mathtt{n})$ is an occurrence of move in $A_1 \times A_2$ with $\mathtt{n}$ initial in A_2, or if $\mathtt{m}$ is a left move in A_0. $\mathtt{s}_r$ is given in a symmetric way. $\mathtt{s}_l$ and $\mathtt{s}_r$ can be seen as sequences of moves in $A \to A$. The linear view function $\mathtt{ctr}_A$ on $A \times A \to A$ is $\{\mathtt{s} \in \mathcal{P}^P_{A\times A\to A} \mid \mathtt{s} \text{ view} \wedge \mathtt{s}_l \in \mathtt{id}_A \wedge \mathtt{s}_r \in \mathtt{id}_A\}$.

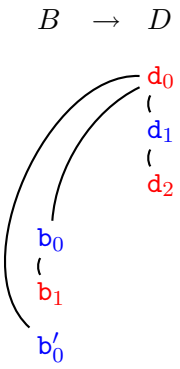
Example 6
The definition of product gives the following kind of view in $\sigma \times \tau$:



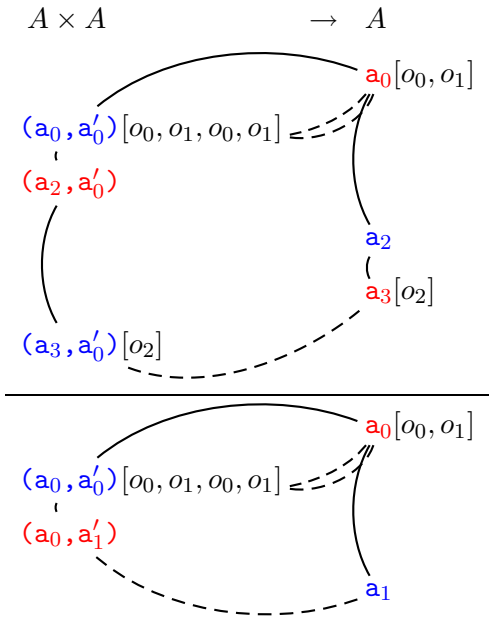
where $\sigma : A \to C$ is linear and contains the view:



and $\tau : B \to D$ contains the view:



Example 7
The (quite complicated) definition of contraction gives the following (simple) views:



where A is the arena of Example 2 and $A \times A$ is described in Example 5.

Theorem 1 (Control category of games)
The category $\mathcal{G}$ of arenas and view functions is a control category.

In this control category, central morphisms are linear strategies.

PROOF: All the technical results on strategies corresponding to the control category structure are developed in Appendix B. □

Definition 11 (Total strategy)
A strategy $\sigma : A$ is *total* if whenever $\mathtt{s} \in \sigma$ and $\mathtt{sm} \in \mathcal{P}_A$, there exists some $\mathtt{smn}$ in σ.

A total strategy is *maximal* for inclusion: if σ is total and $\sigma \subseteq \tau$ then $\sigma = \tau$.

Definition 12 (Finite strategy)
The *size* of a strategy is the sum of the lengths of its views. A strategy is *finite* if its size is finite.

The identity strategy is total and finite, and total and finite strategies compose (see Appendix B). This allows us to define the sub-category $\mathcal{G}^{\mathtt{tf}}$ of $\mathcal{G}$ containing only total finite strategies, which is also a control category (easy to check).

We now turn to the constructions corresponding to quantification.

First-order constructions. Concerning arenas, if A is an arena, x is a $\mathcal{P}$-variable and y is a fresh $\mathcal{A}$-variable, the *quantification* $\forall y A[^{y}/_{x}]$ is obtained by renaming x into y in A and then by pushing y on the first-order label of each root. We will sometimes use the notation $\forall x A$ for this arena (since the particular choice of the name y is not important, see the discussion on bound variables in Section 2.1).

Example 8
The arena of Example 2 is the interpretation of the formula $\forall x \forall y (Y \to (\forall z X(fyz) \to \bot) \to Y)$.

Let $\sigma : A \to B$ be a view function (with a $\mathcal{P}$-variable $x \notin A$), $\forall x.\sigma$ is the view function on $A \to \forall y B[^{y}/_{x}]$ given by: $\forall x.\sigma = \{\varepsilon\} \cup \{(\mathtt{m}[x]\mathtt{s})[^{o_{i+1}}/_{o_i}][^{o_0}/_{x}] \mid \mathtt{ms} \in \sigma\}$ where $\mathtt{m}[x]$ is obtained from $\mathtt{m}$ by pushing x on its instantiation. If σ is linear then $\forall x.\sigma$ is still linear.

Let A be an arena, the *linear* view function $\mathtt{inst}_t$ on $\forall x A \to A[^{t}/_{x}]$ is given by: $\mathtt{inst}_t = \{\mathtt{s} \in \mathcal{P}^{P}_{\forall x A \to A[^{t}/_{x}]} \mid \mathtt{s} \text{ view} \wedge \forall \mathtt{t} \leq^{P} \mathtt{s}, \mathtt{t} \upharpoonright_{\forall x A}[t] = \mathtt{t} \upharpoonright_{A[^{t}/_{x}]}\}$ where $\mathtt{t}[t]$ is obtained from $\mathtt{t}$ by pushing t on the instantiation of its initial move.

2.5 Interpretation of the $\lambda\mu$-calculus

A typing derivation ending with a judgment $\Gamma \vdash M : A \mid \Delta$ is interpreted as a strategy $[\![M]\!]$ on $\sum \Gamma \to A \times \prod \Delta$. Using Theorem 1, there is a canonical way of interpreting the usual propositional constructions of the $\lambda\mu$-calculus in our model (following [Sel01]). Moreover this ensures the validation of those of the $\beta\eta\mu\rho\theta$ equalities which are not dealing with first-order constructs.

For the interpretation of the ($\forall$-introduction) rule, we transform σ into $\forall x.\sigma$ (since the arenas $\forall x(A \times B)$ and $(\forall x A) \times B$ are isomorphic if $x \notin B$). For the interpretation of the ($\forall$-elimination) rule, we transform σ into $\sigma\,;\mathtt{inst}_t$.

Lemma 2 (First-order correctness)
The following equalities are valid through the interpretation in games:

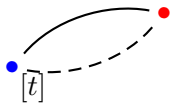
$$\begin{array}{rcll} (\Lambda x.M)\{t\} & =_{\beta} & M[^{t}/_{x}] & : A \\ \Lambda x.M\{x\} & =_{\eta} & M & : \forall x A \qquad x \notin M \\ (\mu\alpha.M)\{t\} & =_{\mu} & \mu\alpha.M[^{[\alpha]L\{t\}}/_{[\alpha]L}] & : A \end{array}$$

PROOF:
- $(\Lambda x.M)\{t\}$: The view function interpreting this term is $\forall x.\sigma\,;(\mathtt{inst}_t \times \mathtt{id}_{\Delta})$ which is $\{\mathtt{s}[^{t}/_{x}] \mid \mathtt{s} \in \sigma\}$. One easily checks it is also the interpretation of $M[^{t}/_{x}]$.
- $\Lambda x.M\{x\}$: An immediate computation shows the interpretation of this term to be the same as the interpretation of M.
- $(\mu\alpha.M)\{t\}$: this case is a consequence of the centrality of the morphism $\mathtt{inst}_t$ in the control category of games (see [dL07, Chapter 7] for example). □

We denote by $\mathcal{G}_1$ (resp. $\mathcal{G}_1^{\mathtt{tf}}$) the full sub-category of $\mathcal{G}$ (resp. $\mathcal{G}^{\mathtt{tf}}$) containing only arenas with at most one element in the atomic labels of moves (called 1-*arenas*). According to the previous interpretation of typing derivations, these categories are expressive enough to interpret the closed terms of our $\lambda\mu$-calculus[3] and we will mainly focus on them in the sequel.

Example 9
The interpretation of the closed term $\lambda a.a\{t\}$ of type $\forall x X x \to Xt$ is the view function containing the empty view and the view:

$$\forall x X x \quad \to \quad Xt$$

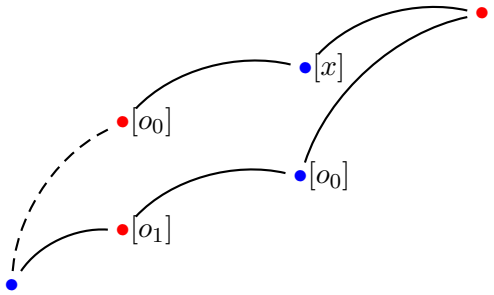


The interpretation of the closed term:

$$\lambda f.(f\{x\})\Lambda y.\lambda d.\mu\alpha.(f\{y\})\Lambda z.\lambda a.\mu\delta[\alpha]a$$

of type $\forall x(\forall y(Xx \to Xy) \to \bot) \to \bot$ is the view function containing the following unique maximal view:

$$\forall x \quad (\quad \forall y \quad (\quad Xx \quad \to \quad Xy \quad) \quad \to \quad \bot \quad) \quad \to \quad \bot$$



In order to prove the completeness of the model, instead of working by induction on the size of strategies and of building incrementally the corresponding term, we will use a more geometric and global approach through $\lambda\mu$-forests (an intermediate notion between terms and strategies in the spirit of Böhm trees — a similar approach is used in [Her97]). Let us start with the simple case of formulas and arenas. A formula in canonical form $\bigwedge_{1\leq i\leq n}\forall\vec{x}(A_1^i \to \cdots \to A_{k_i}^i \to R^i)$ can be rewritten into $\bigwedge_{1\leq i\leq n}[R^i,\vec{x}](A_1^i,\ldots,A_{k_i}^i)$. By considering $[R^i,\vec{x}]$ as an operator/constructor with k_i arguments and by looking at the forest given from the syntactic trees of the $[R^i,\vec{x}](A_1^i,\ldots,A_{k_i}^i)$, we obtain nothing but the arena associated with the original formula. This shows in particular that nodes in the arena are in bijection with occurrences of atomic formulas in a canonical form.

Example 10
The formula given in Example 8 and interpreted by the arena of Example 2 would be represented as $[Y,x,y]([Y],[\bot]([X(fyz),z]))$.

[3]These two categories are not control categories, since the product of two 1-arenas is not a 1-arena in general. The existence of surrounding control categories would allow us to extend the $\lambda\mu$-calculus with a disjunction connective in types. We prefer not to do it since the calculus would become even more complex.

We now develop the same kind of correspondence at the level of terms, $\lambda\mu$-forests and strategies.

Definition 13 ($\lambda\mu$-forest)
A $\lambda\mu$-*forest* is a forest with two additional disjoint finite sets of edges — λ-edges (labelled with a natural number) and μ-edges — and with a list of $\mathcal{OP}$-terms associated to each node and λ- or μ-variables associated to some nodes, satisfying:

- The nodes of even polarity have exactly one son.
- The source of an edge is always a node of odd polarity and the target is always of even polarity. Moreover the target is above the source.
- A node cannot be the source of two different λ-edges or of two different μ-edges.
- If a node of odd polarity is not the source of a λ-edge, it is labelled with a λ-variable. If a node of odd polarity is not the source of a μ-edge, it may be labelled with a μ-variable.
- The list of terms associated with a node of even polarity is a list of $\mathcal{O}$-variables.
- An $\mathcal{O}$-variable appearing in an $\mathcal{OP}$-term of the list associated with a node of odd polarity must appear above in the list associated with a node of even polarity.
- The list of $\mathcal{O}$-variables appearing along a branch (obtained by concatenating the lists associated with the nodes of even polarity in the branch according to the order in which they appear) is a prefix of the enumeration $(o_i)_{i\in\mathbb{N}}$.

If there is no node labelled with a λ- or μ-variable, the $\lambda\mu$-forest is *closed*.

With any $\lambda\mu$-term in canonical normal form is naturally associated a $\lambda\mu$-forest:

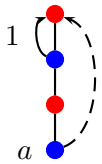
- with $\star$ is associated the empty forest
- with a tuple of $\lambda\mu$-terms is associated the forest whose trees correspond to each $\lambda\mu$-term
- with a $\lambda\mu$-term $\Lambda\vec{x}.\lambda\vec{a}.\underline{\mu\alpha[\beta]}(b\{\vec{t}\})\vec{M}$ is associated the following tree: we first consider the tree with a root `r` which has one son `n` whose sons are the trees corresponding to the $\vec{M}$s then
  - we put the labels b and β on `n`
  - we apply the substitution $[{}^{o_{i+k}}/_{o_i} \mid i \in \mathbb{N}]$ (with $\vec{x} = x_1 \dots x_k$)
  - we associate the list $[\vec{x}]$ with `r` and the list $[\vec{t}]$ with `n`
  - we apply the substitution $[{}^{o_0}/_{x_1}, \dots, {}^{o_{k-1}}/_{x_k}]$
  - for each node labelled with the λ-variable a_i ($\vec{a} = a_1 \dots a_k$), we remove the label and we put a λ-edge with target `r` and label i
  - for each node labelled with the μ-variable α, we remove the label and we put a μ-edge with target `r`

If the $\lambda\mu$-term is closed then the associated $\lambda\mu$-forest is closed.

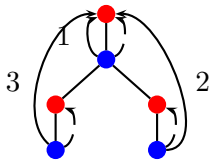
Example 11
We represent λ-edges by plain edges and μ-edges by dashed edges in $\lambda\mu$-forests.

$\lambda a.\mu\delta.a$

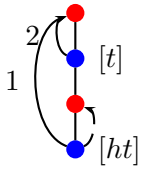
$\lambda f.\mu\alpha.(f)\lambda d.[\alpha]a$



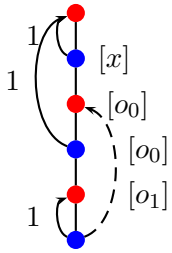
$\lambda f.\lambda a.\lambda b.\mu\gamma[\gamma]((f)\mu\beta[\beta]b)\mu\alpha[\alpha]a$



$\lambda a.\lambda f.(f\{t\})\mu\alpha[\alpha]a\{ht\}$



$\lambda f.(f\{x\})\Lambda y.\lambda d.\mu\alpha.(f\{y\})\Lambda z.\lambda a.\mu\delta[\alpha]a$



All these $\lambda\mu$-forests, except the second one, are closed.

By translating $\lambda\mu$-terms as $\lambda\mu$-forests, there is a loss of information. For example $\lambda a.a$ and $\lambda a.\lambda b.a$ are both translated as:

We have to use types to recover the missing information. A $\lambda\mu$-forest is *typed* if a formula in $\rightarrow$-canonical form (see page 5) is associated with each node in such a way that:

- if the node $\mathtt{n}$ (with formula A) is source of a λ-edge with label i and target $\mathtt{r}$ (with formula B and list of terms $\vec{o}$), then $B = \forall\vec{x}(B_1 \rightarrow \cdots \rightarrow B_k \rightarrow S)$ with $A = B_i[\vec{o}/\vec{x}]$;
- if the node $\mathtt{n}$ (with formula A and list of terms $\vec{t}$) is source of a μ-edge with target $\mathtt{r}$ (with formula B and list of terms $\vec{o}$), then $A = \forall\vec{x}(A_1 \rightarrow \cdots \rightarrow A_k \rightarrow R)$ and $B = \forall\vec{z}(B_1 \rightarrow \cdots \rightarrow B_p \rightarrow S)$ with $R[\vec{t}/\vec{x}] = S[\vec{o}/\vec{z}]$;
- if the node $\mathtt{n}$ of odd polarity has formula $A = \forall\vec{x}(A_1 \rightarrow \cdots \rightarrow A_k \rightarrow R)$ then $R = \bot$ if and only if $\mathtt{n}$ is neither the source of a μ-edge nor labelled with a μ-variable;
- if the node $\mathtt{r}$ of even polarity (with formula A) is the ith son of the node $\mathtt{n}$ (with formula B and list of terms $\vec{t}$), then $B = \forall\vec{x}(B_1 \rightarrow \cdots \rightarrow B_k \rightarrow S)$ with $A = B_i[\vec{t}/\vec{x}]$.

The type of the $\lambda\mu$-forest is the conjunction of the types of its roots.

We can extend the translation from $\lambda\mu$-terms to $\lambda\mu$-forests with types: when translating $\Lambda\vec{x}.\lambda\vec{a}.\mu\alpha[\beta](b\{\vec{t}\})\vec{M}$ of type A with b of type B, we associate A with $\mathtt{r}$ and B with $\mathtt{n}$.

