Innocent Game Semantics via Intersection Type Assignment Systems*

Pietro Di Gianantonio[1] and Marina Lenisa[2]

1 **Dipartimento di Matematica e Informatica, Università di Udine, Italy**
`pietro.digianantonio@uniud.it`
2 **Dipartimento di Matematica e Informatica, Università di Udine, Italy**
`marina.lenisa@uniud.it`

Abstract

The aim of this work is to correlate two different approaches to the semantics of programming languages: *game semantics* and *intersection type assignment systems* (ITAS). Namely, we present an ITAS that provides the description of the semantic interpretation of a typed lambda calculus in a *game* model based on innocent strategies. Compared to the traditional ITAS used to describe the semantic interpretation in domain theoretic models, the ITAS presented in this paper has two main differences: the introduction of a notion of labelling on moves, and the omission of several rules, *i.e.* the subtyping rules and some structural rules.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages. Denotational semantics

Keywords and phrases Game Semantics, Intersection Type Assignment System, Lambda Calculus.

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Game semantics has been proved a powerful and flexible tool to describe the semantics of programming languages. Its main idea is to define the behaviour of a program as a sequence of elementary interactions between the program and the environment. *Intersection types* have first been used to provide *logical descriptions* of domain theory models of λ-calculus [7, 8], but they can be applied to general programming languages. The approach can be outlined as follows. The semantics of the λ-calculus can be given in two forms: a term can be interpreted either *denotationally* by a point in a particular domain, or *logically* by a set of *properties*. *Stone-duality*, as presented in [1], establishes an equivalence between these two alternative descriptions for suitable categories of domains. In the ITAS approach, properties of terms are normally called "types". The logical semantics consists of the set of rules which allow to derive the properties satisfied by a term. ITAS can be used to provide concrete, *finitary* approximations of the semantics of a term.

The present work continues the line of research of [10], aiming at correlating the game semantics and ITAS. These two approaches to the semantics of programming languages seem, at first sight, quite distant one from the other, establishing a relation can enlighten a different perspective on them. Moreover, compared to game semantics, intersection types have a simpler and more direct presentation, so it is interesting to consider what aspects of game

* Work partially supported by the Italian MIUR PRIN Project CINA 2010LHT4KM, and by the ICT COST Action BETTY IC1201.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 1–17

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

semantics can be described through them. In [10], the authors have considered a simply-typed λ-calculus, a game model for it, based on games à la Abramsky-Jagadeesan-Malacaria [2] (AJM games), and a corresponding ITAS. It has been shown that such ITAS gives a precise description of the interpretation of λ-terms in the AJM-game model. In particular, a type t for a term M describes a set of moves that the Proponent and the Opponent may exchange in some phases of the interaction of the term M with the environment, and the set of types assigned to a term gives a complete description of the history-free strategy of the term, seen as a partial function on moves. However, some aspects of game semantics are not captured by the ITAS description of AJM-games, for example in AJM-games the same strategies have several possible descriptions, differing by the use of indexes on moves. In order to capture the equivalence relation between strategies, a notion of equivariance between plays has been introduced. However, in the ITAS description, it is not clear how to capture this equivalence relation in a natural and simple way.

In the present work, the aim is to extend results presented in [10], by enlarging the aspects of game semantics that can be described through ITAS, and by considering alternative paradigms of game semantics. In particular, we obtain an ITAS description of *innocent games*, *i.e.* games based on innocent strategies. In this way, we show that the ITAS approach can be used for different paradigms of game semantics. Moreover, for innocent games, each strategy has a single representation, therefore a drawback of the ITAS description of AJM-games, namely the missing equivalence relation between alternative representations of the same strategy, is avoided. In more detail, we show that, by introducing in the ITAS a limited set of structural rules, it is possible to describe the interpretation of λ-terms in the framework of innocent games. The structural rules considered state that the $\wedge$ operator satisfies the associative and commutative properties, but *not* the idempotency. So, from an ITAS perspective, the difference between the two main paradigms of game semantics is reflected by the presence/absence of some structural rules. Technically, intersection types for innocent games carry more structure, since they represent sets of moves which are partitioned via a suitable labelling, and are (implicitly) endowed with justification pointers. Non-idempotent intersection types have been also considered in [15, 16, 9]. In particular, in [15, 16], it has be shown that in non-idempotent ITAS, any term t has a principal type τ that gives a complete description of the normal form of t.

In this paper, we chose, for the sake of simplicity, as target language unary PCF, and we provide an innocent game model for it. The ITAS presentation of this model exploit an alternative description of innocent strategies via *partitioned positions*, which we introduce in this paper. Given a play on a game A, by forgetting the linear order along with moves are played, one obtains a set of moves together with justification pointers. We call this kind of structure *position*. A partitioned position is a play where only part of the information concerning the order in which the moves have been laid down has been omitted. In the ITAS that we present, types correspond to partitioned positions. The induced type semantics turns out to provide the same theory of the game semantics. When simple positions without partitions are considered, the corresponding types and ITAS result simplified, however the theory induced by the game semantics is strictly included in that induced by the type semantics.

The idea to remove, partially or completely, the order information on games has been considered several times in the literature [3, 12, 14, 4, 6]. In [3], timeless games are used to build a model for classical linear logic. In timeless games the order on plays is completely forgotten as in the presentation of innocent strategies via simple positions. In [4], it has been shown that the operation of forgetting the time order can be described by a suitable

functor. In [14], it has been shown that, for the particular class of asynchronous games, a strategy can be completely characterized by the set of its *positions*. In [12, 6], a faithful functor from the category of games to a category of relations is presented, that forgets part of the time order along with moves are played.

Synopsis. In Section 2, we recall basic definitions and constructions on arenas and innocent strategies. In particular, we provide a characterisation of innocent strategies via partitioned positions. In Section 3, we present a game model of unary PCF. In Section 4, we introduce and study an ITAS giving a finitary description of the model of Section 3. In Section 5, we establish the connection between the ITAS and the game model of unary PCF. Finally, in Section 6, we discuss further developments.

Acknowledgements. The authors thank the anonymous referees for their comments, which allowed to improve the presentation of the paper.

2 The Category of Arenas and Innocent Strategies

In this section, we recall basic notions and constructions on arenas and innocent strategies in the style of [11]. Notice that we define justification sequences as containing exactly one initial move.

The following are the usual definitions of arena and strategy:

▶ **Definition 1** (Arena). An *arena* has two participants: the *Proponent* and the *Opponent*. An arena is specified by a triple $A = (M_A, \lambda_A, \vdash_A)$, where

- M_A is the set of *moves*.
- $\lambda_A : M_A \to \{OQ, OA, PQ, PA\}$ is the *labelling* function: it tells us if a move is taken by the Opponent or by the Proponent, and if it is a question or an answer. We denote by $^-$ the function which exchanges Proponent and Opponent.
- $\vdash_A$ is a relation between $M_A + \{\star\}$ and M_A, called *enabling*, which satisfies
  - $\star \vdash_A a \implies (b \vdash_A a \Leftrightarrow b = \star) \wedge \lambda_A(a) = OQ$, and
  - $a \vdash_A b \wedge a \neq \star \implies a$ is a question, *i.e.* $\pi_2 \circ \lambda_A(a) = Q$, and $\pi_1 \circ \lambda_A(a) \neq \pi_1 \circ \lambda_A(b)$.

The enabling relation tells us either that a move a is *initial* and needs no justification ($\star \vdash_A a$), or that it can be justified by another move b, if b has been played ($b \vdash_A a$).

▶ **Definition 2.**

- A *justified sequence* s of moves in an arena A is a sequence of moves together with *justification pointers* such that: *the first move is the only initial move*, and for each other move a in s there is a pointer to an earlier move b of s such that $b \vdash_A a$. We say that the move b *justifies* the move a, and we extend this terminology to say that a move b *hereditary justifies* a if the chain of pointers back from a passes through b.
- Given a justified sequence s, the *view* of s, $view(s)$, also called *P-view*, is defined as follows: $view(\epsilon) = \epsilon$, where ϵ denotes the empty sequence, $view(s \cdot a) = a$, if $s = \epsilon$ or a is an initial move, $view(s \cdot a \cdot t \cdot b) = view(s) \cdot a \cdot b$, if the move b is justified by a.
- If s is a justified sequence containing a move a, we say that a is *visible at* s if a appears in $view(s)$.
- A non-empty justified sequence s is a *play* iff
  - O moves first: $s = as'$ and $\pi_1 \circ \lambda(a) = O$
  - s is *alternating*: if $s = s_1 ab s_2$ then $\pi_1 \circ \lambda(a) \neq \pi_1 \circ \lambda(b)$
  - the *visibility condition* holds: if $s = s_1 a s_2$, and a is not initial, then the justifier of a is visible at s_1

- the *well-bracketing condition* holds: if $s = s_1 a s_2$, and a is an answer, then it must be justified by the most recent unanswered question.

The set of plays of an arena A is denoted by P_A.

Notice that the above definition is slightly different from the standard one: plays are *not* empty, and the *initial move* is *unique*. This presentation will provide a better correspondence with the intersection type assignment system.

▶ **Definition 3** (Strategy). A *strategy* for the Proponent on an arena A is a set $\sigma \subseteq P_A^{even}$ of plays of even length such that: $sab \in \sigma \implies s \in \sigma$ and $sab, sac \in \sigma \implies b = c$.

A strategy σ on an arena A is *innocent* if for all $sab, t \in \sigma$, if $ta \in P_A$ and $view(sa) = view(ta)$, then also $tab \in \sigma$, with b justified by the same element of $view(ta) = view(sa)$ as in sab.

A strategy can be seen as a set of rules which tells the Proponent which move to take after the last move by the Opponent. Innocent strategies are strategies which depend only on the *view*.

Constructions on Arenas

▶ **Definition 4** (Product). Given arenas A and B, the product $A \times B$ is the arena defined as follows:

- $M_{A\times B} = M_A + M_B$
- $\lambda_{A\times B} = [\lambda_A, \lambda_B]$
- $\star \vdash_{A\times B} m \iff \star \vdash_A m \vee \star \vdash_B m$ and $m \vdash_{A\times B} n \iff m \vdash_A n \vee m \vdash_B n$.

Here $+$ denotes disjoint union of sets, that is $A + B = \{(l, a) \mid a \in A\} \cup \{(r, b) \mid b \in B\}$, and $[-,-]$ is the usual (unique) decomposition of a function defined on disjoint unions.

The unit for $\times$ is $I = (\emptyset, \emptyset, \emptyset)$.