Example 12
The $\lambda\mu$-forests of Example 11 can be turned into typed $\lambda\mu$-forests with the following respective

types:

$$\bot \to X$$
$$((X \to \bot) \to \bot) \to X$$
$$(Y \to X \to Z) \to X \to Y \to Z$$
$$\forall x X x \to (\forall x (X(hx) \to \bot)) \to \bot$$
$$\forall x (\forall y (Xx \to Xy) \to \bot) \to \bot$$

Starting from a typed $\lambda\mu$-forest, we can build a unique typed $\lambda\mu$-term. We decompose the $\lambda\mu$-forest into $\lambda\mu$-trees, we compute the corresponding $\lambda\mu$-terms and the $\lambda\mu$-term associated with the $\lambda\mu$-forest is the induced tuple. Concerning a $\lambda\mu$-tree T, if the root $\mathtt{r}$ has formula $A = \forall \vec{x}(A_1 \to \cdots \to A_k \to R)$, we introduce k fresh λ-variables $a_1, \ldots, a_k$, and a fresh μ-variable α. To any node which is source of a λ-edge with index i and target $\mathtt{r}$, we add the label a_i. To any node which is source of a μ-edge with target $\mathtt{r}$, we add the label α. Let $\mathtt{n}$ be the son of $\mathtt{r}$ and let b and β be the labels obtained for it. Let $\vec{M}$ be the $\lambda\mu$-terms inductively associated with the sub-trees under $\mathtt{n}$. The $\lambda\mu$-term associated with T is $\Lambda\vec{y}\lambda a_1 \ldots \lambda a_k \underline{\mu\alpha[\beta]}(b\{\vec{t}\})\vec{M}$ where $\vec{y}$ are the terms labelling $\mathtt{r}$ and $\vec{t}$ are the terms labelling $\mathtt{n}$.

With any total finite view function on the arena associated with a type A is associated a closed $\lambda\mu$-forest: we consider views ordered with the prefix ordering (so that moves in the views give nodes in the forest), we remove the λ-pointers of O-moves, the other pointers give the (λ and μ) edges of the $\lambda\mu$-forest, the instantiations give the lists of terms. Concerning the labels of λ-edges, if the node $\mathtt{n}$ corresponds to the move $\mathtt{m}$ corresponding itself to the occurrence R of an atomic formula in A and if R appears in a sub-formula $\forall \vec{x}(B_1 \to \cdots \to B_k \to S)$ of A as the final atom of B_i then the λ-edge with source $\mathtt{n}$ has label i.

With any node of a closed typed $\lambda\mu$-forest of type A, we can associate a move of the arena associated with A:

- if $\mathtt{r}$ is the ith root of the $\lambda\mu$-forest, the corresponding move is the ith root of the arena;
- if the node $\mathtt{n}$ of even polarity has a λ-edge with label i to a node with associated move $\mathtt{m}$, the move associated with $\mathtt{n}$ is the ith son of $\mathtt{m}$ in the arena;
- if the node $\mathtt{r}$ of odd polarity is the ith son of the node $\mathtt{n}$ with associated move $\mathtt{m}$, the move associated with $\mathtt{r}$ is the ith son of $\mathtt{m}$ in the arena.

In this way, we can associate a view with any branch of a closed typed $\lambda\mu$-forest. Nodes of even polarity become Opponent moves. Nodes of odd polarity become Player moves. λ-edges give the justification pointers. μ-edges give the μ-pointers. Lists of $\mathcal{OP}$-terms give instantiations. Finally we add justification pointers going from each Opponent move to the preceding one.

Definition 14 (Arena isomorphism)
An *arena isomorphism* f from A to B is a bijection between the nodes of A and the nodes of B which respects the order, but also the atomic labels up to the first-order labels: the move $\mathtt{m}$ and the move $f(\mathtt{m})$ must have first-order labels of the same length, this induces a mapping of the elements of the first-order label of $\mathtt{m}$ to the elements of the first-order label of $f(\mathtt{m})$; using this mapping, the atomic label of any node $\mathtt{n}$ must be mapped to the atomic label of $f(\mathtt{n})$.

Lemma 3 (Isomorphic arenas)
If there exists an arena isomorphism between two arenas, they are isomorphic in the category $\mathcal{G}$.

PROOF: The arena isomorphism f induces a strategy $\{\mathtt{s} \in \mathcal{P}^P_{A\to B} \mid \mathtt{s}\ \mu\text{-rigid} \wedge \forall \mathtt{t} \leq^P \mathtt{s}, f(\mathtt{t} \upharpoonright_A) = \mathtt{t} \upharpoonright_B\}$ which is an isomorphism in $\mathcal{G}$ (see [Lau05a, Proposition 6]). □

Theorem 2 (Equivalence completeness)
The game model is equivalence complete: the categories $\mathcal{S}$ and $\mathcal{G}_1^{tf}$ are equivalent.

PROOF: We have already seen in Section 1.3 that $\mathcal{S}^c$ is equivalent to $\mathcal{S}$. We want to show that the interpretation functor from $\mathcal{S}^c$ to $\mathcal{G}_1^{\mathtt{tf}}$ defines an equivalence of categories. We have established translations from typed $\lambda\mu$-terms in canonical normal form to typed $\lambda\mu$-forests (and back) and from typed $\lambda\mu$-forests to view functions on the corresponding arena (and back). One can check that all these correspondences are one-to-one. Moreover the view function associated with a given typed $\lambda\mu$-term in canonical normal form by these correspondences is the same as the one obtained through the interpretation given in the beginning of this section (and coming from the control category structure of $\mathcal{G}^{\mathtt{tf}}$). This shows the interpretation functor to be full and faithful.

Finally, any arena is isomorphic to the interpretation of a type in canonical form: by Lemma 3, it is enough to prove that any arena is arena isomorphic to the interpretation of a type in canonical form. This is done by induction on the number of nodes of the arena. □

Corollary 2.1 (Canonical forms are canonical)
Two canonical normal forms equal up to $\beta\eta\mu\rho\theta$ are equal.

PROOF: If M and N are two canonical forms such that $M \simeq_{\beta\eta\mu\rho\theta} N$ then $[\![M]\!] = [\![N]\!]$ thus $M = N$ by faithful completeness. □

The concrete meaning of Theorem 2 is mainly that any total finite strategy on a 1-arena is the interpretation of a unique closed $\lambda\mu$-term in canonical normal form.

This ends the description of our game model.

3 Related models

From the game model of first-order classical logic (and the associated completeness result) described in the previous section, we will define complete models for different systems. Some of these derived models are strongly related with (or even equal to) already known ones.

3.1 Intuitionistic restriction

Inside the category $\mathcal{G}_1$, particular strategies (let us call them λ-*strategies*) are obtained by asking μ-pointers to always have the preceding move as target (μ-rigid strategies are a particular kind of λ-strategies). Through the completeness results above (Theorem 2), they correspond to $\lambda\mu$-terms in which the μ and $[\_]$ constructions are always used together in the shape $\mu\alpha[\alpha]$. By θ equivalence, such a $\lambda\mu$-term is equal to a λ-term (all the $\mu\alpha[\alpha]$ can be erased). As a consequence, λ-strategies provide an equivalence complete model of the Church style first-order λ-calculus and, thus, of first-order intuitionistic logic.

3.2 λ-calculus over one/many atom(s)

We consider the simply typed λ-calculus over one atom. We interpret this atom exactly as $\bot$. In this case there are no labels on the arenas associated with types (neither atomic formulas, nor first-order variables). Such arenas are called *ground arenas* and the full sub-category $\mathcal{G}_{00}$ of $\mathcal{G}$ is given by the restriction to ground arenas. $\mathcal{G}_{00}$ is exactly the category of games presented in [DHR96] since plays are using neither instantiations nor μ-pointers. As shown in [DHR96], it is a fully complete (and even equivalence complete) game model of the simply typed λ-calculus over one atom.

To take into account multiple atoms, we go back to arenas with atomic labels on nodes. By considering atoms as 0-ary relation symbols, types are interpreted as arenas with an atomic label of length 1 associated with each node. Simply typed λ-terms are interpreted as λ-strategies (introduced just above) and μ-pointers are replaced by the condition that the (unique element of the) atomic label of a Player move is always the same as the atomic label of the previous move (exactly in the spirit of *token-reflecting* strategies of [Mur01, page 122]). This extends the model of [DHR96] to an equivalence complete model of the general simply typed λ-calculus.

Equivalently this gives a complete game model for Π^1 formulas. Indeed we can distinguish three levels in full completeness results for logics with propositional atomic formulas and second-order quantification:

1. only constant atomic formulas (or just one atomic formula since it can be identified with $\bot$) thus no quantification;
2. many atomic formulas but no quantification (or equivalently Π^1 formulas since outermost universal quantification has no impact), this is the level corresponding to propositional logic;
3. general quantification over propositional variables (this is much more difficult and will not be addressed here, see Section 6 for future work and references).

3.3 Well bracketed HO/N games (propositional logic)

In the original works on HO/N games [HO00, Nic94], nodes in arenas had an additional labelling with `Q`/`A` labels corresponding to a notion of *questions* and *answers*. We are going to compare the information encoded with questions and answers and the one given through μ-pointers.

In this section we consider games (arenas and strategies) without first-order information. In order to avoid confusion with strategies as given in [HO00], (innocent) strategies as used in the previous sections will be called *μ-strategies* here.

Definition 15 (`QA`-arena)
A `QA`-*arena* is an arena (without any first-order label) such that if a node has a non-empty atomic label on it, then this label is a singleton and the node is a leaf and is not a root.

Labelled nodes are called *answers* and the others are called *questions*.

Definition 16 (Label-rigid strategy)
A play `s` on a `QA`-arena is *rigid* if for any P-prefix `tmn` of `s`, `m` is an answer if and only if `n` is. It is *label-rigid* if moreover the labels of `m` and `n` are the same.

A strategy on a `QA`-arena is *rigid* (resp. *label-rigid*) if all its plays are rigid (resp. label-rigid).

The notion of rigidity corresponds to having both *forward rigidity* and *backward rigidity* as defined in [HL06]. In the case (as here) where answers are leaves and are not roots, forward rigidity

is then the same thing as the notion of rigidity introduced in [DH01]. Label-rigidity is strongly related with token-reflection in [Mur01, page 122].

QA-arenas and label-rigid strategies define a control category [Lau05b] and thus a model of the $\lambda\mu$-calculus (with or without totality). Let us call it the QA-*game model*.

A rigid view of even length contains either only questions or its only answers are its last two moves (conversely a view satisfying one of these properties is rigid).

Definition 17 (Well bracketed strategy)
A view $\mathtt{s}$ on a QA-arena is *well bracketed* if any Player answer of $\mathtt{s}$ is justified by the last Opponent question. A strategy on a QA-arena is *well bracketed* if all its views are well bracketed.

For a rigid view, being well bracketed means either containing only questions or being of the shape $\mathtt{s}q_1\widehat{q_2a_1}a_2$.

Definition 18 (Folding and unfolding of arenas)
Let A be an arena, the *unfolding* of A is the QA-arena $\widehat{A}$ obtained in the following way: to any node $\mathtt{m}$ of A having atomic label $[X_1,\dots,X_n]$, we add n new sons (denoted $\widehat{\mathtt{m}}_{X_i}$ and labelled with X_i, $1\leq i\leq n$) and we erase the atomic label of $\mathtt{m}$.

Let A be a QA-arena, the *folding* of A is the arena $\underline{A}$ obtained in the following way: we remove all the answers of A and for each of them we put its label in the atomic label of its father.

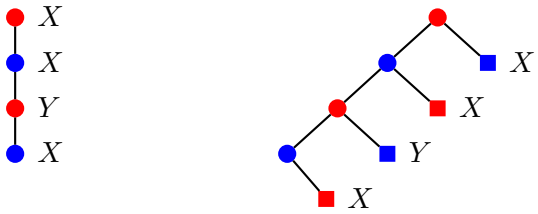
Example 13
The arena corresponding to the type X and its unfolding (a QA-arena with answers represented as squares), which is the interpretation of X in the QA-model, are:

The interpretation of $\bot$ is the same in the two models and is its own folding/unfolding.

Definition 19 (Folding and unfolding of strategies)
Let A be an arena and $\mathtt{s}$ be a view on A, the *unfolding* $\widehat{\mathtt{s}}$ of $\mathtt{s}$ is the set of views given by $\widehat{\varepsilon}=\{\varepsilon\}$ and $\widehat{\mathtt{smn}}=\{\mathtt{s}_0\mathtt{mn}\}\cup\{\mathtt{s}_0\mathtt{mn}\widehat{\mathtt{n}}_X\widehat{\mathtt{n}'}_X \mid X \text{ in the atomic label of } \mathtt{n}\}$ where $\mathtt{s}_0$ is $\mathtt{s}$ without its μ-pointers, the μ-pointer of $\mathtt{smn}$ starting from the element X of the atomic label of $\mathtt{n}$ goes to the element X of the atomic label of the occurrence of move $\mathtt{n}'$, and $\widehat{\mathtt{n}'}_X$ points to this occurrence of $\mathtt{n}'$. The unfolding $\widehat{\sigma}$ of a μ-strategy σ (in fact of its view function) on A is the union of the unfoldings of its views.

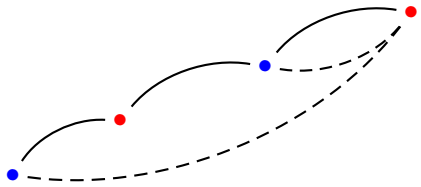
Let A be a QA-arena, σ be a total label-rigid strategy on A and $\mathtt{s}$ be a view in σ ending with a question (more precisely, not ending with an answer), the *folding* $\underline{\mathtt{s}}$ of $\mathtt{s}$ is the view on $\underline{A}$ given by $\underline{\varepsilon}=\varepsilon$ and $\underline{\mathtt{smn}}=\underline{\mathtt{s}}\mathtt{mn}$ where the μ-pointers of $\mathtt{n}$ are obtained in the following way: for each atomic label X of $\mathtt{n}$ in $\underline{A}$, there is exactly one corresponding answer $\mathtt{n}_X$ in A, we consider the only play of the shape $\mathtt{smnn}_X\mathtt{n}'_X$ (for some $\mathtt{n}'_X$ which is an answer pointing to some $\mathtt{n}'$) in σ and we put a μ-pointer from the label X of $\mathtt{n}$ to the label X of $\mathtt{n}'$. The folding of σ is the set of the foldings of its question-ending views.

Example 14
The arena corresponding to the type $(((X\to Y)\to X)\to X)$ and its unfolding are:



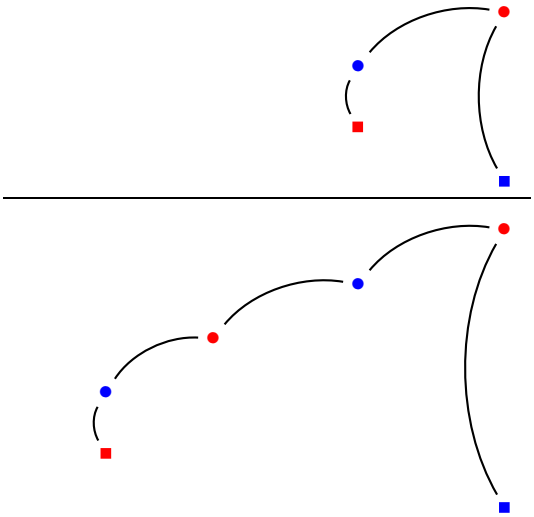
The following view with its prefixes of even length define a μ-strategy on the starting arena:

$$(\ (\ X\ \to\ Y\ )\ \to\ X\ )\ \to\ X$$



The maximal views of its unfolding are:

$$(\ (\ X\ \to\ Y\ )\ \to\ X\ )\ \to\ X$$



Lemma 4
If σ is a μ-strategy and $\mathtt{s}$ *and* $\mathtt{t}$ *are two plays in σ which differ only on their μ-pointers then* $\mathtt{s} = \mathtt{t}$.

Proof: By induction on the common length of $\mathtt{s}$ and $\mathtt{t}$, or by Lemma 9. □

Lemma 5
If $\mathtt{s} \in \widehat{\sigma}$ *does not end with an answer, there exists a unique play* $\mathtt{t}$ *in σ such that* $\mathtt{s} \in \widehat{\mathtt{t}}$. *Moreover* $\mathtt{s}$ *is obtained from* $\mathtt{t}$ *by removing its μ-pointers.*

Proof: The existence of $\mathtt{t}$ is given by definition of $\widehat{\sigma}$ and $\mathtt{t}$ is such that removing its μ-pointers gives $\mathtt{s}$. If there exist two plays satisfying the required constraints, they must differ only on their μ-pointers (since forgetting them leads to $\mathtt{s}$ in both cases), thus they are equal by Lemma 4. □

Theorem 3 (Completeness of unfolding)
Let A be an arena, $\sigma \mapsto \widehat{\sigma}$ and $\tau \mapsto \underline{\tau}$ define a bijection pair between total μ-strategies on A and total label-rigid strategies on $\widehat{A}$.

PROOF: We are going to prove the following statements:

1. If σ is a μ-strategy on A then $\widehat{\sigma}$ is a label-rigid strategy on $\widehat{A}$ (and if σ is total then so is $\widehat{\sigma}$).
2. If τ is a total label-rigid strategy on A then $\underline{\tau}$ is a total μ-strategy on $\underline{A}$.
3. $\underline{(\widehat{A})} = A$ and $\widehat{(\underline{A})} = A$.
4. If σ and τ are total, $\underline{(\widehat{\sigma})} = \sigma$ and $\widehat{(\underline{\tau})} = \tau$.