▶ **Definition 5** (Implication). Given games A and B, the compound game $A \to B$ is defined as follows:

- $M_{A\to B} = M_A + M_B$
- $\lambda_{A\to B} = [\overline{\lambda_A}, \lambda_B]$
- $\star \vdash_{A\to B} m \iff \star \vdash_B m$ and
  $m \vdash_{A\to B} n \iff m \vdash_A n \vee m \vdash_B n \vee (\star \vdash_B m \wedge \star \vdash_A n)$.

The Game Category $\mathcal{G}$

Objects: arenas.
Morphisms: a morphism between arenas A and B is an innocent strategy σ on $A \to B$.
Composition: given innocent strategies $\sigma : A \to B$ and $\tau : B \to C$, $\tau \circ \sigma : A \to C$ is defined by: $\tau \circ \sigma = \{s \restriction (A, C) \mid s \in \sigma || \tau\}^{even}$, where $\sigma || \tau = \{s \in (M_A + M_B + M_C)^* \mid s \restriction (A, B) \subseteq \sigma \ \& \ s \restriction (B, C) \in \overline{\tau}\}$, with $s \restriction (A, B)$ denoting the subsequence of s consisting of moves in A and B.
Identity: $id_A : A \to A$, $id_A = \{s \in P_A^{even} \mid \forall t$ even-length prefix of s. $t \restriction 1 = t \restriction 2\}$, where $t \restriction 1$ ($t \restriction 2$) denotes the restriction of s to the first (second) A component.

The arena constructions of product and implication can be made functorial, in such a way that

▶ **Proposition 6.** *The category $\mathcal{G}$ is cartesian closed with $\times$ as cartesian product and $\to$ as exponential. The arena I is the terminal object of the category $\mathcal{G}$.*

2.1 An Alternative Description of Innocent Strategies

The type assignment system we present describes the strategies associated to λ-terms in an indirect way. To establish the connection between ITAS and games semantics interpretation it is necessary to introduce an alternative description of strategies. Instead of describing an innocent strategy by a set of plays, we describe it by a set of *partitioned positions*. Given a play on a game A, by forgetting the linear order along with moves are played, one obtains a set of moves together with justification pointers for all moves but one (the initial move). We call this kind of structure *position*. For a particular class of games, *i.e.* the asynchronous games, Melliès [14] shows that a strategy is completely characterised by the set of its *positions*. This result is not anymore true for generic innocent games. We therefore introduce the new concept of partitioned position. A partitioned position is a play where only part of the information concerning the order in which the moves have been laid down has been omitted. The innocence condition on strategies assures that using the reduced information allows to reconstruct the original full description of the strategy.

▶ **Definition 7** (Partitioned Position)**.** Let A be an arena.

- We define a *position* on the arena A as an unordered tree, whose nodes are (instances) of moves on A and such that
  1. the root is an initial move,
  2. for any node n, all children of n are moves enabled by n.

  We denote by $(m, \{p_1, \dots, p_n\})$ the position with root m and subtrees $p_1, \dots, p_n$.
- A *partitioned position* is a pair (p, E_p), formed by a position p and a partition E_p on the nodes of p. On partitioned positions we consider the partial order $\subseteqq$ given by $(p, E_p) \subseteqq (p, E'_p)$ if E_p is a partition finer than E'_p, *i.e.* each equivalence class of E_p is contained in an equivalence class of E'_p. Since a partition E_p can be also seen as an equivalence relation, for convenience, in some definitions, we will treat E_p as an equivalence relation.

▶ **Definition 8** (Position from Play)**.**

- Given a play s on the arena A, we denote by $[s]^*$ the partitioned position (p, E_p), where
  - the position p is formed by the moves in s together with their justification pointers (a move n is a child of a move m in p if and only if n is justified by m in s);
  - two distinct moves m, n lie in the same set of the partition E_p if and only if m is an Opponent move, and n is the Proponent move immediately following m in the play s.
- Given a strategy σ on the arena A, we denote with $[\sigma]^*$ the set of partitioned positions $\{(p, E_p) \mid \exists s \in \sigma \,.\, [s]^* \subseteqq (p, E_p)\}$.

The function $[\,]^*$ on plays is not injective, that is there can be two distinct plays s and t generating the same partitioned position $[s]^*$ it is not possible to completely recover the linear order of moves in a play s. However, the function $[\,]^*$ is injective on P-views, in fact, given a P-view s and any move m in $[s]^*$, it is possible to define the predecessor of m in s: if m is a Proponent move then its predecessor is the Opponent move laying in the same partition, while if m is an Opponent move, by P-view definition, the predecessor of m is its parent in the tree. Since an innocent strategy is uniquely determined by the set of P-views that it contains ([11], Section 5.2), it follows that the function $[\,]^*$ is injective on innocent strategies. Moreover, from the set $[\sigma]^*$, it is possible to reconstruct the innocent strategy σ. In fact, given a partitioned position (p, E), it is decidable to check if (p, E) is the image of a P-view s along $[\,]^*$, and in this case to reconstruct the P-view s. Therefore, from $[\sigma]^*$, it is possible to define the set of P-views

of σ, and using the construction presented in [11] Section 5.2, from the set of P-views one can define the set of plays of σ.

On the sets of partitioned positions it is possible to define an operation of composition in the following way. A partitioned position (q, E_q) on the arena $A \to B$ can be decomposed in a partitioned position on B, denoted by $(q, E_q) \restriction B$, and in a multiset of partitioned positions in A, denoted by $(q, E_q) \restriction A$. In more detail, if $q = (m, \{p_1, \ldots, p_m, q_1, \ldots, q_n\})$ with $p_1, \ldots, p_m$ having moves in B and $q_1, \ldots, q_n$ having moves in A, $(p, E_p) \restriction B$ is the position $(m, \{p_1, \ldots, p_m\})$ with the inherited partition. The multiset $(p, E_p) \restriction A$ is composed of the multiset of positions $\{q_1, \ldots, q_n\}$ with the inherited partitions.

▶ **Definition 9** (Composition)**.**

- A finite multiset of partitioned positions $\{(q_1, E_{q_1}), \ldots, (q_n, E_{q_n})\}$ in $A \to B$ and a partitioned position (p, E_p) in $B \to C$ *compose* if $(p, E_p) \restriction B = \{(q_1, E_{q_1}) \restriction B, \ldots, (q_n, E_{q_n}) \restriction B\}$. In this case, the composition $\{(q_1, E_{q_1}) \ldots (q_n, E_{q_n})\} \circ (p, E_p)$ is defined as the position:

$$(m,\ \{p_1, \ldots, p_m\} \cup \bigcup_{i \in \{1,\ldots,n\}} \{q_{i,1}, \ldots, q_{i,n_i}\}\ )\ ,$$

  under the hypothesis that $(p, E_p) \restriction C = ((m, \{p_1, \ldots, p_m\}), E_{p'})$ and $(q_i, E_{q_i}) \restriction A = \{(q_{i,1}, E_{q_{i,1}}), \ldots, (q_{i,n_i}, E_{q_{i,n_i}})\}$.
  On the above position we define a partition E as follows: two nodes m, n are related by E iff one of the following conditions holds:
  – the nodes m, n are related either by E_p or by E_{q_i};
  – there exist an index i and a node m' in the arena B such that $(m, m') \in E_p$ and $(n, m') \in E_{q_i}$;
  – there exist indexes i, j and nodes m', n' in the arena B such that $(m, m') \in E_{q_i}$, $(n, n') \in E_{q_j}$, and $(m', n') \in E_p$.
- Given two sets of partitioned positions S in $B \to C$ and T in $A \to B$, the composition $S \circ T$ is defined by

$$\{\{(q_1, E_{q_1}), \ldots, (q_n, E_{q_n})\} \circ (p, E_p)\ \mid\ \{(q_1, E_{q_1}), \ldots, (q_n, E_{q_n})\} \subseteq S,\ (p, E_p) \in T\ \text{ compose}\}.$$

With the above definition of composition, arenas and sets of partitioned positions form the objects and the arrows of a category. It is possible to refine the notion of partitioned position by defining the notion of *well-formed* partitioned position characterizing those positions that are the image, along $[\ ]^*$, of a play. Then one can further define a subcategory having as arrows sets of well-formed partitioned positions. In this subcategory the function $[\ ]^*$ defines the arrow part of a faithful functor from the category of innocent strategies to the one of sets of partitioned positions. However, in the present work we omit this lengthy definition of the category, and we just prove the main property that will be used in the rest of the paper: the function $[\ ]^*$ on strategies preserves composition. The proof of the proposition below appears in the Appendix.

▶ **Proposition 10.** *For any pair of innocent strategies $\sigma : A \to B$ and $\tau : B \to C$, we have that $[\tau \circ \sigma]^* = [\tau]^* \circ [\sigma]^*$.*

2.2 Timeless Games

It is worthwhile to notice that the construction presented above can be repeated using the simpler notion of position, instead of partitioned position. Along this line, one can define a notion of composition between sets of positions, and a function $[\ ]^\bullet$ that associates to

a strategy the set of positions of its plays. As corollary of Proposition 10, one can show that also the function $[\ ]^\bullet$ preserves composition. Although presented in different form, the function $[\ ]^\bullet$ appears in [4]. In this work, positions are described as relations, and it has been shown that $[\ ]^\bullet$ constitutes the arrow part of a functor from the category of arenas and innocent strategies to a suitable category of sets and relations. It turns out that positions are not sufficient to describe innocent strategies, in that it can be the case that two different innocent strategies are mapped to the same set of positions, see at the end of Section 3 below for an example.

3 A Game Model of Unary PCF

In this section, we define a game model of unary PCF. We chose to consider unary PCF, since it is a simple language, with a minimal set of constants, and it allows for a concise presentation of our ITAS. However, the ideas presented in this paper can be immediately extended to more elaborated functional languages with call-by-name reduction.

Models of unary PCF have been extensively studied in the literature, especially extensional ones, see *e.g.* [13, 5]. Here we are interested in the intensional game model arising from the *Sierpinski arena*, which induces the theory of normal forms. In Section 4, we will provide a description of this model via a type assignment system.

We recall that unary PCF is a typed λ-calculus with two ground constants, $\bot, \top$, and a *sequential composition* constant $\&$[1], which takes two arguments of ground type: if its first argument is $\top$, then $\&$ returns its second argument, otherwise, if its first argument is $\bot$, then $\&$ returns $\bot$.