First statement:

- If $\mathtt{smn} \in \widehat{\sigma}$, by construction, either $\mathtt{m}$ is a question and so is $\mathtt{n}$ or $\mathtt{m}$ is an answer and $\mathtt{n}$ is also an answer and has the same label. As a consequence, $\mathtt{smn}$ is label-rigid.
- $\widehat{\sigma}$ contains ε, contains only views and is P-prefix closed by definition.
- If $\mathtt{smn} \in \widehat{\sigma}$ and $\mathtt{smn'} \in \widehat{\sigma}$, we first consider the case where $\mathtt{m}$ is a question (then both $\mathtt{n}$ and $\mathtt{n'}$ are questions). By Lemma 5, there exist two plays $\mathtt{tmn}$ and $\mathtt{t'mn'}$ in σ such that $\mathtt{smn}$ (resp. $\mathtt{smn'}$) is obtained from $\mathtt{tmn}$ (resp. $\mathtt{t'mn'}$) by removing its μ-pointers. By Lemma 4, we know that $\mathtt{t} = \mathtt{t'}$ so that, by determinism of σ, $\mathtt{tmn} = \mathtt{t'mn'}$ and thus $\mathtt{smn} = \mathtt{smn'}$.
  In the case where $\mathtt{m}$ is an answer (then both $\mathtt{n}$ and $\mathtt{n'}$ are answers and these three moves have the same label), then $\mathtt{smn}$ and $\mathtt{smn'}$ belongs to some $\widehat{\mathtt{t}}$ and $\widehat{\mathtt{t'}}$ that can only differ on their μ-pointers thus, by Lemma 4, $\mathtt{t} = \mathtt{t'}$. As a consequence $\mathtt{n}$ and $\mathtt{n'}$ are the same move (the unique answer of $\widehat{A}$ corresponding to the label X of the μ-justifier of the label X of the last move of $\mathtt{s}$) and have the same justifier (this μ-justifier of the last move of $\mathtt{s}$).
- We now look at totality. If $\mathtt{s} \in \widehat{\sigma}$ and $\mathtt{sm}$ is a view on $\widehat{A}$ then $\mathtt{s}$ contains only questions and, by Lemma 5, we can find a play $\mathtt{t}$ in σ such that $\mathtt{s}$ is obtained from $\mathtt{t}$ by removing its μ-pointers. First, if $\mathtt{m}$ is a question, by totality of σ, there exists $\mathtt{tmn}$ in σ and thus $\mathtt{smn}$ (obtained from it by removing the μ-pointers) is in $\widehat{\sigma}$. Second, if $\mathtt{m}$ is an answer, by definition of $\widehat{t}$, there exists some $\mathtt{n}$ such that $\mathtt{smn} \in \widehat{t} \subseteq \widehat{\sigma}$.

Second statement:

- We first check that $\mathtt{s} \in \underline{\tau}$ is a well defined play. There is exactly one μ-pointer for each formula of each Player move: each such occurrence leads to an answer in $\widehat{A}$ thus corresponds to an Opponent move which has been played in some play of τ by totality.
- Then, by definition, $\underline{\tau}$ is a non-empty P-prefix closed set of even length views.
- If $\mathtt{smn} \in \underline{\tau}$ and $\mathtt{smn'} \in \underline{\tau}$, then $\mathtt{n} = \mathtt{n'}$ and they have the same justification pointer by determinism of τ. Finally their μ-pointers are the same since they are computed in the same way, thus $\mathtt{smn} = \mathtt{smn'}$.
- $\underline{\tau}$ is total by immediate application of the totality of τ.

Third statement:

- The moves of $\underline{(\widehat{A})}$ are the moves of A and their atomic labels are obtained by moving them on a new leaf and then moving them back to their father so that $\underline{(\widehat{A})} = A$.
- The questions of $\widehat{(\underline{A})}$ are the moves of $\underline{A}$ which are the questions of A. The answers of $\widehat{(\underline{A})}$ corresponds to labels of moves of $\underline{A}$ which corresponds to answers of A thus $\widehat{(\underline{A})} = A$.

Fourth statement: for the two equalities, we consider total strategies and this allows us to prove only one inclusion in each case since totality entails maximality.

- We consider a play in $\underline{(\widehat{\sigma})}$, and we prove by induction on its length that it belongs to σ. If it is ε then it belongs to σ. Otherwise it is of the shape $\underline{\mathtt{smn}}$ with $\mathtt{smn} \in \widehat{\sigma}$ and $\mathtt{n}$ is a question in $\widehat{A}$. By Lemma 5, this entails that we can find a play $\mathtt{t} \in \sigma$ such that $\mathtt{smn} \in \widehat{\mathtt{t}}$, and $\mathtt{smn}$ is obtained from $\mathtt{t}$ by removing all the μ-pointers. $\underline{\mathtt{s}}$ is a P-prefix of $\underline{\mathtt{smn}}$ thus $\underline{\mathtt{s}} \in \underline{(\widehat{\sigma})}$ and, by induction hypothesis, $\underline{\mathtt{s}} \in \sigma$. The only possible difference between $\mathtt{t}$ and $\underline{\mathtt{smn}}$ is about the μ-pointers of $\mathtt{n}$, but these μ-pointers are built from the elements of $\widehat{\mathtt{t}}$ so that they are exactly the same as the corresponding μ-pointers in $\mathtt{t}$. Finally $\underline{\mathtt{smn}} = \mathtt{t} \in \sigma$.
- We prove, by induction on the length of $\mathtt{t}$, that if $\mathtt{t} \in \underline{\tau}$ then $\widehat{\mathtt{t}} \subseteq \tau$. First, $\widehat{\varepsilon} \subseteq \tau$. Otherwise we have $\mathtt{t} = \mathtt{t}'\mathtt{mn}$. Let $\mathtt{t}_0\widehat{\mathtt{n}}_X\widehat{\mathtt{n}'}_X$ be an element of $\widehat{\mathtt{t}}$, we easily see that $\mathtt{t}_0 \in \tau$. We look at the μ-pointer of $\mathtt{n}$ corresponding to the label X in $\mathtt{t}'\mathtt{mn}$: it goes to the label X of the move $\mathtt{n}'$, but since it is an element of $\underline{\tau}$, this means that $\mathtt{t}_0\widehat{\mathtt{n}}_X\widehat{\mathtt{n}'}_X \in \tau$. □

Proposition 4 (Comparing the two models)
Let M be a $\lambda\mu$-term (resp. A be a propositional formula) interpreted as σ (resp. A_0) in our game model and as τ (resp. A_1) in the QA*-game model, we have $\sigma = \underline{\tau}$ (resp. $A_1 = \underline{A_0}$).*

Proof: The part concerning formulas and arenas is obtained by a simple induction.

Concerning terms and strategies, we prove it by induction on the term M with the interesting cases given by the $\mu\alpha.M$ and $[\alpha]M$ constructions. Since σ and $\underline{\tau}$ are total, it is enough to prove $\underline{\tau} \subseteq \sigma$.

- The interpretation of $\mu\alpha.M$ is almost the same as the interpretation of M.
- For a term of the shape $[\alpha]M$, by induction hypothesis, if σ_0 (resp. τ_0) is the interpretation of M in our model (resp. in the QA-model), $\sigma_0 = \underline{\tau_0}$. We go by induction on the length of a play of τ for proving $\underline{\tau} \subseteq \sigma$. Assume $\mathtt{smn} \in \tau$ where $\mathtt{n}$ is a question, by induction hypothesis, $\underline{\mathtt{s}} \in \sigma$. There is a unique $\underline{\mathtt{smn}_1}$ in σ, we want to show $\mathtt{n}$ and $\mathtt{n}_1$ to be the same move with the same pointers in $\underline{\mathtt{smn}_1}$ and in $\underline{\mathtt{smn}}$. Since $\sigma_0 = \underline{\tau_0}$, it comes from a look at the contraction strategies on $A \times A \to A$ ($A_1 \times A_2 \to A_0$ with meaningless indexes) in both models. We directly have that $\mathtt{n}$ and $\mathtt{n}_1$ come from the same node in the arena with the same justification pointer. Concerning the μ-pointers, in our model each μ-pointer for a move in A_0 coming from a move in A_1 is given in a μ-rigid way and goes to the move in A_1 (and the same in the other direction). In the QA-model, this corresponds to an answer $\widehat{n}_X$ played by Opponent in A_0 and copied in A_1 (with a justification pointer going to the last Opponent move in A_1 in a well-bracketed way). After folding, one obtains the same μ-pointer. □

We now consider Theorem 3 in the particular case where arenas are trees with at most one label on each node and where QA-arenas never have two answers with the same father. This corresponds to the interpretations of types built with propositional variables, $\bot$ and $\to$.

Proposition 5 (Full completeness of the QA-model)
Let A be a formula built with propositional variables, $\bot$ and $\to$, and let σ be a total finite label-rigid strategy on the QA-arena interpreting A,

- *σ is the interpretation of a $\lambda\mu$-term of type A*
- *σ is the interpretation of a λ-term if and only if σ is well bracketed*

PROOF: These two results are known and come with direct proofs by induction on the size of the strategy. We give here an alternative proof using our games and the notion of folding.

By Theorem 3, $\underline{\sigma}$ is a total finite μ-strategy on $\underline{A}$. By Theorem 2, $\underline{\sigma}$ is the interpretation of a $\lambda\mu$-term M of type A. By Proposition 4, the interpretation of M in the QA-game model is σ.

Finally, $\underline{\sigma}$ is the interpretation of a λ-term if and only if the μ-pointers are always going to the previous move. This corresponds in σ to the fact that any view is well bracketed. □

There is a purely syntactic counterpart to these semantic results: starting from a $\lambda\mu$-term, it is possible to compute the associated strategy, then to unfold it and by completeness (of our first model as in Section 3.2, not of the QA-game model) to get back a λ-term. This is a translation of the simply typed $\lambda\mu$-calculus into the simply typed λ-calculus already studied in [Par97]. The unfolding of arenas corresponds to a translation of simple types into simple types (see Table 5). A typing judgment $\Gamma \vdash M : A \mid \Delta$ is translated as $\Gamma^\star, \Delta^\bullet \vdash M^\star : A^\star$ where $\Delta^\bullet$ is obtained by transforming any $\alpha : A_1 \to \cdots \to A_n \to X$ into $\alpha_1 : A_1^\star, \ldots, \alpha_n : A_n^\star, \alpha : X$, and any $\alpha : A_1 \to \cdots \to A_n \to \bot$ into $\alpha_1 : A_1^\star, \ldots, \alpha_n : A_n^\star$. The translation of $\lambda\mu$-terms is given in Table 5.

The present analysis is developed in the context of propositional logic since the two-moves game model was already known. However following these ideas of folding and unfolding of arenas, one could also define a first-order two-moves game model in correspondence with the one-move one and providing an encoding of the μ-pointers.

3.4 Forgetting structure

In the spirit of comparing the various game models for various logical systems presented before, we can define forgetful transformations between them.

This is mainly a digressive section since the remarks mentioned here can easily be given without any use of game semantics. However, we find particularly straightforward to understand them by means of games, following the idea of the forgetful functor GR used in Appendix B.

From Church style first-order to Curry style first-order. In type systems dealing with quantification, we usually have the choice between two main presentations: Church style systems and Curry style systems. The first ones are explicitly mentioning the quantification inference rules inside terms (as we do here from the beginning) while the second ones are not modifying the typed term when dealing with such a rule:

$$\frac{\Gamma \vdash M : A \mid \Delta}{\Gamma \vdash M : \forall x A \mid \Delta}\, x \notin \Gamma, \Delta \qquad\qquad \frac{\Gamma \vdash M : \forall x A \mid \Delta}{\Gamma \vdash M : A[^t/_x] \mid \Delta}$$

Types.

$$X^\star = X \to \bot$$
$$\bot^\star = \bot$$
$$(A \to B)^\star = A^\star \to B^\star$$

Typed terms.

$$\begin{array}{rll}
a^\star = a & & \\
(\lambda a.M)^\star = \lambda a.M^\star & & \\
((M)N)^\star = (M^\star)N^\star & & \\
([\alpha]M)^\star = (M^\star)\alpha_1 \dots \alpha_n\alpha & & \text{if } \alpha \text{ has type } A_1 \to \dots \to A_n \to X \\
([\alpha]M)^\star = (M^\star)\alpha_1 \dots \alpha_n & & \text{if } \alpha \text{ has type } A_1 \to \dots \to A_n \to \bot \\
(\mu\alpha.M)^\star = \lambda\alpha_1 \dots \lambda\alpha_n\lambda\alpha.M^\star & & \text{if } \alpha \text{ has type } A_1 \to \dots \to A_n \to X \\
(\mu\alpha.M)^\star = \lambda\alpha_1 \dots \lambda\alpha_n.M^\star & & \text{if } \alpha \text{ has type } A_1 \to \dots \to A_n \to \bot
\end{array}$$

Table 5: A translation from the $\lambda\mu$-calculus to the λ-calculus

From these considerations, it is absolutely immediate by starting from a Church style typed $\lambda\mu$-term of type A and by erasing the $\Lambda x.\_$ and $\_\{t\}$ constructions to get a Curry style typed $\lambda\mu$-term of type A.

An example is given by:

$$\begin{array}{l}
A_1 = \forall x(\forall y(Xx \to Xy) \to \bot) \to \bot \\
M_1 = \lambda f.(f\{x\})\Lambda y.\lambda d.\mu\alpha.(f\{y\})\Lambda z.\lambda a.\mu\delta[\alpha]a \\
M_2 = \lambda f.(f)\lambda d.\mu\alpha.(f)\lambda a.\mu\delta[\alpha]a
\end{array}$$

where M_1 is of type A_1 in the Church style system and M_2 is of type A_1 in the Curry style system.

Our game models are useless at this level since we have not considered models of Curry style systems. Let us go to the next step.

From Curry style first-order to classical propositional. The term language is the same for a Curry style first-order type system and for a propositional type system. If M is such a $\lambda\mu$-term typable with type A in the Curry style first-order system, then M is typable with type A' in the system of simple types (that is in propositional logic) where A' is obtained by erasing all the first-order information in A (*i.e.* $\forall x.B \mapsto B$ and $X\vec{t} \mapsto X$).

Proposition 6 (Church to propositional)
By composing these two steps we obtain a $\lambda\mu$-term typed in the simple types system from a $\lambda\mu$-term typed in the Church style first-order system.

PROOF: If σ is a first-order strategy on the first-order arena A, by removing all the instantiation information, we obtain a propositional strategy on the propositional arena obtained from A

by removing the first-order information in labels (we remove the first-order labels and we apply $X\vec{t} \mapsto X$ in the atomic labels).

Through the correctness and full completeness theorems, these transformations described on strategies and arenas exactly correspond to the syntactic transformations going from Church style typed $\lambda\mu$-terms to simply typed $\lambda\mu$-terms. □

As an example, the $\lambda\mu$-term M_2 above has type A_1 in the Curry style type system but also the type $A_3 = ((X \to X) \to \bot) \to \bot$ in the system of simple types.

From classical propositional to intuitionistic propositional. It is possible to go one step further by erasing all the $\mu\alpha$ and $[\alpha]$ constructs in a simply typed $\lambda\mu$-term of type A, and by transforming A into A' by mapping all the propositional variables of A to the same atomic formula $\bot$.

Proposition 7 (Classical to intuitionistic)
This transformation gives a simply typed λ-term from a simply typed $\lambda\mu$-term.

PROOF: Starting from a propositional strategy σ on the arena A and by removing all the μ-pointers, we obtain a strategy on A' in the model with one move for atoms [DHR96] where A' is obtained from A by removing all the labels.

By correctness and full completeness of these models, we derive the syntactic result. □

Going on with our example, while M_2 has type A_3, the corresponding λ-term:

$$M_4 = \lambda f.(f)\lambda d.(f)\lambda a.a$$

has type $A_4 = ((\bot \to \bot) \to \bot) \to \bot$ in the simply typed λ-calculus with one atom denoted $\bot$.

The global move from the Church style classical first-order game model to the intuitionistic propositional game model is nothing but the application of the GR functor (see Appendix B).

3.5 Linear $\lambda\mu$-calculus

Following the definition of the linear λ-calculus, the *linear $\lambda\mu$-calculus* is one of the possible presentations of a linearized version of classical logic [Lau02, Chapter 14]. A closed $\lambda\mu$-term is linear if each variable (λ-variable or μ-variable) has exactly one occurrence (two if we count the occurrence with the binder λ or μ). A typical example is the term $\lambda f.\mu\alpha.(f)\lambda a.[\alpha]a$ of type $((X \to \bot) \to \bot) \to X$.

Since the linear $\lambda\mu$-calculus is defined as a restriction of the $\lambda\mu$-calculus, any model of the $\lambda\mu$-calculus is a model of the linear $\lambda\mu$-calculus. However it is not always easy to define a fully complete sub-model for the linear sub-calculus. Various works on game semantics propose ways of going in this direction [Gir01, FH02, Lai05] by mainly asking for each move to be played once in a strategy. We are going to show that it is also possible to use the pointers (the key point being the introduction of μ-pointers in our work).

Concerning λ-variables (thus λ-pointers), the very natural definition of a *λ-linear strategy* is just to ask, in the tree of views, that for each occurrence of an Opponent move `m` there is exactly one occurrence of each Player move enabled by `m` which is λ-pointing to `m`.

We apply the same kind of idea to μ-pointers: a strategy is *μ-linear* if, in the tree of views, there is exactly one μ-pointer from a Player move going to each atomic label of each Opponent move. A particular case of μ-linear strategy is given by the λ-strategies of Section 3.1.