Unary PCF

▶ **Definition 11.** The class *SimType* of simple types over a ground type o is defined by:

$$\text{(SimType } \ni) \quad A \ ::= \ o \ \mid \ A \to A \ .$$

Raw Terms are defined as follows:

$$\Lambda \ni \ M \ ::= \bot \ \mid \ \top \ \mid \ \& \ \mid \ x \ \mid \ \lambda x{:}A.M \ \mid \ MM \ ,$$

where $\bot, \top, \&$ are constants, and $x \in \mathrm{Var}$. We denote by Λ^0 the set of closed λ-terms.

Well-typed terms. We introduce a *proof system* for deriving *typing judgements* of the form $\Delta \vdash M : A$, where Δ is a *type environment*, *i.e.* a finite set $x_1 : A_1, \ldots, x_k : A_k$. The rules of the proof system are the following:

$$\frac{}{\Delta \vdash \bot : o} \qquad \frac{}{\Delta \vdash \top : o} \qquad \frac{}{\Delta \vdash \& : o \to o \to o} \qquad \frac{}{\Delta, x : A \vdash x : A}$$

$$\frac{\Delta, x : A \vdash M : B}{\Delta \vdash \lambda x{:}A.M : A \to B} \qquad \frac{\Delta \vdash M : A \to B \quad \Delta \vdash N : A}{\Delta \vdash MN : B}$$

Conversion rules. The conversion relation between well-typed terms is the least relation generated by the following rules together with the rules for congruence closure (which we

[1] In the literature, this constant is usually denoted by $\wedge$; here we prefer to denote it by $\&$, since the symbol $\wedge$ is used in the intersection type assignment system.

omit):
$\Delta \vdash (\lambda x{:}A.M)N = M[N/x]$
$\Delta \vdash \&\bot M = \bot \quad \Delta \vdash \&\top M = M \quad \Delta \vdash \&M\top = M$

Notice that the conversion rules for & include reductions where the first or the second argument is $\top$, but only the reduction where the first argument is $\bot$. The reduction in the case the second argument is $\bot$ is omitted, in order to keep the correspondence between normal forms and strategies (see Theorem 15 below).

Game Model

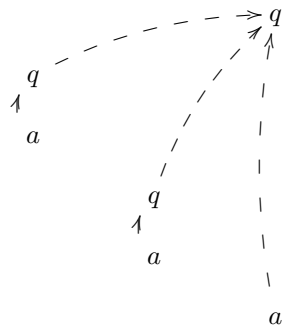
In the cartesian closed category $\mathcal{G}$, simple types are interpreted by the hierarchy of arenas over the following *Sierpinski arena*:

▶ **Definition 12** (Sierpinski Arena)**.** The arena $\mathcal{O}$ is defined as follows:
- $M_{\mathcal{O}} = \{q, a\}$
- $\lambda_{\mathcal{O}}(q) = OQ \quad \lambda_{\mathcal{O}}(a) = PA$
- $\star \vdash_{\mathcal{O}} q$ and $q \vdash_{\mathcal{O}} a$

In the game model, terms in contexts are interpreted as innocent strategies in the usual way, using standard categorical combinators, i.e. $x_1 : A_1, \ldots, x_k : A_k \vdash M : A$ is interpreted as a strategy on the arena $[\![A_1]\!]^{\mathcal{G}} \times \ldots \times [\![A_k]\!]^{\mathcal{G}} \to [\![A]\!]^{\mathcal{G}}$. Before giving the formal interpretation of terms, we first need to define the interpretation of constants.

Interpretation of the basic constants. The interpretation of the constants $\bot$, $\top$ is given by the two strategies on the Sierpinski arena: $[\![\bot]\!]^{\mathcal{G}}$ is the empty strategy, while $[\![\top]\!]^{\mathcal{G}} = \{qa\}$. The interpretation of the constant & is the strategy $[\![\&]\!]^{\mathcal{G}}$ on the arena $\mathcal{O} \to \mathcal{O} \to \mathcal{O}$, defined by the set of plays generated by the even-prefix closure of the play $(r,(r,q))(l,q)(l,a)(r,(l,q))(r,(l,a))(r,(r,a))$ (where justification pointers are omitted).

$[\![\&]\!]^{\mathcal{G}} : \mathcal{O} \longrightarrow \mathcal{O} \longrightarrow \mathcal{O}$



Given an arena A, we denote by $!_A$ the unique empty strategy from the arena A to the terminal arena I. With the obvious isomorphism, a strategy on the arena A can also be seen as a strategy on the arena $I \to A$.

The complete definition of the type and term interpretation in the model is the following:

▶ **Definition 13** (Type and Term Interpretation)**.**
Type interpretation:
$[\![o]\!]^{\mathcal{G}} = \mathcal{O} \qquad [\![A \to B]\!]^{\mathcal{G}} = [\![A]\!]^{\mathcal{G}} \to [\![B]\!]^{\mathcal{G}}$.
Term interpretation:
- $[\![x_1 : A_1, \ldots, x_k : A_k \vdash c : A]\!]^{\mathcal{G}} = [\![c]\!]^{\mathcal{G}} \circ !_{[\![A_1]\!]^{\mathcal{G}} \times \ldots \times [\![A_k]\!]^{\mathcal{G}}}$ if c is a constant.
- $[\![x_1 : A_1, \ldots, x_k : A_k \vdash x_i : A_i]\!]^{\mathcal{G}} = \pi_i : [\![A_1]\!]^{\mathcal{G}} \times \ldots \times [\![A_k]\!]^{\mathcal{G}} \to [\![A_i]\!]^{\mathcal{G}}$
- $[\![\Delta \vdash \lambda x{:}A.M : A \to B]\!]^{\mathcal{G}} = \Lambda([\![\Delta, x : A \vdash M : B]\!]^{\mathcal{G}})$
- $[\![\Delta \vdash MN : B]\!]^{\mathcal{G}} = ev \circ \langle [\![\Delta \vdash M : A \to B]\!]^{\mathcal{G}}, [\![\Delta \vdash N : A]\!]^{\mathcal{G}} \rangle$