Proposition 8 (Linear equivalence completeness)
λ-linear and μ-linear strategies give an equivalence complete model of the linear $\lambda\mu$-calculus.

PROOF: Easy to check by following the proofs of Section 2.5. □

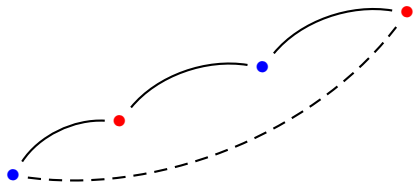
These ideas can easily be extended to notions of λ-affine and μ-affine strategies. Notice that being linear or affine depends on the type associated with a strategy as shown in the following examples.

Example 15
The simplest proof of the double negation elimination:

$$\vdash \lambda f.\mu\alpha.(f)\lambda a.[\alpha]a : ((X \to \bot) \to \bot) \to X$$

corresponds to a λ-linear and μ-linear strategy:

$$(\quad (\quad X \quad \to \quad \bot \quad) \quad \to \quad \bot \quad) \quad \to \quad X$$

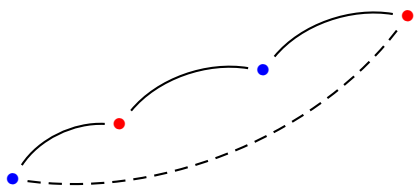


By slightly modifying the type, we have to slightly modify the proof:

$$\vdash \lambda f.\mu\alpha.(f)\lambda a.\mu\delta[\alpha]a : ((X \to Y) \to \bot) \to X$$

which corresponds to a λ-linear and μ-affine strategy (which is not μ-linear):

$$(\quad (\quad X \quad \to \quad Y \quad) \quad \to \quad \bot \quad) \quad \to \quad X$$

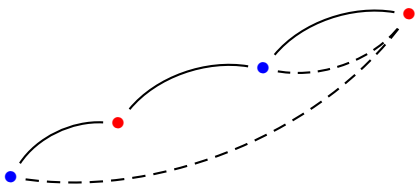


The simplest proof of Peirce's law:

$$\vdash \lambda f.\mu\alpha[\alpha](f)\lambda a.\mu\delta[\alpha]a : ((X \to Y) \to X) \to X$$

corresponds to the λ-linear strategy (which is not μ-affine) already given in Example 14:

$$(\quad (\quad X \quad \to \quad Y \quad) \quad \to \quad X \quad) \quad \to \quad X$$



4 Type isomorphisms

It is proved in Appendix A.2 that all the equations of Table 3 are syntactic type isomorphisms. We are going to prove that no other isomorphism is syntactically valid, by means of game semantics through the method developed in [Lau05a].

Definition 20 (Zig-zag play)
A play $\mathtt{s}$ in the arena $A \to B$ is a *zig-zag play* if it is μ-rigid (see page 13) and:

- each Player move following an Opponent move in A (resp. B) is in B (resp. A)
- each Player move in A following an initial Opponent move in B is λ-justified by it
- the λ-pointers in $\mathtt{s} \upharpoonright_A$ and $\mathtt{s} \upharpoonright_B$ are the same

We denote by $\overline{\mathtt{s}}$ the unique zig-zag play on $B \to A$ such that $\overline{\mathtt{s}} \upharpoonright_A = \mathtt{s} \upharpoonright_A$ and $\overline{\mathtt{s}} \upharpoonright_B = \mathtt{s} \upharpoonright_B$.

In order to reuse results from [Lau05a], we define the function GR from arenas to ground arenas which erases the labels of its argument. If $\mathtt{s}$ is a play on A, we define $\mathrm{GR}(\mathtt{s})$ as the play on $\mathrm{GR}(A)$ obtained by erasing the instantiations and the μ-pointers. We extend GR to sets of plays by applying it point-wise. The main properties of GR are given in Appendix B.

Lemma 6 (Zig-zag lemma)
If (σ, τ) defines an isomorphism between A and B in $\mathcal{G}_1$, then they contain only zig-zag plays and $\tau = \overline{\sigma}$.

Proof: Since $(\mathrm{GR}(\sigma), \mathrm{GR}(\tau))$ is an isomorphism between $\mathrm{GR}(A)$ and $\mathrm{GR}(B)$ (by Lemma 11 page 48), we already know that plays in $\mathrm{GR}(\sigma)$ and $\mathrm{GR}(\tau)$ satisfy all the conditions of zig-zag plays but maybe μ-rigidity (see [Lau05a, Proof of Theorem 9]). We also know that $\mathrm{GR}(\sigma)$ and $\mathrm{GR}(\tau)$ are total thus σ and τ are total, by Lemma 19 page 52.

We prove by induction on k that if $\mathtt{s}$ is a play of length k in σ then $\mathtt{s}$ is zig-zag, $\overline{\mathtt{s}}$ is in τ and there is an interaction sequence $\mathtt{u}$ on A, B and A such that $u \upharpoonright_{A \to B} = \mathtt{s}$ and $u \upharpoonright_{B \to A} = \overline{\mathtt{s}}$. It is immediate for $k = 0$ since $\mathtt{s} = \varepsilon$ and thus $\mathtt{u} = \varepsilon$. Let $\mathtt{smn}$ be a play in σ of length $k + 2$, by induction hypothesis, we have $\overline{\mathtt{s}} \in \tau$ and an interaction sequence $\mathtt{u}$. Assume $\mathtt{m}$ is in A (the case $\mathtt{m}$ in B is symmetric) with a singleton atomic label (the case of an empty atomic label is simpler). We already know that $\mathtt{n}$ is in B. We define $\mathtt{u}' = \mathtt{um_1nm'}$ where $\mathtt{m}_1$ is a copy of the move $\mathtt{m}$ in the first A and $\mathtt{m}'$ is such that $\mathtt{u}' \upharpoonright_{B \to A} \in \tau$ (this move exists by totality of τ). We have $\mathtt{u}' \upharpoonright_{A \to B} = \mathtt{smn} \in \sigma$ thus $\mathtt{u}' \upharpoonright_{A \to A} \in \mathtt{id}_A$. This means that in $\mathtt{u}' \upharpoonright_{A \to A}$ the last move $\mathtt{m}'$ has its μ-pointer going to the previous move $\mathtt{m}_1$ and its instantiation is the same as the instantiation of $\mathtt{m}_1$. The only way to have these properties is that $\mathtt{m}'$ μ-points to $\mathtt{n}$ and $\mathtt{n}$ μ-points to $\mathtt{m}_1$ in $\mathtt{u}'$, and also that if $\vec{x}$ is the $\mathcal{O}$-instantiation of $\mathtt{m}_1$, $\vec{y}$ is the $\mathcal{P}$-instantiation of $\mathtt{n}$, $\vec{z}$ is the $\mathcal{O}$-instantiation of $\mathtt{n}$ and $\vec{w}$ is the $\mathcal{P}$-instantiation of $\mathtt{m}'$ in $\mathtt{u}'$ then $\vec{x} = \vec{y}$ and $\vec{z} = \vec{w}$. This proves $\mathtt{smn}$ to be a zig-zag play. Moreover $\mathtt{u}' \upharpoonright_{A \to A} \in \mathtt{id}_A$ also entails that $\mathtt{m}'$ and $\mathtt{m}_1$ are "the same move with the same pointers" thus $\mathtt{u}' \upharpoonright_{B \to A} = \overline{\mathtt{smn}} \in \tau$. □

Theorem 4 (Game isomorphisms)
Two arenas are isomorphic in $\mathcal{G}_1$ if and only if they are arena isomorphic (Definition 14 page 23).

PROOF: We start with the first direction. By [Lau05a, Theorem 9], there exists an arena isomorphism f between $\mathrm{GR}(A)$ and $\mathrm{GR}(B)$. Moreover if $\mathtt{smn} \in \sigma$ (with $\mathtt{m} \in A$) then the node corresponding to $\mathtt{n}$ in $\mathrm{GR}(B)$ (thus in B) is the image by f of the node corresponding to $\mathtt{n}$ in $\mathrm{GR}(A)$ (thus in A) and there exists such a play for any move $\mathtt{m}$ of A.

We prove by induction on the depth of the node $\mathtt{m}$ in A that f maps the labels of $\mathtt{m}$ to the labels of $f(\mathtt{m})$ turning it into an arena isomorphism between A and B. Let $\mathtt{smn}$ be the view in σ dealing with the node $\mathtt{m}$, it is μ-rigid (Lemma 6), thus if $X\vec{t}$ is the atomic label of $\mathtt{m}$ and $X'\vec{t'}$ is the atomic label of $\mathtt{n}$, we have $X = X'$ and $\vec{t}\theta_{\mathtt{m}} = \vec{t'}\theta_{\mathtt{n}}$. Moreover $\mathtt{m}$ and $\mathtt{n}$ have the same instantiation (thus first-order labels $[x_1, \dots, x_k]$ and $[y_1, \dots, y_k]$ of the same length in their respective arenas). If $\mathtt{m}$ is a root of A, its instantiation is $[o_0, \dots, o_{k-1}]$ and $\theta_{\mathtt{m}} = \{x_1 \mapsto o_0, \dots, x_k \mapsto o_{k-1}\}$. $\mathtt{n}$ is also a root and it has instantiation $[o_0, \dots, o_{k-1}]$ thus $\theta_{\mathtt{n}} = \{y_1 \mapsto o_0, \dots, y_k \mapsto o_{k-1}\}$. From this we can deduce $\vec{t'} = \vec{t}[{}^{y_1}/_{x_1}, \dots, {}^{y_k}/_{x_k}]$ showing that f respects the atomic label of $\mathtt{m}$ and $\mathtt{n}$. If $\mathtt{m}$ is not a root in A neither is $\mathtt{n}$, and we assume $\mathtt{m}$ is an Opponent move (otherwise we work with f^{-1}) so is $\mathtt{n}$. Let $\mathtt{m}_0$ be the justifier of $\mathtt{m}$ in $\mathtt{s}$ (and $\mathtt{n}_0$ be the justifier of $\mathtt{n}$), $\mathtt{m}$ and $\mathtt{n}$ have instantiation $[o_i, \dots, o_{i+k-1}]$ in $\mathtt{smn}$, and $\theta_{\mathtt{m}} = \theta_{\mathtt{m}_0} \cup \{x_1 \mapsto o_i, \dots, x_k \mapsto o_{i+k-1}\}$ and $\theta_{\mathtt{n}} = \theta_{\mathtt{n}_0} \cup \{y_1 \mapsto o_i, \dots, y_k \mapsto o_{i+k-1}\}$. From this we obtain that the mapping of first-order labels induced by f maps $\vec{t}$ to $\vec{t'}$.

The second direction is given by Lemma 3 page 24. □

All these results can easily be extended to $\mathcal{G}$ but are not required here. See Section 6 for possible applications of this extension.

Corollary 4.1 (Type isomorphisms)
Table 3 exactly characterizes the type isomorphisms of the Church style first-order $\lambda\mu$-calculus.

PROOF: Let A and B be two isomorphic types. They are both isomorphic to canonical forms A_0 and B_0 (see Section 1.3). By soundness of the game model, the arenas $[\![A_0]\!]$ and $[\![B_0]\!]$ interpreting A_0 and B_0 are isomorphic in $\mathcal{G}_1$ thus are arena isomorphic (Theorem 4). We prove by induction on the common number of nodes of $[\![A_0]\!]$ and $[\![B_0]\!]$ that A_0 and B_0 are equal up to the equations of Table 3 (so that A and B are also equal up to these equations). Let f be the arena isomorphism between $[\![A_0]\!]$ and $[\![B_0]\!]$, f defines a bijection between the roots of the two arenas. If there is more than one root, A_0 and B_0 are conjunctions and each tree corresponds to one component. We apply the induction hypothesis to the pairs of trees whose roots are related through f. Since Table 3 contains the associativity and commutativity of conjunction, one obtains that A_0 and B_0 are equal up to the equations. If $[\![A_0]\!]$ and $[\![B_0]\!]$ are trees, A_0 and B_0 are $\to$-canonical forms. Let $[x_1, \dots, x_k]$ and $[y_1, \dots, y_k]$ be the first-order labels of their roots (they have the same length), their atomic label are either empty or $X\vec{t}$ and $X\vec{t}[{}^{y_1}/_{x_1}, \dots, {}^{y_k}/_{x_k}]$. Let $\mathcal{F}_A$ and $\mathcal{F}_B$ be the forests under these roots. By induction hypothesis, $\mathcal{F}_A$ and $\mathcal{F}_B[{}^{x_1}/_{y_1}, \dots, {}^{x_k}/_{y_k}]$ correspond to canonical forms $\bigwedge_{1\le i\le n} A_i$ and $\bigwedge_{1\le j\le n} B_j$ equal up to the equations. We can conclude since $\forall\vec{x}(A_1 \to \dots \to A_n \to R)$ and $\forall\vec{y}(B_1[{}^{y_1}/_{x_1}, \dots, {}^{y_k}/_{x_k}] \to \dots \to B_n[{}^{y_1}/_{x_1}, \dots, {}^{y_k}/_{x_k}] \to S)$ are equal up to the equations (with $R = S = \bot$ if the roots of $[\![A_0]\!]$ and $[\![B_0]\!]$ have an empty atomic label and with $R = X\vec{t}$ and $S = X\vec{t}[{}^{y_1}/_{x_1}, \dots, {}^{y_k}/_{x_k}]$ otherwise).

Finally, by Appendix A.2, if A and B are equal up to the equations of Table 3, A and B are isomorphic. □

5 Krivine's realizability

Realizability is used in the study of the computational properties of proofs in intuitionistic logic. The setting introduced by J.-L. Krivine allows for the extension to classical logic and even set theory [Kri01]. In particular, he showed more recently that computational interpretations of classical proofs can be given through a notion of games by means of his realizability interpretation. We are going to explain the close relation between his games and game semantics as presented in this paper.

5.1 A quick introduction

In the setting of Krivine's realizability, terms are studied through their computational behaviour via a notion of execution very similar to what happens in Krivine's abstract machine [Kri07].

A state is a pair of a Curry style $\lambda\mu$-term M (see Section 3.4) and of a stack π:

$$\pi ::= \varepsilon \mid \alpha \mid M.\pi$$

Such a state is written $M \bowtie \pi$ and the execution is given by a relation $\succ$ between states:

$$\begin{aligned}
\lambda a.M \bowtie N.\pi &\succ M[{}^{N}/_{a}] \bowtie \pi \\
(M)N \bowtie \pi &\succ M \bowtie N.\pi \\
\mu\alpha.M \bowtie \pi &\succ M[{}^{\pi}/_{\alpha}] \bowtie \varepsilon \\
[\alpha]M \bowtie \varepsilon &\succ M \bowtie \alpha
\end{aligned}$$

where $M[{}^{M_1 \dots M_k.\varepsilon}/_{\alpha}] = M[{}^{(N)M_1 \dots M_k}/_{[\alpha]N}]$ and $M[{}^{M_1 \dots M_k.\beta}/_{\alpha}] = M[{}^{[\beta](N)M_1 \dots M_k}/_{[\alpha]N}]$.

Through the following embedding of states into $\lambda\mu$-terms:

$$\begin{aligned}
M \bowtie M_1 \dots M_k.\varepsilon &\mapsto (M)M_1 \dots M_k \\
M \bowtie M_1 \dots M_k.\alpha &\mapsto [\alpha](M)M_1 \dots M_k
\end{aligned}$$

these execution rules are simulated by the $\beta\mu\rho$ reduction.

5.2 $\mathcal{UVA}$ provability game

In this section we give a straightforward adaption of results in [Kri06, pages 77–86] to our setting. We introduce the notion of $\mathcal{UVA}$ *game*.

A *position* is a triple $(\mathcal{U}, \mathcal{V}, \mathcal{A})$ where $\mathcal{U}$ and $\mathcal{V}$ are sets of formulas in $\rightarrow$-canonical form and $\mathcal{A}$ is a set of non-constant atomic formulas.

An Opponent move consists in choosing a formula $\forall\vec{x}(A_1 \rightarrow \dots \rightarrow A_n \rightarrow R)$ in $\mathcal{V}$ and first-order terms $\vec{t}$. We then go to the position $(\mathcal{U} \cup \{A_1[{}^{\vec{t}}/_{\vec{x}}], \dots, A_n[{}^{\vec{t}}/_{\vec{x}}]\}, \mathcal{V}, \mathcal{A} \cup \{R[{}^{\vec{t}}/_{\vec{x}}]\})$ if $R \neq \bot$ and $(\mathcal{U} \cup \{A_1[{}^{\vec{t}}/_{\vec{x}}], \dots, A_n[{}^{\vec{t}}/_{\vec{x}}]\}, \mathcal{V}, \mathcal{A})$ otherwise.

A Player move consists in choosing a formula $\forall\vec{x}(A_1 \rightarrow \dots \rightarrow A_n \rightarrow R)$ in $\mathcal{U}$ and first-order terms $\vec{t}$ such that, if $R \neq \bot$, $R[{}^{\vec{t}}/_{\vec{x}}]$ is in $\mathcal{A}$. We then go to the position $(\mathcal{U}, \{A_1[{}^{\vec{t}}/_{\vec{x}}], \dots, A_n[{}^{\vec{t}}/_{\vec{x}}]\}, \mathcal{A})$.

An *initial position* is a position with $\mathcal{U} = \mathcal{A} = \emptyset$ (in particular the initial position associated with a formula A is $(\emptyset, \{A_1, \dots, A_n\}, \emptyset)$ if the canonical form of A is $\bigwedge_{1 \leq i \leq n} A_i$). A *final position* is a position with $\mathcal{V} = \emptyset$ (it corresponds to positions where Opponent cannot play).

A *play* is a (possibly empty) sequence of moves in which players alternate and which starts from an initial position by an Opponent move. If an initial position is given, Player is said to have a *winning strategy* if he is able to choose his moves in such a way that he is always able to play and to eventually reach a final position (meaning that Opponent wins in infinite plays).

The main property of $\mathcal{UVA}$ games is the following theorem of Krivine:

Theorem 5
A is a provable formula if and only if there exists a winning strategy for Player in the associated $\mathcal{UVA}$ game.

More precisely, if M is a Curry style $\lambda\mu$-term of type A, M *implements* a winning strategy for Player in the $\mathcal{UVA}$ game associated with A:

With each formula A we associate a λ-variable a_A and with each non-constant atomic formula R we associate a μ-variable α_R. If M has type $A = \forall\vec{x}(A_1 \to \cdots \to A_n \to R)$, we choose first-order terms $\vec{t}$ and we look at the execution of $M \bowtie a_{A_1[\vec{t}/\vec{x}]} \dots a_{A_n[\vec{t}/\vec{x}]}.\alpha_{R[\vec{t}/\vec{x}]}$ (or with ε instead of $\alpha_{R[\vec{t}/\vec{x}]}$ if $R = \bot$). It is shown that execution will stop in a state $a_{A_i[\vec{t}/\vec{x}]} \bowtie \pi$ with $\pi = M_1 \dots M_k.\alpha$ if $A_i[\vec{t}/\vec{x}] = \forall\vec{y}(B_1 \to \cdots \to B_k \to S)$ (or with ε instead of α if $S = \bot$). There exists a choice of first-order terms $\vec{u}$ such that for any choice of $1 \leq j \leq k$ and any choice of the first-order terms $\vec{v}$ with $B_j[\vec{u}/\vec{y}] = \forall\vec{z}(C_1 \to \cdots \to C_p \to T)$, the execution of $M_j \bowtie a_{C_1[\vec{v}/\vec{z}]} \dots a_{C_p[\vec{v}/\vec{z}]}.\alpha_{T[\vec{v}/\vec{z}]}$ will stop in a state $a_D \bowtie \pi'$ and so on... Whatever choices (for j and $\vec{v}$) we make between the steps of this sequence of runs, it will stop in a state $a \bowtie \alpha$ (or $a \bowtie \varepsilon$).

If we interpret execution steps (with the choices of $\vec{u}$) as Player moves and index choices (with the choices of $\vec{v}$) as Opponent moves, this shows that a Curry style $\lambda\mu$-term of type A induces a winning strategy for Player in the $\mathcal{UVA}$ game associated with A: each sequence of runs is a play in this game.

5.3 Relation with game semantics

We are now going to extend the previous correspondence between execution of terms and the $\mathcal{UVA}$ provability games to a correspondence between execution and our game model by just a slight modification on the execution sequence. This correspondence is in fact the starting point of the present work on game semantics.

The modifications we have to do concern the use of Church style $\lambda\mu$-terms and the possibility of recovering pointers in games.

The explicit use of first-order terms in the construction of $\lambda\mu$-terms corresponds to the following execution rules:

$$\Lambda x.M \bowtie t.\pi \succ M[t/x] \bowtie \pi$$
$$M\{t\} \bowtie \pi \succ M \bowtie t.\pi$$

where stacks are extended with the construction $t.\pi$.

With each formula A we associate a denumerable set of λ-variables $(a^l_A)_{l\in\mathbb{N}}$ and with each non-constant atomic formula R we associate a denumerable set of μ-variables $(\alpha^l_R)_{l\in\mathbb{N}}$. Starting with a Church style $\lambda\mu$-term M of type $A = \forall\vec{x}(A_1 \to \cdots \to A_n \to R)$ with $\vec{x}$ of length n', we proceed as follows:

- We start with a state: $M \bowtie t_1 \dots t_{n'}.a^1_{A_1[\vec{t}/\vec{x}]} \dots a^1_{A_n[\vec{t}/\vec{x}]}.\alpha^1_{R[\vec{t}/\vec{x}]}$.

- When execution stops (for the ith time) in a state: $a^m_B \bowtie u_1 \dots u_{k'}.M_1 \dots M_k.\alpha$ with $B = \forall \vec{y}(B_1 \to \dots \to B_k \to S)$ (with ε instead of α if $S = \bot$), we choose first-order terms $\vec{v}$ and an index $1 \leq j \leq k$ and we start a new execution $M_j \bowtie v_1 \dots v_{p'}.a^{i+1}_{C_1[\vec{v}/\vec{z}]} \dots a^{i+1}_{C_p[\vec{v}/\vec{z}]}.\alpha^{i+1}_{T[\vec{v}/\vec{z}]}$ where $B_j[\vec{u}/\vec{y}] = \forall \vec{z}(C_1 \to \dots \to C_p \to T)$ (with ε instead of $\alpha^{i+1}_{T[\vec{v}/\vec{z}]}$ if $T = \bot$).

An important point in the choice of the a^i_As and α^i_Rs is that they are always fresh (*i.e.* not used yet).

Such an execution sequence can be interpreted as a play (as defined in Section 2.2) where Opponent moves are given by the choices ($v_1 \dots v_{p'}.a^{i+1}_{C_1[\vec{v}/\vec{z}]} \dots a^{i+1}_{C_p[\vec{v}/\vec{z}]}.\alpha^{i+1}_{T[\vec{v}/\vec{z}]}$ for example) and Player moves are given by the results of executions ($a^m_B \bowtie u_1 \dots u_{k'}.M_1 \dots M_k.\alpha$ for example). More precisely we can rebuild a view with its pointers and instantiations from such a sequence. We consider only the case where first-order terms introduced in Opponent moves are fresh variables (by innocence, this is enough to recover the corresponding strategy).

Since we focus here on $\to$-canonical forms, the corresponding arenas are trees, and the starting state $M \bowtie x_1 \dots x_{n'}.a^1_{A_1[\vec{t}/\vec{x}]} \dots a^1_{A_n[\vec{t}/\vec{x}]}.\alpha^1_{R[\vec{t}/\vec{x}]}$ is interpreted as Opponent playing the root of this arena with instantiation $\vec{x}$ ($a^1_{A_1[\vec{t}/\vec{x}]} \dots a^1_{A_n[\vec{t}/\vec{x}]}$ and $\alpha^1_{R[\vec{t}/\vec{x}]}$ are given as possible targets for future pointers). From that point, a result of execution $a \bowtie u_1 \dots u_{k'}.M_1 \dots M_k.\alpha$ is interpreted as a Player move: its λ-pointer is going to the Opponent move $\mathtt{m}$ where a has been introduced (and let i be its position in the state corresponding to $\mathtt{m}$), the move $\mathtt{n}$ itself is the ith son of $\mathtt{m}$ in the arena, the instantiation is $\vec{u}$ and the μ-pointer is going to the move where α has been introduced (if we have ε instead of α, there is no μ-pointer). The next starting point of execution $M_j \bowtie y_1 \dots y_{p'}.\pi$ is interpreted as Opponent playing the jth son of $\mathtt{n}$ in the arena with a λ-pointer going to the previous move $\mathtt{n}$ (as required for a view) and instantiations $\vec{y}$.

Proposition 9
We obtain this way a view on the associated arena if Opponent plays its instantiations according to $(o_i)_{i\in\mathbb{N}}$.

Proof: We build the view $\mathtt{s}$ by induction on the length of the execution using the fact that types are appropriately preserved along the execution (as shown through the embedding into the $\lambda\mu$-calculus given in Section 5.1):

- The starting state of the shape $M \bowtie o_0 \dots o_{n'-1}.a^1_{A_1[\vec{o}/\vec{x}]} \dots a^1_{A_n[\vec{o}/\vec{x}]}.\alpha^1_{R[\vec{o}/\vec{x}]}$ is interpreted as a valid Opponent initial move $\mathtt{m}$ on the arena associated with A. $\mathtt{s}$ is just this move. Moreover, this defines a substitution $\theta_{\mathtt{m}} = \{x_1 \mapsto o_0, \dots, x_{n'} \mapsto o_{n'-1}\}$ as in the definition of plays.
- If we arrive at a state $a^q_B \bowtie u_1 \dots u_{k'}.M_1 \dots M_k.\alpha^{q'}_S$, let $\mathtt{m}$ be the move of $\mathtt{s}$ corresponding to the state where a^q_B has been introduced and let i be the position of a^q_B in the sequence $\vec{a}$ in this state. Let $\mathtt{n}$ be the node of the arena A which is the ith son of the node corresponding to $\mathtt{m}$ in A, we extend $\mathtt{s}$ with $\mathtt{n}$ with a λ-pointer going to $\mathtt{m}$ (this is a correct λ-pointer). By preservation of typing in the execution, B is of the shape $\forall y_1 \dots \forall y_{k'}(B_1 \to \dots \to B_k \to S')$. This means that the node $\mathtt{n}$ in A has a first-order label of length k' and has k sons. As a consequence, $u_1 \dots u_{k'}$ is an appropriate instantiation for the move $\mathtt{n}$, and we can define $\theta_{\mathtt{n}} = \theta_{\mathtt{m}} \cup \{y_1 \mapsto u_1, \dots, y_{k'} \mapsto u_{k'}\}$. The atomic label of $\mathtt{n}$ in A is S_0 with $S_0\theta_{\mathtt{n}} = S'[{}^{u_1}/_{y_1}, \dots, {}^{u_{k'}}/_{y_{k'}}] = S = S_1\theta_{\mathtt{m}'}$ where S_1 is the atomic

label of the move $\mathtt{m}'$ corresponding to the state where $\alpha_S^{q'}$ has been introduced. It is then valid to put a μ-pointer from $\mathtt{n}$ to $\mathtt{m}'$ and $\mathtt{sn}$ is a correct view in A.

- Almost the same with a state $a_B^q \bowtie u_1 \dots u_{k'}.M_1 \dots M_k.\varepsilon$ but without μ-pointer.
- In order to run the execution again, we build a new state:

$$M_j \bowtie o_i \dots o_{i+p'-1}.a^l_{C_1[\vec{v}/\vec{z}]} \dots a^l_{C_p[\vec{v}/\vec{z}]}.\alpha^l_{T[\vec{v}/\vec{z}]}$$

(which respects the types). This corresponds to playing the jth son $\mathtt{m}$ of the node corresponding to the last move of $\mathtt{s}$ with a λ-pointer going to it and an instantiation $[o_i \dots o_{i+p'-1}]$ where $\mathtt{m}$ has a first-order label of length p' in A. This proves $\mathtt{sm}$ to be correct view on A. □

6 Extensions and additional comments

We have chosen to work, on the syntactic side, with a natural deduction system (the $\lambda\mu$-calculus). Another possibility would have been to deal with a sequent calculus system (as in [Lau05b] for example). It would not make very important differences. The notion of proofs in canonical form in the sequent calculus is given by cut-free proofs with expanded axioms (introducing only non-constant atomic formulas). In this context, μ-pointers would precisely correspond to these atomic axioms connecting together two dual occurrences of an atomic formula. In [Lau05b], an encoding was required and μ-pointers were somehow the missing data.

Another specific choice was to deal with a $\lambda\mu$-calculus without disjunction or negation. Negation is easily definable through $\neg A = A \to \bot$ and disjunction can be introduced as in Selinger's calculus [Sel01] with two new term constructs: $\mu(\alpha, \beta).M$ and $[\alpha, \beta]M$. The associated typing rules are:

$$\frac{\Gamma, a : A \vdash M : \bot \mid \Delta}{\Gamma \vdash \ell a.M : \neg A \mid \Delta} \qquad \frac{\Gamma \vdash M : \neg A \mid \Delta \qquad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash M \bullet N : \bot \mid \Delta}$$

$$\frac{\Gamma \vdash M : A \vee B \mid \Delta, \alpha : A, \beta : B}{\Gamma \vdash [\alpha, \beta]M : \bot \mid \Delta, \alpha : A, \beta : B} \qquad \frac{\Gamma \vdash M : \bot \mid \Delta, \alpha : A, \beta : B}{\Gamma \vdash \mu(\alpha, \beta).M : A \vee B \mid \Delta}$$

All the required material for interpreting these extensions is already given in the game category $\mathcal{G}$ we have described. In particular, the interpretation of disjunction makes $\mathcal{G}_1$ not big enough and requires us to work with the full control category $\mathcal{G}$ (as done in [Sel01]).

The explicit treatment of these extensions would require some additional work on the syntactic side, in particular for the notion of canonical form. However no surprise would come from this and the results presented in this paper would extend without any particular problem. We can mention

that, concerning type isomorphisms, the following equations would be derived:

$$\begin{aligned}
A \vee (B \vee C) &= (A \vee B) \vee C \\
A \vee B &= B \vee A \\
A \vee \bot &= A \\
A \vee (B \wedge C) &= (A \vee B) \wedge (A \vee C) \\
A \vee \top &= \top \\
A \vee \forall x B &= \forall x (A \vee B) && x \notin A \\
\neg(A \wedge B) &= \neg A \vee \neg B \\
\neg\top &= \bot \\
A \to B &= \neg A \vee B
\end{aligned}$$

Enumerated data types such as `Bool` or `Nat` are usually interpreted in game semantics by `QA`-arenas with one root (a question) which has as many sons (which are answers) as elements of the data type (possibly an infinity). The traditional approach of game semantics is to build everything from these enumerated data types without any use of atomic formulas. We have shown in Section 3.3 how the label-rigidity constraint makes answers exactly as expressive as μ-pointers. It would be interesting to relax this constraint to deal with systems with both enumerated data types and atomic formulas. Some work has already been done in [HL06] to understand the expressive power of various possible restrictions on the use of answers (in particular some are weaker than label-rigidity). This would help to understand more precisely the possible applications of μ-pointers to the semantics of programming languages in relation with extensions of PCF with control operators (starting from [Lai97] and [Lai01]).

The question of introducing enumerated data types in Krivine's realizability as been considered in [Bef04]. It would be nice to also extend the correspondence between games and realizability to data types.

Let us now look at the most natural logical extensions of this work. Concerning first-order logic, it would be important to introduce the *equality predicate* and to be able to deal with given equational axioms and not only with the free first-order language. Once again this could be developed in relation with what happens in Krivine's realizability. What makes such an extension difficult is the requirement of some dynamism in arenas: an arena has to dynamically evolve during a play according to the moves of the players. Concerning equality, it is just a matter of a node being able to disappear (when an equational atomic label $t = u$ of a node becomes true).

This dynamics induced by moves on arenas is at the core of the game interpretations of (propositional) second-order logic (see [Hug97, dL07] for example). Being able to mix our work with second-order interpretations is the main direction for future work leading to an equivalence complete game model of full second-order logic. An immediate consequence would be the associated characterization of type isomorphisms. We conjecture the corresponding equational theory to be the union of ours with the second-order one given in [dL08], together with $\forall x \forall X A = \forall X \forall x A$.

Acknowledgements

I would like to thank Joachim de Lataillade, Russ Harmer, Martin Hyland and Jean-Louis Krivine for important discussions during the development of this work. I also thank the anonymous referees for their useful suggestions.

$$
\begin{array}{rcll}
(\lambda a.M)N & \to_\beta & M[^N/_a] & \\
\lambda a.(M)a & \to_\eta & M & a \notin M \\
(\mu\alpha.M)N & \to_\mu & \mu\alpha.M[^{[\alpha](L)N}/_{[\alpha]L}] & \\
[\beta]\mu\alpha.M & \to_\rho & M[^\beta/_\alpha] & \\
\mu\alpha[\alpha]M & \to_\theta & M & \alpha \notin M \\
\mu\alpha.M & \to_\nu & \lambda a.\mu\alpha.M[^{[\alpha](L)a}/_{[\alpha]L}] & a \notin M
\end{array}
$$

Table 6: Reduction rules for the $\lambda\mu$-calculus

A Some properties of the $\lambda\mu$-calculus

A.1 Canonical normal forms

A $\lambda\mu$-term M is *simple* if it is in the grammar:

$$M \quad ::= \quad a \mid \lambda a.M \mid (M)M \mid \mu\alpha[\beta]M \mid \Lambda x.M \mid M\{t\}$$

We first remark that any typed $\lambda\mu$-term without pairs, projections or $\star$ is equal to a typed simple $\lambda\mu$-term up to ρ: we choose a particular fresh variable ξ of type $\bot$, we transform any $[\alpha]M$ into $\mu\xi[\alpha]M$ (by $[\alpha]M =_\rho [\xi]\mu\xi[\alpha]M =_\rho \mu\xi[\alpha]M$) and any $\mu\alpha.M$ into $\mu\alpha[\xi]M$ (since M must be of type $\bot$). To see that the result is typed, we add $\xi : \bot$ to the right-hand side typing context.