where π_i denotes the i-th projection, ev denotes the natural transformation, and Λ denotes the functor characterizing $\mathcal{G}$ as cartesian closed category.

Using standard methods, one can prove that the theory induced by the game model is the theory of $\beta\eta$-normal forms. The notions of β-normal forms and $\beta\eta$-normal forms on unary PCF are the following:

▶ **Definition 14** (β-normal forms, $\beta\eta$-normal forms)**.**
(i) A typed term $\Delta \vdash M : A$ is in *β-normal form* if
- $M \equiv \lambda x_1 : A_1 \dots x_n : A_n.\ \bot$ or
- $M \equiv \lambda x_1 : A_1 \dots x_n : A_n.\top$ or
- $M \equiv \lambda x_1 : A_1 \dots x_n : A_n.\&MM'$, where M, M' are in β-normal form, $M \neq \bot, \top$, and $M' \neq \top$, or
- $M \equiv \lambda x_1 : A_1 \dots x_n : A_n.x_i M_1 \dots M_{q_i}$, where $M_1, \dots, M_{q_i}$ are in β-normal form.

(ii) A typed term $\Delta \vdash M : A$ is in *$\beta\eta$-normal form* if it is in β-normal form and each occurrence of a variable x of type $B_1 \to \dots \to B_k \to o$ in M appears applied to k arguments of types $B_1, \dots, B_k$, respectively.

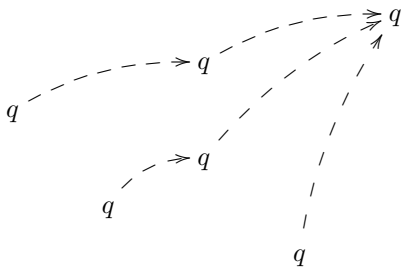
We omit the proof of the following theorem, which is standard:

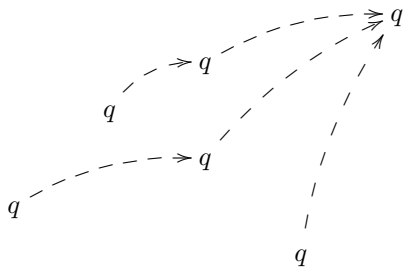
▶ **Theorem 15.** *The theory $Th^{\mathcal{G}}$ induced by the game model $[\![\ ]\!]^{\mathcal{G}}$ is the $\beta\eta$-theory.*

In view of the results in [13], the extensional quotient of the above game model is universal for the observational equivalence on unary PCF (see [13] for more details).

▶ **Example 16.** We conclude this section by providing an example of two different innocent strategies with the same set of positions. Namely, let us consider the terms $P \equiv \lambda x : o \to o \to o.\lambda y : o.x(x\,\bot\,(\&\,y\,\bot))(x\,\bot\,\bot)$ and $Q \equiv \lambda x : o \to o \to o.\lambda y : o.x(x\,\bot\,\bot)(x(\&\,y\,\bot)\,\bot)$. Then, the strategies σ_P and σ_Q interpreting P and Q are different for only two plays:

$\sigma_P : (\mathcal{O} \to \mathcal{O} \to \mathcal{O}) \to \mathcal{O} \to \mathcal{O}$ $\qquad$ $\sigma_Q : (\mathcal{O} \to \mathcal{O} \to \mathcal{O}) \to \mathcal{O} \to \mathcal{O}$





The first play is contained in σ_P but not in σ_Q, while the second one is contained in σ_Q but not in σ_P. However, the two plays above induce the same position, so as all plays extending them, and hence the strategies interpreting P and Q have the same sets of positions.

4 The Type Assignment System

In this section, we introduce and study a type assignment system, which gives a finitary description of the game model of Section 3. The types involved are essentially the standard intersection types, where some structural rules are missing. Our approach to intersection types is "typed", *i.e.* intersection types are built inductively over arenas. The usual untyped intersection semantics (for the untyped λ-calculus) can be recovered as a special case of the typed case.

Intuitively, a type on an arena A represents a partitioned position induced by a play on A. Types on the Sierpinski arena are just sets of moves contained in the possible plays on this arena. As a further ingredient, moves in types are indexed on natural numbers. Indexes are

used to describe partitions: two moves lie in the same partition if and only if they have the same index. A type $(t_1 \wedge \ldots \wedge t_n) \to t$ on the arena $A \to B$ represents a partitioned position composed by a partitioned position on B, described by t, and several partitioned positions on A, described by the types $t_1, \ldots, t_n$. In this approach, the intersection type constructor ($\wedge$) is used to build types on exponential arenas, possibly having multiple instances of the same move. Consequently, the $\wedge$ constructor is *not* idempotent.

The formal correspondence between the type semantics and the game semantics is established in Section 5.

We define a syntax for types that is more complex than the standard one for intersection types. The extra conditions we put on types reflect the alternating and well-bracketing conditions on plays. Namely, for each arena A, we define the set of corresponding intersection types, which divides into *P-types* (t^A_P) and *O-types* (t^A_O), *i.e.* types representing partitioned positions where the Proponent is next to move and types representing partitioned positions where the Opponent is next to move, respectively. Moreover, O-types are divided into "resolved types" (t^A_{Or}), which are intended to represent plays with no pending questions, and "pending types" (t^A_{Op}), which represent plays with pending questions. Notice that all P-types are pending types in this sense.

▶ **Definition 17** (Types). We define two families of *types*, *i.e. Proponent types* (P-types), $\{Type^A_P\}_A$, and *Opponent types* (O-types), $\{Type^A_O\}_A$, these latter are divided into *Opponent resolved types* and *Opponent pending types*, by induction on the structure of the arena A via the following abstract syntax:

- *Types on Sierpinski arena*:

$$(Type^{\mathcal{O}}_P \ni)\ t^{\mathcal{O}}_P ::= \{q_i\} \qquad (Type^{\mathcal{O}}_O \ni)\ t^{\mathcal{O}}_{Or} ::= \{q_i, a_j\} \qquad i,j \in N$$

- *Types on arrow arenas*:

$$\begin{aligned}
&(Type^{A\to B}_P \ni)\ t^{A\to B}_P ::= t^{!A}_{Or} \to t^B_P \ \mid\ t^{!A}_{Op} \to t^B_P \\
&(Type^{A\to B}_O \ni)\ t^{A\to B}_{Or} ::= t^{!A}_{Or} \to t^B_{Or} \\
&(Type^{A\to B}_O \ni)\ t^{A\to B}_{Op} ::= t^{!A}_{Or} \to t^B_{Op} \ \mid\ t^{!A}_{Op} \to t^B_{Op} \ \mid\ t^{!A}_P \to t^B_P
\end{aligned}$$

where

$$\begin{aligned}
&(MType^{!A}_O \ni)\ t^{!A}_{Or} ::= t^A_{Or} \ \mid\ \emptyset^A \ \mid\ t^{!A}_{Or} \wedge t^{!A}_{Or} \\
&(MType^{!A}_O \ni)\ t^{!A}_{Op} ::= t^A_{Op} \ \mid\ t^{!A}_{Op} \wedge t^{!A}_{Op} \ \mid\ t^{!A}_{Op} \wedge t^{!A}_{Or} \\
&(MType^{!A}_P \ni)\ t^{!A}_P ::= t^A_P \ \mid\ t^{!A}_{Or} \wedge t^{!A}_P \ \mid\ t^{!A}_{Op} \wedge t^{!A}_P
\end{aligned}$$

$\emptyset^A$ denotes the empty type on A, and $MType^{!A}_P$ ($MType^{!A}_O$) denotes the set of *Proponent multiple types* (*Opponent multiple types*).

Moreover, we define $Type^A = Type^A_P \cup Type^A_O$, $MType^{!A} = MType^{!A}_P \cup MType^{!A}_O$, $Type = \bigcup_A Type^A$, and $MType = \bigcup_{!A} MType^{!A}$. We use the symbols t^A, u^A, and $t^{!A}, u^{!A}$ to denote types and multiple types respectively, and the symbols t^A_P, u^A_P ($t^{!A}_P, u^{!A}_P$) and t^A_O, u^A_O ($t^{!A}_O, u^{!A}_O$) to denote P (multiple) types and O (multiple) types, respectively.

Finally, we endow the types with the equivalence relation induced by:

$$\emptyset^A \wedge t^{!A} = t^{!A}\ \text{(identity)} \qquad t^{!A}_1 \wedge t^{!A}_2 = t^{!A}_2 \wedge t^{!A}_1\ \text{(commutativity)}$$
$$(t^{!A}_1 \wedge t^{!A}_2) \wedge t^{!A}_3 = t^{!A}_1 \wedge (t^{!A}_2 \wedge t^{!A}_3)\ \text{(associativity)}\ .$$

In the definition of types, justification pointers are not explicitly represented, but they can be recovered from the structure of types.

▶ **Example 18.** The partitioned positions describing the copycat strategy on the arena $\mathcal{O} \to \mathcal{O}$ are induced by the types $\{q_0\} \to \{q_0\}$ and $\{q_0, a_1\} \to \{q_0, a_1\}$.

Notice that, since the type $\{q_0, a_1\} \to \{q_0, a_1\}$ contains as subexpression the type $\{q_0, a_1\}$, the grammar for types needs to generate also types where all indexes are distinct.

To make a more complex example, the two plays that differentiate the strategies σ_P and σ_Q in Example 16 are described by the types:

$$((\{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\}) \wedge (\emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\})) \to \{q_2\} \to \{q_0\}$$
$$((\{q_2\} \to \emptyset^{\mathcal{O}} \to \{q_1\}) \wedge (\emptyset^{\mathcal{O}} \to \{q_1\} \to \{q_0\})) \to \{q_2\} \to \{q_0\}$$

Notice that types on the arena $\mathcal{O} \to \mathcal{O}$ containing a single move are P-types in the form $\emptyset \to \{q_i\}$, while types containing two moves are either Opponent resolved types in the form $\emptyset \to \{q_i, a_j\}$ or Opponent pending types in the form $\{q_j\} \to \{q_i\}$.