Our goal is to show the existence of a canonical normal form for any typed simple $\lambda\mu$-term. We are going to adapt results coming from [Py98, Chapter 5]. We recall the reductions defined in that work in Table 6. Notice that they are all validated by the $\beta\eta\mu\rho\theta$ equational theory. The only interesting case being the ν-reduction:

$$\mu\alpha.M =_\eta \lambda a.(\mu\alpha.M)a =_\mu \lambda a.\mu\alpha.M[^{[\alpha](L)a}/_{[\alpha]L}]$$

However, in a typed setting, the ν-reduction cannot always be applied (it requires $\mu\alpha.M$ to be of arrow type), so that we cannot directly apply the results of [Py98].

A simply typed simple $\lambda\mu$-term M of type $A = A_1 \to \cdots \to A_n \to X$ is in *canonical normal form* if $M = \lambda a_1 \ldots \lambda a_n \mu\alpha[\beta](b)M_1 \ldots M_k$ with a_i of type A_i, α of type X, β of type Y, b of type B and M_j canonical normal form of type B_j (with $B = B_1 \to \cdots \to B_k \to Y$).

Lemma 7 (Canonical normal form (simple types))
Let M be a simply typed μ-closed simple $\lambda\mu$-term, there exists a canonical normal form which is $\beta\eta\mu\rho\theta$ equivalent to M. More precisely turning M into a canonical normal form only requires $\beta\mu\rho\nu$-reductions and $\eta\theta$-expansions.

Proof: The $\beta\mu\rho$-reduction is normalizing [Par97], so that we can concentrate on $\beta\mu\rho$-normal forms.

We first prove, by induction on the size of $A = A_1 \to \cdots \to A_n \to X$, that if a is a λ-variable of type A then it $\eta\theta$-expands into a canonical normal form: by induction hypothesis applied to the λ-variables a_i of type A_i, we obtain the canonical normal forms M_i, and we have:

$$a \leftarrow^*_{\eta\theta} \lambda a_1 \ldots \lambda a_n \mu\alpha[\alpha](a)a_1 \ldots a_n \leftarrow^*_{\eta\theta} \lambda a_1 \ldots \lambda a_n \mu\alpha[\alpha](a)M_1 \ldots M_n$$

By induction on the size of N, we prove that any $\beta\mu\rho$-normal form N of type A is $\beta\eta\mu\rho\theta$ equivalent to a *pre-canonical form*, that is a $\lambda\mu$-term of the shape $\overline{\lambda\mu}(b)M_1 \dots M_k$ where $\overline{\lambda\mu}$ is a sequence of λs and $\mu[]$s ending with a $\mu[]$ and containing no pair of consecutive $\mu[]$s, M_j is a pre-canonical form of type B_j, and moreover b is of type $B = B_1 \to \dots \to B_k \to Y$ (that is each variable has as many arguments as possible).

If N is a $\beta\mu\rho$-normal form (note that it immediately ensures the elimination of consecutive $\mu[]$s), under a bunch of λs and $\mu[]$s, we find a term of the shape $N_0 = (b)M_1 \dots M_p$ with b of type $B = B_1 \to \dots \to B_k \to Y$ and $p \leq k$. If $p = k$, either N_0 is under a $\mu[]$ and it is in pre-canonical normal form or N_0 is under a λ and we apply a θ-expansion to N_0. If $p < k$, we replace N_0 by N_1 (which is equivalent to it) obtained from $\lambda b_{p+1} \dots \lambda b_k \mu\xi[\xi](b)M_1 \dots M_p b_{p+1} \dots b_k$ (ξ fresh) by replacing each b_j by its canonical normal form (using the first result above). Finally we replace each M_i by its pre-canonical form (using the induction hypothesis).

Notice that, in a pre-canonical form M, if α is of type $A \to B$ and $[\alpha]N$ is a sub-term of M then N starts with a λ: otherwise it could only be a μ (but it would not be ρ-normal), or of the shape $(c)\vec{L}$ (but such a sub-term of a pre-canonical form under a $[]$ is always of atomic type). This remark allows us to define the following modified ν-reduction on pre-canonical forms:

$$\mu\alpha.M \to_\nu \lambda a.\mu\alpha.M[^{[\alpha]L}/_{[\alpha]\lambda a.L}]$$

if α is of arrow type.

We finally prove the lemma by showing that any pre-canonical form M is equivalent to a canonical normal form. We prove this by induction on the sum of the sizes of the types of the μ-variables in M. If all the types of the μ-variables of M are atomic, then M is in canonical normal form. Otherwise, let $\mu\alpha.N$ be a sub-term of M of arrow type (remember that the $\lambda\mu$-term is μ-closed), we apply a ν-reduction, this makes the induction size decrease but we are not sure to obtain directly a pre-canonical form again: we may have to apply some ρ-reductions to have a pre-canonical form, but these reductions terminate (the size of the $\lambda\mu$-term decreases) and do not make the induction size increase. □

In order to extend this result to the first-order case, we first define an embedding of simple $\lambda\mu$-terms typed in first-order logic (without $\wedge$ or $\top$) into the simply typed $\lambda\mu$-calculus. We consider an injective embedding of first-order function symbols, first-order variables and λ-variables into λ-variables (we still denote by f, x and a the translation of f, x and a), and an embedding of relation symbols into 0-ary relation symbols (the image of X will be denoted by X). Moreover we choose two particular atomic types O and F. The translation is given in Table 7.

We extend the modified ν-reduction to first-order constructs:

$$\mu\alpha.M \to_\nu \Lambda x.\mu\alpha.M[^{[\alpha]L}/_{[\alpha]\Lambda x.L}]$$

if α is of type $\forall x A$ and all the occurrences of $[\alpha]$ in M are followed by Λx.

Lemma 8 (Properties)
The translation $(.)^\star$ *has the following properties:*

- *If* $\Gamma \vdash M : A \mid \Delta$ *then* $\Lambda, \Gamma^\star \vdash M^\star : A^\star \mid \Delta^\star$*, where* $\Gamma^\star$ *contains the translations of the typing declarations in* Γ *and typing declarations* $x : O$ *for (at least) the free first-order variables of*

$$
\begin{array}{rcl@{\qquad}rcl}
\bot^\star & = & F & a^\star & = & a \\
(X\vec{t})^\star & = & X & (\lambda a.M)^\star & = & \lambda a.M^\star \\
(A \to B)^\star & = & A^\star \to B^\star & ((M)N)^\star & = & (M^\star)N^\star \\
(\forall x A)^\star & = & O \to A^\star & (\mu\alpha[\beta]M)^\star & = & \mu\alpha[\beta]M^\star \\
 & & & (\Lambda x.M)^\star & = & \lambda x.M^\star \\
x^\star & = & x & (M\{t\})^\star & = & (M^\star)t^\star \\
(ft_1 \dots t_k)^\star & = & (f)t_1^\star \dots t_k^\star & & &
\end{array}
$$

Table 7: A translation from first-order typing to simple types

M, and Λ contains a typing declaration $f : O^k \to O$ for each function symbol f of arity k occurring in M.

- $(M[^t/_x])^\star = M^\star[^{t^\star}/_x]$
- *There is a one-to-one correspondence between the source and the target language for β-reduction, μ-reduction, ρ-reduction, ν-reduction, η-expansion and θ-expansion. This means that if $\to$ is one of these rewriting rules, we have both simulation (if $M \to N$ then $M^\star \to N^\star$) and co-simulation (if $M^\star \to C$, there exists N such that $C = N^\star$ and $M \to N$).*

PROOF: • We first show that $\Lambda, \Gamma^\star \vdash t^\star : O \mid \Delta^\star$ if $\Gamma^\star$ contains typing declarations $x : O$ for (at least) the free first-order variables of t and Λ contains typing declarations for the function symbols occurring in t:

$$\frac{}{\Lambda, \Gamma^\star, x : O \vdash x : O \mid \Delta^\star}$$

$$\frac{\dfrac{}{\Lambda, \Gamma^\star \vdash f : O^k \to O \mid \Delta^\star} \qquad \Lambda, \Gamma^\star \vdash t_1^\star : O \mid \Delta^\star}{\Lambda, \Gamma^\star \vdash (f)t_1^\star : O^{k-1} \to O \mid \Delta^\star}$$

$$\vdots$$

$$\frac{\Lambda, \Gamma^\star \vdash (f)t_1^\star \dots t_{k-1}^\star : O \to O \mid \Delta^\star \qquad\qquad \Lambda, \Gamma^\star \vdash t_k^\star : O \mid \Delta^\star}{\Lambda, \Gamma^\star \vdash (f)t_1^\star \dots t_k^\star : O \mid \Delta^\star}$$

The only two interesting cases of $\lambda\mu$-terms are for first-order constructs:

$$\frac{\Lambda, \Gamma^\star, x : O \vdash M^\star : A^\star \mid \Delta^\star}{\Lambda, \Gamma^\star \vdash \lambda x.M^\star : O \to A^\star \mid \Delta^\star}$$

$$\frac{\Lambda, \Gamma^\star \vdash M^\star : O \to A^\star \mid \Delta^\star \qquad \Lambda, \Gamma^\star \vdash t^\star : O \mid \Delta^\star}{\Lambda, \Gamma^\star \vdash (M^\star)t^\star : A^\star \mid \Delta^\star}$$

- We first check that $(u[^t/_x])^\star = u^\star[^{t^\star}/_x]$ for first-order terms. Then we work by induction

on M, with only one interesting case:

$$\begin{aligned}(M\{u\}[^t/_x])^\star &= (M[^t/_x]\{u[^t/_x]\})^\star\\ &= (M[^t/_x]^\star)u[^t/_x]^\star\\ &= (M^\star[^{t^\star}/_x])u^\star[^{t^\star}/_x]\\ &= (M^\star)u^\star[^{t^\star}/_x]\\ &= (M\{u\})^\star[^{t^\star}/_x]\end{aligned}$$

- We first prove the simulation results. For redexes without first-order construct, the result is immediate, for the other ones we have:
  - $(\Lambda x.M)\{t\} \to_\beta M[^t/_x]$ becomes $(\lambda x.M^\star)t^\star \to_\beta M^\star[^{t^\star}/_x]$ and we apply the previous result
  - $M \leftarrow_\eta \Lambda x.M\{x\}$ becomes $M^\star \leftarrow_\eta \lambda x.(M^\star)x$
  - $(\mu\alpha.M)\{t\} \to_\mu \mu\alpha.M[^{[\alpha]L\{t\}}/_{[\alpha]L}]$ becomes $(\mu\alpha.M^\star)t^\star \to_\mu \mu\alpha.M^\star[^{[\alpha](L)t^\star}/_{[\alpha]L}]$ and we conclude from the fact that $[\,]$s are not modified by the translation
  - $\mu\alpha.M \to_\nu \Lambda x.\mu\alpha.M[^{[\alpha]L}/_{[\alpha]\Lambda x.L}]$ becomes $\mu\alpha.M^\star \to_\nu \lambda x.\mu\alpha.M^\star[^{[\alpha]L}/_{[\alpha]\lambda x.L}]$ since each Λx under a $[\alpha]$ becomes a λx and $\forall x A$ becomes $O \to A^\star$.
- We now look at co-simulation for each rewriting rule:
  - If $M^\star$ contains a β-redex, either both λ and application are already present in M and the result is immediate, or they both come from the corresponding first-order constructs and we can apply the simulation result, or exactly one come from a first-order construct and this would be a typing error in M.
  - If $M^\star$ contains a μ-redex, either the application is already present in M and the result is immediate, or it comes from a first-order application and we can apply the simulation result.
  - If $M^\star$ contains a ρ-redex, so do M, and the result is immediate.
  - If $M^\star$ contains a ν-redex starting with $\mu\alpha$, so do M, and we can apply the same reduction in M (with a Λ if α is of type $O \to A$ and with a λ otherwise). Finally we conclude with the simulation result.
  - If $M^\star$ η-expands to $\lambda a.(M^\star)a$ then $N = \lambda a.(M)a$ and if $M^\star$ η-expands to $\lambda x.(M^\star)x$ then $N = \Lambda x.M\{x\}$. We conclude with the simulation result.
  - By the simulation result, the case of a θ-expansion is immediate. □

A simple $\lambda\mu$-term M of type $A = \forall x_1 \dots \forall x_p(A_1 \to \dots \to A_n \to R)$ is in *quasi canonical normal form* if $M = \Lambda x_1 \dots \Lambda x_p \lambda a_1 \dots \lambda a_n \mu\alpha[\beta](b\{t_1\} \dots \{t_q\})M_1 \dots M_k$ with a_i of type A_i, α of type R, β of type $S[^{t_1}/_{y_1}, \dots, ^{t_q}/_{y_q}]$, b of type B and M_j canonical normal form of type $B_j[^{t_1}/_{y_1}, \dots, ^{t_q}/_{y_q}]$ (with $B = \forall y_1 \dots \forall y_q(B_1 \to \dots \to B_k \to S)$). A canonical form is obtained from a quasi canonical normal form by removing the μs and $[\,]$s acting on μ-variables of type $\bot$.

Proposition 10 (Canonical normal form)
Let M be a typed μ-closed simple $\lambda\mu$-term, there exists a canonical normal form which is $\beta\eta\mu\rho\theta$ equivalent to M.

PROOF: We first show how to obtain quasi canonical normal forms. We translate M into $M^\star$ and we apply Lemma 7 to get a simply typed canonical normal form C. By Lemma 8 (since going from $M^\star$ to C requires only $\beta\mu\rho\nu$-reduction and $\eta\theta$-expansion) there exists a $\lambda\mu$-term N equivalent to M and such that $N^\star = C$. Going from C to N transforms some λs into Λs and some applications into first-order applications, but using the type we can see that we do not have choices: the Λs are all arriving before the λs (for each bunch of λs of C) and the same for applications. As a consequence N is a quasi canonical normal form.

Finally, if the quasi canonical normal form M contains some $\mu\alpha[\beta]$ with α of type $\bot$ we apply $\mu\alpha[\beta]N =_\rho [\alpha]\mu\alpha[\beta]N =_\rho [\beta]N$, and if M contains some $[\alpha]N$ with α of type $\bot$, we erase the $[\alpha]$ by ρ. □

A.2 Isomorphisms

The notion of isomorphism is very standard in algebra and in category theory. There is a natural corresponding notion in extensions of the λ-calculus [DC76]. We consider such an extension of the λ-calculus endowed with the equational theory on terms generated by the reduction rules (containing the $\beta\eta$ equality). The term M is an isomorphism if there exists a term N such that $\lambda x.(M)(N)x = \lambda x.x$ and $\lambda y.(N)(M)y = \lambda y.y$ (we say that M and N give an isomorphism pair).

In a typed setting (*i.e.* if the calculus comes with a type system extending the simply typed λ-calculus), we can consider isomorphisms between types [DC95]. The types A and B are isomorphic if there exist two terms M and N such that $\vdash M : A \to B$, $\vdash N : B \to A$ and M and N give an isomorphism pair. The main question about type isomorphisms is usually to find the equational theory characterizing them in a given calculus.

Proving that a given equation on types is a valid isomorphism only requires us to exhibit a pair of typed terms (of appropriate types) and to prove that their compositions in both directions are equal to the identity up to the appropriate equational theory on terms.

This is what we are looking at here with the Church style first-order $\lambda\mu$-calculus with terms equal up to the $\beta\eta\mu\rho\theta$ equality.

Proposition 11 (Syntactic isomorphisms)
For each equation of Table 3, the Table 8 gives such a pair of terms validating the equation.

PROOF: This proof is left to the reader. We just give one example (for the equation $\forall x(A \wedge B) = \forall xA \wedge \forall xB$) of the kind of computation we have to apply:

$$\begin{aligned}
&\lambda c.(\lambda a.\langle \Lambda x.\pi_1 a\{x\}, \Lambda x.\pi_2 a\{x\}\rangle)(\lambda a.\Lambda x.\langle(\pi_1 a)\{x\}, (\pi_2 a)\{x\}\rangle)c\\
&\quad =_\beta \lambda c.(\lambda a.\langle \Lambda x.\pi_1 a\{x\}, \Lambda x.\pi_2 a\{x\}\rangle)\Lambda x.\langle(\pi_1 c)\{x\}, (\pi_2 c)\{x\}\rangle\\
&\quad =_\beta \lambda c.\langle \Lambda x.\pi_1(\Lambda x.\langle(\pi_1 c)\{x\}, (\pi_2 c)\{x\}\rangle)\{x\}, \Lambda x.\pi_2(\Lambda x.\langle(\pi_1 c)\{x\}, (\pi_2 c)\{x\}\rangle)\{x\}\rangle\\
&\quad =_\beta \lambda c.\langle \Lambda x.\pi_1\langle(\pi_1 c)\{x\}, (\pi_2 c)\{x\}\rangle, \Lambda x.\pi_2(\Lambda x.\langle(\pi_1 c)\{x\}, (\pi_2 c)\{x\}\rangle)\{x\}\rangle\\
&\quad =_\beta \lambda c.\langle \Lambda x.\pi_1\langle(\pi_1 c)\{x\}, (\pi_2 c)\{x\}\rangle, \Lambda x.\pi_2\langle(\pi_1 c)\{x\}, (\pi_2 c)\{x\}\rangle\rangle\\
&\quad =_\beta \lambda c.\langle \Lambda x.(\pi_1 c)\{x\}, \Lambda x.\pi_2\langle(\pi_1 c)\{x\}, (\pi_2 c)\{x\}\rangle\rangle\\
&\quad =_\beta \lambda c.\langle \Lambda x.(\pi_1 c)\{x\}, \Lambda x.(\pi_2 c)\{x\}\rangle\\
&\quad =_\eta \lambda c.\langle \pi_1 c, \Lambda x.(\pi_2 c)\{x\}\rangle\\
&\quad =_\eta \lambda c.\langle \pi_1 c, \pi_2 c\rangle\\
&\quad =_\eta \lambda c.c
\end{aligned}$$

$$\lambda a.\langle\langle \pi_1 a, \pi_1\pi_2 a\rangle, \pi_2\pi_2 a\rangle : A \wedge (B \wedge C) \to (A \wedge B) \wedge C$$
$$\lambda a.\langle \pi_1\pi_1 a, \langle \pi_2\pi_1 a, \pi_2 a\rangle\rangle : (A \wedge B) \wedge C \to A \wedge (B \wedge C)$$

$$\lambda a.\pi_1 a : A \wedge \top \to A$$
$$\lambda a.\langle a, \star\rangle : A \to A \wedge \top$$

$$\lambda a.\pi_2 a : \top \wedge A \to A$$
$$\lambda a.\langle \star, a\rangle : A \to \top \wedge A$$

$$\lambda a.\lambda b.\lambda c.(a)\langle b, c\rangle : ((A \wedge B) \to C) \to A \to B \to C$$
$$\lambda a.\lambda b.(a)\pi_1 b\pi_2 b : (A \to B \to C) \to (A \wedge B) \to C$$

$$\lambda a.(a)\star : (\top \to A) \to A$$
$$\lambda a.\lambda d.a : A \to \top \to A$$

$$\lambda a.\langle \lambda b.\pi_1(a)b, \lambda b.\pi_2(a)b\rangle : (A \to (B \wedge C)) \to (A \to B) \wedge (A \to C)$$
$$\lambda a.\lambda b.\langle(\pi_1 a)b, (\pi_2 a)b\rangle : ((A \to B) \wedge (A \to C)) \to A \to (B \wedge C)$$

$$\lambda a.\star : (A \to \top) \to \top$$
$$\lambda a.\lambda d.a : \top \to A \to \top$$

$$\lambda a.\langle \Lambda x.\pi_1 a\{x\}, \Lambda x.\pi_2 a\{x\}\rangle : \forall x(A \wedge B) \to \forall x A \wedge \forall x B$$
$$\lambda a.\Lambda x.\langle(\pi_1 a)\{x\}, (\pi_2 a)\{x\}\rangle : (\forall x A \wedge \forall x B) \to \forall x(A \wedge B)$$

$$\lambda a.\star : \forall x\top \to \top$$
$$\lambda a.\Lambda x.a : \top \to \forall x\top$$

$$\lambda a.\Lambda x.\lambda b.((a)b)\{x\} : (A \to \forall x B) \to \forall x(A \to B)$$
$$\lambda a.\lambda b.\Lambda x.(a\{x\})b : \forall x(A \to B) \to A \to \forall x B$$

$$\lambda a.\langle \pi_2 a, \pi_1 a\rangle : A \wedge B \to B \wedge A$$
$$\lambda a.\langle \pi_2 a, \pi_1 a\rangle : B \wedge A \to A \wedge B$$

$$\lambda a.\Lambda y.\Lambda x.a\{x\}\{y\} : \forall x\forall y A \to \forall y\forall x A$$
$$\lambda a.\Lambda x.\Lambda y.a\{y\}\{x\} : \forall y\forall x A \to \forall x\forall y A$$

Table 8: Isomorphisms

□

The difficult point about type isomorphisms is always the opposite direction: showing that all the equations have been found. This is the purpose of Section 4.

B Categorical properties of games

Let A be an arena without any label (such an arena is called a *ground arena*), strategies on A are exactly those defined in [DHR96, Lau05b]. In particular ground arenas and strategies allow us to define a category $\mathcal{G}_{00}$ (this is the category also used in Section 3.2).

We define the function GR from arenas to ground arenas which erases the labels of its argument. If $\mathtt{s}$ is a play on A, we define $\mathrm{GR}(\mathtt{s})$ as the play on $\mathrm{GR}(A)$ obtained by erasing the instantiations and the μ-pointers (and the same with interaction sequences). We extend GR to sets of plays by applying it point-wise.

Lemma 9
Let $\sigma : A$ be a strategy, if $\mathtt{sm} \in \sigma$ and $\mathtt{tn} \in \sigma$ with $\mathrm{GR}(\mathtt{s}) = \mathrm{GR}(\mathtt{t})$ then there exists an $\mathcal{O}$-renaming ς such that $\mathtt{sm} = \mathtt{tn}\varsigma$.

Proof: We prove it by induction on the common length of $\mathtt{sm}$ and $\mathtt{tn}$. If $\mathrm{GR}(\mathtt{m}_1\mathtt{m}_2) = \mathrm{GR}(\mathtt{n}_1\mathtt{n}_2)$ then $\mathtt{m}_1$ and $\mathtt{n}_1$ are Opponent moves corresponding to the same initial move in A, thus they can only differ on their $\mathcal{O}$-instantiations $[x_1, \ldots, x_k]$ and $[y_1, \ldots, y_k]$ (which must have the same length k). We consider ς such that $\varsigma(y_1) = x_1$, ..., $\varsigma(y_k) = x_k$. We have $\mathtt{m}_1 = \mathtt{n}_1\varsigma$, $\mathtt{m}_1\mathtt{m}_2 \in \sigma$, $\mathtt{n}_1\mathtt{n}_2 \in \sigma$ thus $\mathtt{n}_1\mathtt{n}_2\varsigma \in \sigma$ (by uniformity), and finally $\mathtt{m}_1\mathtt{m}_2 = \mathtt{n}_1\mathtt{n}_2\varsigma$ by determinism. In a similar way, if $\mathrm{GR}(\mathtt{sm}_1\mathtt{m}_2\mathtt{m}_3) = \mathrm{GR}(\mathtt{tn}_1\mathtt{n}_2\mathtt{n}_3)$ then $\mathrm{GR}(\mathtt{s}) = \mathrm{GR}(\mathtt{t})$ and by induction hypothesis $\mathtt{sm}_1 = \mathtt{tn}_1\varsigma$. Moreover $\mathtt{m}_2$ and $\mathtt{n}_2$ are Opponent moves corresponding to the same move in A, they have the same justification pointer and thus they can only differ on their $\mathcal{O}$-instantiations $[x_1, \ldots, x_k]$ and $[y_1, \ldots, y_k]$. We modify ς into ς' in such a way that $\varsigma'(y_1) = x_1$, ..., $\varsigma'(y_k) = x_k$ and ς and ς' agree on the $\mathcal{O}$-variables appearing in $\mathtt{tn}_1$. We have $\mathtt{sm}_1\mathtt{m}_2 = \mathtt{tn}_1\mathtt{n}_2\varsigma'$, $\mathtt{sm}_1\mathtt{m}_2\mathtt{m}_3 \in \sigma$, $\mathtt{tn}_1\mathtt{n}_2\mathtt{n}_3 \in \sigma$ thus $\mathtt{tn}_1\mathtt{n}_2\mathtt{n}_3\varsigma' \in \sigma$ (by uniformity), and finally $\mathtt{sm}_1\mathtt{m}_2\mathtt{m}_3 = \mathtt{tn}_1\mathtt{n}_2\mathtt{n}_3\varsigma'$ by determinism. □

Lemma 10 (Preservation of strategies)
If σ is a strategy on A, then $\mathrm{GR}(\sigma)$ is a strategy on $\mathrm{GR}(A)$.

Proof: It is immediate that $\mathrm{GR}(\sigma)$ is a non-empty set of even length plays which is closed under even length prefixes.

If $\mathtt{sm} = \mathrm{GR}(\mathtt{t}_1\mathtt{m}')$ and $\mathtt{sn} = \mathrm{GR}(\mathtt{t}_2\mathtt{n}')$ with $\mathtt{t}_1\mathtt{m}' \in \sigma$ and $\mathtt{t}_2\mathtt{n}' \in \sigma$. By Lemma 9, $\mathtt{t}_1\mathtt{m}' = \mathtt{t}_2\mathtt{n}'\varsigma$ for some $\mathcal{O}$-renaming ς. As a consequence $\mathtt{sm} = \mathtt{sn}$.

There is nothing to say about uniformity since moves in $\mathrm{GR}(\sigma)$ have no instantiation. □

Lemma 11 (Preservation of composition and identity)
Let σ and τ be strategies on $A \to B$ and $B \to C$, $\mathrm{GR}(\sigma\,;\tau) = \mathrm{GR}(\sigma)\,;\mathrm{GR}(\tau)$ and $\mathrm{GR}(\mathtt{id}_A) = \mathtt{id}_{\mathrm{GR}(A)}$.

Proof: We first prove the inclusion $\mathrm{GR}(\sigma;\tau) \subseteq \mathrm{GR}(\sigma);\mathrm{GR}(\tau)$. Let $\mathtt{s}$ be a play in $\mathrm{GR}(\sigma;\tau)$ coming from the interaction sequence $\mathtt{u}$. $\mathrm{GR}(\mathtt{u})$ is an interaction sequence on $\mathrm{GR}(A)$, $\mathrm{GR}(B)$ and

$\mathrm{GR}(C)$ with $\mathrm{GR}(\mathtt{u}\restriction_{A\to B}) = \mathrm{GR}(\mathtt{u})\restriction_{A\to B}$, $\mathrm{GR}(\mathtt{u}\restriction_{B\to C}) = \mathrm{GR}(\mathtt{u})\restriction_{B\to C}$, and $\mathrm{GR}(\mathtt{u}\restriction_{A\to C}) = \mathrm{GR}(\mathtt{u})\restriction_{A\to C}$. Since $\mathrm{GR}(\mathtt{s}) = \mathrm{GR}(\mathtt{u})\restriction_{A\to C}$, we conclude $\mathrm{GR}(\mathtt{s}) \in \mathrm{GR}(\sigma)\,;\mathrm{GR}(\tau)$.

Concerning $\mathrm{GR}(\sigma)\,;\mathrm{GR}(\tau) \subseteq \mathrm{GR}(\sigma\,;\tau)$, we consider an interaction sequence $\mathtt{v}$ on $\mathrm{GR}(A)$, $\mathrm{GR}(B)$ and $\mathrm{GR}(C)$ with $\mathtt{v}\restriction_{\mathrm{GR}(A)\to\mathrm{GR}(B)} = \mathrm{GR}(\mathtt{s})$ ($\mathtt{s} \in \sigma$), $\mathtt{v}\restriction_{\mathrm{GR}(B)\to\mathrm{GR}(C)} = \mathrm{GR}(\mathtt{t})$ ($\mathtt{t} \in \tau$). By applying an $\mathcal{O}$-renaming if required, we can assume that the $\mathcal{O}$-variables appearing in $\mathtt{s}$ and $\mathtt{t}$ are different. We build from $\mathtt{v}$ an interaction sequence $\mathtt{u}$ on A, B and C in the following way: if $\mathtt{m}$ is a move in C, we add on it the μ-pointers and instantiations coming from $\mathtt{t}$, if $\mathtt{m}$ is a move in A, we add on it the μ-pointers and instantiations coming from $\mathtt{s}$, if $\mathtt{m}$ is a move in B, we add on it the μ-pointers and instantiations coming from both $\mathtt{s}$ and $\mathtt{t}$. We can check that $\mathtt{u}\restriction_{A\to C} \in \mathcal{P}^P_{A\to C}$ so that $\mathtt{u}\restriction_{A\to C} \in \sigma\,;\tau$ (the only point to verify is the condition relating μ-pointers and atomic labels, but it comes easily). This gives $\mathtt{v}\restriction_{\mathrm{GR}(A)\to\mathrm{GR}(C)} = \mathrm{GR}(\mathtt{u}\restriction_{A\to C}) \in \mathrm{GR}(\sigma\,;\tau)$.

The case of the identity is immediate by definition of $\mathtt{id}_A$. □

Lemma 12 (Zipping)
Let $\sigma : A \to B$ and $\tau : B \to C$ be two strategies, let $\mathtt{u}$ and $\mathtt{v}$ be two interaction sequences on A, B and C such that $\mathtt{u}\restriction_{A\to B} \in \sigma$, $\mathtt{v}\restriction_{A\to B} \in \sigma$, $\mathtt{u}\restriction_{B\to C} \in \tau$ and $\mathtt{v}\restriction_{B\to C} \in \tau$, the first move which differs between $\mathtt{u}$ and $\mathtt{v}$ is an Opponent move in $A \to C$.

Proof: This is the very natural extension (with μ-pointers and instantiations) of the usual result for HO/N games proved by induction on the length of the maximal common prefix of $\mathtt{u}$ and $\mathtt{v}$, using the determinism of σ and τ (see [Lau05a] for example). □

Lemma 13
The identity is a strategy, and the composition of two strategies is a strategy.

Proof: $\mathtt{id}_A$ is a non-empty set of even length plays which is closed under even length prefixes. It is clearly uniform. It is deterministic: if $\mathtt{sm} \in \mathtt{id}_A$ and $\mathtt{sn} \in \mathtt{id}_A$, the node in A underlying $\mathtt{m}$ and $\mathtt{n}$ is the same and the justification pointers are the same since $\mathrm{GR}(\mathtt{id}_A)$ is a strategy (Lemma 11), and the μ-pointers and the instantiations are the same by definition.

Let $\sigma : A \to B$ and $\tau : B \to C$ be two strategies, $\sigma\,;\tau$ is a non-empty set of even length plays which is closed under even length prefixes.

If $\mathtt{sm} \in \sigma\,;\tau$ and $\mathtt{sn} \in \sigma\,;\tau$, let $\mathtt{u}$ be an interaction sequence corresponding to $\mathtt{sm}$ and $\mathtt{v}$ be an interaction sequence corresponding to $\mathtt{sn}$, by Lemma 12, one of $\mathtt{u}$ and $\mathtt{v}$ must be a prefix of the other. As a consequence $\mathtt{sm} = \mathtt{sn}$ since they have the same length.

Let $\mathtt{s}$ be a play in $\sigma\,;\tau$, $\mathtt{u}$ be a corresponding interaction sequence and ς be an $\mathcal{O}$-renaming, we define ς' as an $\mathcal{O}$-renaming which coincides with ς on the $\mathcal{O}$-instantiations of $\mathtt{u}$ in A and C (so that $\mathtt{s}\varsigma = \mathtt{s}\varsigma'$) and which maps $\mathcal{O}$-instantiations of $\mathtt{u}$ in B to fresh $\mathcal{O}$-variables (neither appearing in $\mathtt{u}$ nor in the image of the $\mathcal{O}$-instantiations of $\mathtt{u}$ by ς). $\mathtt{u}\varsigma'$ is an interaction sequence on A, B and C such that $\mathtt{u}\varsigma'\restriction_{A\to B} = \mathtt{u}\restriction_{A\to B}\varsigma' \in \sigma$, $\mathtt{u}\varsigma'\restriction_{B\to C} = \mathtt{u}\restriction_{B\to C}\varsigma' \in \tau$, and $\mathtt{u}\varsigma'\restriction_{A\to C} = \mathtt{s}\varsigma' = \mathtt{s}\varsigma$, thus $\mathtt{s}\varsigma \in \sigma\,;\tau$. □

To turn arenas and strategies into a category, we still have to show the composition to be associative and the identity to be neutral for composition.

Proposition 12 (Category of strategies)
Arenas and strategies define a category.

PROOF: Let $\sigma : A \to B$, $\tau : B \to C$ and $\rho : C \to D$ be three strategies, by Lemma 11, we know that $\mathrm{GR}((\sigma\,;\tau)\,;\rho) = \mathrm{GR}(\sigma\,;(\tau\,;\rho))$. This means that we only have to look at μ-pointers and instantiations to show that $(\sigma\,;\tau)\,;\rho = \sigma\,;(\tau\,;\rho)$. We just sketch the arguments (a more precise proof would go through a zipping lemma [Har99, Lemma 3.2.3]).

If $\mathtt{sm} \in (\sigma\,;\tau)\,;\rho$ with $\mathtt{m} \in A$ equipped with a μ-pointer to a move in D, it is obtained by following μ-pointers through B in an interaction sequence coming from the composition $\sigma\,;\tau$ until arriving in C and then by following μ-pointers in C in an interaction sequence coming from the composition $(\sigma\,;\tau)\,;\rho$ until arriving in D. In $\sigma\,;(\tau\,;\rho)$, this μ-pointer is obtained by building a μ-pointer p from B to D by following the μ-pointers through C in an interaction sequence coming from the composition $\tau\,;\rho$ and then by following μ-pointers from A to the source of p through B in an interaction sequence coming from the composition $\sigma\,;(\tau\,;\rho)$. This means that in both case we build a path going from A to D through moves in B and C in the same way, and thus we obtain the same μ-pointer.

Concerning instantiations, they are built by applying an $\mathcal{O}$-substitution ϑ_1 corresponding to pairs of $\mathcal{O}$-instantiations and $\mathcal{P}$-instantiations in B and an $\mathcal{O}$-substitution ϑ_2 corresponding to pairs of $\mathcal{O}$-instantiations and $\mathcal{P}$-instantiations in C. By disjointness of the $\mathcal{O}$-instantiations in B and C, applying first ϑ_1 and then ϑ_2 or the converse leads to the same instantiations in $(\sigma\,;\tau)\,;\rho$ and $\sigma\,;(\tau\,;\rho)$.

The neutrality of the identity with respect to composition is easy and left to the reader. □

Lemmas 10 and 11 show that GR defines a functor from the category of arenas and strategies to the full sub-category of ground arenas and strategies.

We are now able to prove Proposition 3 (page 15).

Lemma 14
Let σ be a view function, $\ulcorner \mathtt{VC}(\sigma) \urcorner = \sigma$.

Lemma 15
Let σ be an innocent strategy, $\mathtt{VC}(\ulcorner \sigma \urcorner) = \sigma$.

PROOF: By definition, there exists a view function τ such that $\sigma = \mathtt{VC}(\tau)$ thus:

$$\begin{aligned} \mathtt{VC}(\ulcorner \sigma \urcorner) &= \mathtt{VC}(\ulcorner \mathtt{VC}(\tau) \urcorner) \\ &= \mathtt{VC}(\tau) && \text{by Lemma 14} \\ &= \sigma \end{aligned}$$

□

Lemma 16 (Composition of view closures)
Let σ be a view function on $A \to B$ and τ be a view function on $B \to C$, there exists a view function ρ on $A \to C$ such that $\mathtt{VC}(\sigma)\,;\mathtt{VC}(\tau) = \mathtt{VC}(\rho)$.

PROOF: In the usual setting of games for extensions of the simply typed λ-calculus, there is a well known direct characterization of innocent strategies, and they are known to compose (see [McC96, Har99] for example). This tells us that $\mathtt{VC}(\sigma)\,;\mathtt{VC}(\tau) = \mathtt{VC}(\rho_0)$ with $\rho_0 = \underline{\ulcorner \mathtt{VC}(\sigma)\,;\mathtt{VC}(\tau) \urcorner}$ (since our definition of pre-view is the usual definition of view and our definition of view closure is the usual one).

What remains to be checked are our constraints on first-order instantiations which are not present in the traditional setting. More precisely, we define $\rho = \ulcorner \mathtt{VC}(\sigma) ; \mathtt{VC}(\tau) \urcorner$ and we have to show that ρ is a view function such that $\mathtt{VC}(\rho) = \mathtt{VC}(\rho_0)$. ρ is a non-empty set of even length views closed under even length prefixes. Moreover $\rho \subseteq \rho_0$ since ρ_0 is closed under $\mathcal{O}$-renaming (by uniformity of $\mathtt{VC}(\sigma) ; \mathtt{VC}(\tau)$). We deduce that ρ is a view function and that $\mathtt{VC}(\rho) \subseteq \mathtt{VC}(\rho_0)$. Finally $\mathtt{VC}(\rho_0) \subseteq \mathtt{VC}(\rho)$ since, for any play $\mathtt{s}$, $\ulcorner \mathtt{s} \urcorner \in \rho_0$ implies $\ulcorner \mathtt{s} \urcorner \in \rho$. □

Proposition 3
Arenas and view functions give a category $\mathcal{G}$.

Proof: The identity view function is a view function on $A \to A$ (whose view closure is the identity strategy). If σ is a view function on $A \to B$ and τ is a view function on $B \to C$, the composition of σ and τ is $\ulcorner \mathtt{VC}(\sigma) ; \mathtt{VC}(\tau) \urcorner$ which is $\ulcorner \mathtt{VC}(\rho) \urcorner = \rho$ (for some view function ρ, by Lemmas 16 and 14) thus it is a view function.

The identity view function is neutral for composition (we give only one side):

$$\begin{aligned}
\sigma ; \ulcorner \mathtt{id} \urcorner &= \ulcorner \mathtt{VC}(\sigma) ; \mathtt{VC}(\ulcorner \mathtt{id} \urcorner) \urcorner \\
&= \ulcorner \mathtt{VC}(\sigma ; \mathtt{id}) \urcorner && \text{by Lemma 15} \\
&= \ulcorner \mathtt{VC}(\sigma) \urcorner \\
&= \sigma && \text{by Lemma 14}
\end{aligned}$$

The composition of view functions is associative:

$$\begin{aligned}
\sigma ; (\tau ; \rho) &= \ulcorner \mathtt{VC}(\sigma) ; \mathtt{VC}(\ulcorner \mathtt{VC}(\tau) ; \mathtt{VC}(\rho) \urcorner) \urcorner \\
&= \ulcorner \mathtt{VC}(\sigma) ; (\mathtt{VC}(\tau) ; \mathtt{VC}(\rho)) \urcorner && \text{by Lemmas 16 and 15} \\
&= \ulcorner (\mathtt{VC}(\sigma) ; \mathtt{VC}(\tau)) ; \mathtt{VC}(\rho) \urcorner \\
&= \ulcorner \mathtt{VC}(\ulcorner \mathtt{VC}(\sigma) ; \mathtt{VC}(\tau) \urcorner) ; \mathtt{VC}(\rho) \urcorner && \text{by Lemmas 16 and 15} \\
&= (\sigma ; \tau) ; \rho
\end{aligned}$$

□

GR turns view functions into view functions, and thus defines a functor from $\mathcal{G}$ to $\mathcal{G}_{00}$.

Additional structure of this category is given by Theorem 1 (page 18). We will prove it after a few lemmas.

Lemma 17 (Composition of μ-rigid strategies)
Let $\sigma : A \to B$ and $\tau : B \to C$ be two μ-rigid strategies, $\sigma ; \tau$ is μ-rigid.

Proof: We prove (by induction on its length) that any interaction sequence $\mathtt{u}$ on A, B and C, such that $\mathtt{u} \upharpoonright_{A \to B} \in \sigma$, $\mathtt{u} \upharpoonright_{B \to C} \in \tau$, has its μ-pointers given in a μ-rigid way. If $\mathtt{u}$ is empty, the result is immediate, if the last move corresponds to a Player move of σ, the result comes from the μ-rigidity of σ and the same with τ. As a consequence, the μ-pointers in $\mathtt{u} \upharpoonright_{A \to C}$ respect the μ-rigidity.

In a similar way, between two moves $\mathtt{m}$ and $\mathtt{n}$ ($\mathtt{n}$ Player move in $A \to C$) in A or C in $\mathtt{u}$, the $\mathcal{O}$-substitution induced by $\mathtt{u}$ identifies each $\mathcal{O}$-instantiation with the previous one in moves in B, and we deduce that the $\mathcal{P}$-instantiation of $\mathtt{n}$ is the same as the $\mathcal{O}$-instantiation of $\mathtt{m}$. □

Lemma 18
Let σ and τ be two μ-rigid strategies on A, if $\mathrm{GR}(\sigma) = \mathrm{GR}(\tau)$ *then* $\sigma = \tau$.

PROOF: By symmetry, it is enough to show $\sigma \subseteq \tau$. We prove by induction on the length of $\mathtt{s}$ that $\mathtt{s} \in \sigma$ entails $\mathtt{s} \in \tau$. The case of ε is immediate. If $\mathtt{smn} \in \sigma$, there exists $\mathtt{tm'n'} \in \tau$ such that $\mathrm{GR}(\mathtt{smn}) = \mathrm{GR}(\mathtt{tm'n'})$. By induction hypothesis, $\mathtt{s} \in \tau$, thus by Lemma 9 there exists an $\mathcal{O}$-renaming ς such that $\mathtt{s} = \mathtt{t}\varsigma$. We consider an $\mathcal{O}$-renaming ς' such that $\mathtt{s} = \mathtt{t}\varsigma'$ and ς' maps each $\mathcal{O}$-variable in the $\mathcal{O}$-instantiation of $\mathtt{m'}$ to the corresponding $\mathcal{O}$-variable in the $\mathcal{O}$-instantiation of $\mathtt{m}$. We look at $\mathtt{smn''} = (\mathtt{tm'n'})\varsigma' \in \tau$. We want to prove $\mathtt{smn} = \mathtt{smn''}$. We have $\mathrm{GR}(\mathtt{smn}) = \mathrm{GR}(\mathtt{smn''})$, thus $\mathtt{n}$ and $\mathtt{n''}$ correspond to the same node in A and have the same justification pointer. Moreover they are μ-rigid thus their μ-pointers and instantiations are obtained in the same way. □

These two lemmas show that GR, restricted to the sub-category of $\mathcal{G}$ given by μ-rigid strategies, is faithful.

Theorem 1 (Control category of games)
The category $\mathcal{G}$ of arenas and view functions is a control category.

PROOF: Since $\mathcal{G}_{00}$ is a control category [Lau05b] and GR preserves the various constructions on arenas and strategies ($\mathrm{GR}(A+B) = \mathrm{GR}(A)+\mathrm{GR}(B)$, $\mathrm{GR}(\sigma+\tau) = \mathrm{GR}(\sigma)+\mathrm{GR}(\tau)$, $\mathrm{GR}(A \times B) = \mathrm{GR}(A) \times \mathrm{GR}(B)$, $\mathrm{GR}(\sigma \times \tau) = \mathrm{GR}(\sigma) \times \mathrm{GR}(\tau)$, ...) as well as basic morphisms (such as associativity and commutativity of the constructions, which are μ-rigid), any commutative diagram required in the definition of a control category and concerning only μ-rigid strategies commutes in $\mathcal{G}$ (by faithfulness of GR on μ-rigid strategies).

The other properties are about the monoid structure (with respect to the pre-monoidal product) defined on each object and about cartesian closedness. They are not difficult to check and left to the reader. □

Lemma 19
GR *reflects totality and finiteness:* $\mathrm{GR}(\sigma)$ *is total if and only if* σ *is total,* $\mathrm{GR}(\sigma)$ *is finite if and only if* σ *is finite.*

Proposition 13 (Composition of total finite strategies)
The composition of two total finite strategies is a total finite strategy.

PROOF: Let σ and τ be two total finite strategies, $\sigma\,;\tau$ is total finite iff $\mathrm{GR}(\sigma\,;\tau)$ is total finite (by Lemma 19) iff $\mathrm{GR}(\sigma);\mathrm{GR}(\tau)$ is total finite. By the full completeness result of [DHR96], $\mathrm{GR}(\sigma)$ and $\mathrm{GR}(\tau)$ are the interpretations of two simply typed λ-terms M and N and $\mathrm{GR}(\sigma)\,;\mathrm{GR}(\tau)$ is the interpretation of $\lambda x.(M)(N)x$ thus a total finite strategy. □

It would be possible to extend this categorical analysis of our game model by introducing a notion of first-order control hyperdoctrines (in the spirit of control hyperdoctrines [dL08]), and by proving our games to give such a first-order control hyperdoctrine. We do not think it would help a lot in the present work.

References

[AJM00] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, December 2000.

[Bar84] Henk Barendregt. *The lambda calculus, its syntax and semantics*. Number 103 in Studies in Logic and the Foundations of Mathematics. North-Holland, second edition, 1984.

[Bef04] Emmanuel Beffara. Realizability with constants. Technical Report oai:hal.archives-ouvertes.fr:hal-00003724_v1, Laboratoire Preuves, Programmes et Systèmes, May 2004. International Workshop on Formal Methods and Security.

[Coq95] Thierry Coquand. A semantics of evidence for classical arithmetic. *Journal of Symbolic Logic*, 60(1):325–337, March 1995.

[DC76] Mariangiola Dezani-Ciancaglini. Characterization of normal forms possessing an inverse in the $\lambda\beta\eta$-calculus. *Theoretical Computer Science*, 2(3):323–337, 1976.

[DC95] Roberto Di Cosmo. *Isomorphisms of Types*. Progress in Theoretical Computer Science. Birkhäuser, 1995.

[DH01] Vincent Danos and Russell Harmer. The anatomy of innocence. In Laurent Fribourg, editor, *Computer Science Logic*, volume 2142 of *Lecture Notes in Computer Science*, pages 188–202. Springer, September 2001.

[DHR96] Vincent Danos, Hugo Herbelin, and Laurent Regnier. Game semantics and abstract machines. In *Proceedings of the eleventh annual symposium on Logic In Computer Science*, pages 394–405, New Brunswick, July 1996. IEEE, IEEE Computer Society Press.

[dL07] Joachim de Lataillade. *Quantification du second ordre en sémantique des jeux — Application aux isomorphismes de types*. Thèse de doctorat, Université Paris VII, November 2007.

[dL08] Joachim de Lataillade. Second-order type isomorphisms through game semantics. *Annals of Pure and Applied Logic*, 151(2–3):115–150, February 2008.

[Fel85] Walter Felscher. Dialogues, strategies, and intuitionistic provability. *Annals of Pure and Applied Logic*, 28:217–254, 1985.

[FH02] Claudia Faggian and Martin Hyland. Designs, disputes and strategies. In Julian C. Bradfield, editor, *Computer Science Logic*, volume 2471 of *Lecture Notes in Computer Science*, pages 442–457. Springer, September 2002.

[Gir99] Jean-Yves Girard. On the meaning of logical rules I: syntax vs. semantics. In Ulrich Berger and Helmut Schwichtenberg, editors, *Computational Logic*, pages 215–272. Springer, 1999. NATO series F 165.

[Gir01] Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, June 2001.

[Har99] Russell Harmer. *Games and Full Abstraction for Nondeterministic Languages*. Ph.D. thesis, Imperial College and University of London, 1999.

[Her97] Hugo Herbelin. Games and weak-head reduction for classical PCF. In Philippe de Groote and Roger Hindley, editors, *Typed Lambda Calculi and Applications '97*, volume 1210 of *Lecture Notes in Computer Science*, pages 214–230. Springer, 1997.

[HL06] Russ Harmer and Olivier Laurent. The anatomy of innocence revisited. In S. Arun-Kumar and Naveen Garg, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *Lecture Notes in Computer Science*, pages 224–235. Springer, December 2006.

[HO00] Martin Hyland and Luke Ong. On full abstraction for PCF. *Information and Computation*, 163(2):285–408, December 2000.

[Hug97] Dominic Hughes. Games and definability for system F. In *Proceedings of the twelfth annual symposium on Logic In Computer Science* [IEE97], pages 76–86.

[IEE97] IEEE. *Proceedings of the twelfth annual symposium on Logic In Computer Science*, Warsaw, June 1997. IEEE Computer Society Press.

[Kri01] Jean-Louis Krivine. Typed lambda-calculus in classical Zermelo-Fraenkel set theory. *Archive for Mathematical Logic*, 40(3):189–205, 2001.

[Kri06] Jean-Louis Krivine. Realizability: a machine for analysis and set theory. Winter school GeoCal '06 lecture notes. Available at `http://cel.archives-ouvertes.fr/cel-00154509`, February 2006.

[Kri07] Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, September 2007.

[Lai97] James Laird. Full abstraction for functional languages with control. In *Proceedings of the twelfth annual symposium on Logic In Computer Science* [IEE97], pages 58–67.

[Lai01] James Laird. A fully abstract game semantics of local exceptions. In *Proceedings of the sixteenth annual symposium on Logic In Computer Science*, pages 105–114, Boston, June 2001. IEEE, IEEE Computer Society Press.

[Lai05] James Laird. Game semantics and linear CPS interpretation. *Theoretical Computer Science*, 333:199–224, 2005.

[Lau02] Olivier Laurent. *Étude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II, March 2002.

[Lau04] Olivier Laurent. Polarized games. *Annals of Pure and Applied Logic*, 130(1–3):79–123, December 2004.

[Lau05a] Olivier Laurent. Classical isomorphisms of types. *Mathematical Structures in Computer Science*, 15(5):969–1004, October 2005.

[Lau05b] Olivier Laurent. Syntax vs. semantics: a polarized approach. *Theoretical Computer Science*, 343(1–2):177–206, October 2005.

[Lor60] Paul Lorenzen. Logik und agon. In *Atti del Congresso Internazionale di Filosofia*, pages 187–194. Sansoni, 1960.

[McC96] Guy McCusker. *Games and Full Abstraction for a Functional Metalanguage with Recursive Types*. Ph.D. thesis, Imperial College and University of London, 1996. Published in Springer-Verlag's Distinguished Dissertations in Computer Science series, 1998.

[Mim09] Samuel Mimram. The structure of first-order causality. In *Proceedings of the twenty-fourth annual symposium on Logic In Computer Science*. IEEE, IEEE Computer Society Press, August 2009.

[Mur01] Andrzej Murawski. *On Semantic and Type-Theoretic Aspects of Polynomial-Time Computability*. Ph.D. thesis, University of Oxford, 2001.

[Nic94] Hanno Nickau. Hereditarily sequential functionals. In Anil Nerode and Yuri Matiyasevich, editors, *Logical Foundations of Computer Science*, volume 813 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 1994.

[Par92] Michel Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings of International Conference on Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 1992.

[Par97] Michel Parigot. Strong normalization for second order classical natural deduction. *Journal of Symbolic Logic*, 62(4):1461–1479, December 1997.

[Py98] Walter Py. *Confluence en $\lambda\mu$-calcul*. Thèse de doctorat, Université de Savoie, Chambéry, July 1998.

[Sel01] Peter Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, April 2001.