Since the grammar does not contain the production $t_P^{!A} \wedge t_P^{!A}$, the type $(\{q_0\} \wedge \{q_1\}) \to \{q_0\}$ does not belong to the grammar; this type describes a play not respecting the alternating condition.

Since the grammar does not contain the production $t_{O_p}^{!A} \to t_{O_r}^{B}$, the type $(\{q_1\} \to \{q_0\}) \to \{q_0, a_1\})$ does not belong to the grammar; this type describes a play not respecting the bracketing condition.

▶ **Definition 19** (Environments).

- *Environments* Γ are finite sets $\{x_1 : t_1^{!A_1}, \dots, x_k : t_k^{!A_k}\}$ with the variables $x_1, \dots, x_k$ all distinct. For simplicity, we omit braces in writing the environments.
- The symbol $\Gamma_\emptyset$ stands for an environment in the form $x_1 : \emptyset^{A_1}, \dots, x_k : \emptyset^{A_k}$.
- Given two environments Γ, Γ' in the form $\Gamma = x_1 : t_1^{!A_1}, \dots, x_k : t_k^{!A_k}$ and $\Gamma' = x_1 : t_1^{'!A_1}, \dots, x_k : t_k^{'!A_k}$, we define $\Gamma \wedge \Gamma'$ as the environment $x_1 : t_1^{!A_1} \wedge t_1^{'!A_1}, \dots, x_k : t_k^{!A_k} \wedge t_k^{'!A_k}$

We introduce a typing system for deriving judgements of the shape $x_1 : t_1^{!A_1}, \dots, x_k : t_k^{!A_k} \vdash M : t^A$, whose intended meaning is to represent a partitioned position in the strategy interpreting the term M in the game model of Section 3.

▶ **Definition 20** (Typing System). The *typing rules* for deriving judgements $x_1 : t_1^{!A_1}, \dots, x_k : t_k^{!A_k} \vdash M : t^A$ are the following:

$$\frac{i \in \mathbb{N}}{\Gamma_\emptyset \vdash \top : \{q_i, a_i\}} \quad (\top)$$

$$\frac{i \in \mathbb{N}}{\Gamma_\emptyset \vdash \& : \{q_i\} \to \emptyset^{\mathcal{O}} \to \{q_i\}} \quad (\&_1)$$

$$\frac{i, j \in \mathbb{N}}{\Gamma_\emptyset \vdash \& : \{q_i, a_j\} \to \{q_j\} \to \{q_i\}} \quad (\&_2)$$

$$\frac{i, j, k \in \mathbb{N}}{\Gamma_\emptyset \vdash \& : \{q_i, a_j\} \to \{q_j, a_k\} \to \{q_i, a_k\}} \quad (\&_3)$$

$$\frac{t^A \in \mathit{Type}^A}{\Gamma_\emptyset, x : t^A \vdash x : t^A} \quad (\mathrm{var})$$

$$\frac{\Gamma, x : u^{!A} \vdash M : t^B}{\Gamma \vdash \lambda x{:}A.M : u^{!A} \to t^B} \quad \text{(abs)}$$

$$\frac{\Gamma \vdash M : u_1^A \wedge \ldots \wedge u_n^A \to t^B \quad \Gamma_1 \vdash N : u_1^A \quad \ldots \quad \Gamma_n \vdash N : u_n^A}{\Gamma \wedge \Gamma_1 \wedge \ldots \wedge \Gamma_n \vdash MN : t^B} \quad \text{(app)}$$

$$\frac{\Gamma \vdash M : \emptyset^A \to t^B \quad \overline{\Gamma} \vdash N : A}{\Gamma \vdash MN : t^B} \quad \text{(app')}$$

where $\overline{\Gamma}$ denotes the simple type environment induced by Γ.

Notice that, in the judgements derivable in the typing system above there is a clear separation between types appearing in the left part (*i.e.* in the environment) and types appearing in the right part: namely, the types in the left part are multiple types, while in the right part only (arrow) types appear.

The extra rule for application (app') is necessary because the expression $\emptyset^A$ only belongs to the grammar of multiple types but not to the grammar of types.

▶ **Example 21.** By the axioms:
$x : \emptyset^{\mathcal{O}\to\mathcal{O}\to\mathcal{O}}, y : \emptyset^{\mathcal{O}} \vdash \& : \{q_2\} \to \emptyset^{\mathcal{O}} \to \{q_2\}$,
$x : \emptyset^{\mathcal{O}\to\mathcal{O}\to\mathcal{O}}, y : \{q_2\} \vdash y : \{q_2\}$,
$x : \emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\}, y : \emptyset^{\mathcal{O}} \vdash x : \emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\}$,
$x : \{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\}, y : \emptyset^{\mathcal{O}} \vdash x : \{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\}$,
using the rules (app) and (app'), we get $x : \emptyset, y : \{q_2\} \vdash \&y\bot : \{q_2\}$.
Again by the rules (app') and (app), $x : \emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\}, y : \{q_2\} \vdash x\bot(\&y\bot) : \{q_1\}$.
By the rules (app) and (app'),
$x : (\{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\} \wedge \emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\}), y : \{q_2\} \vdash x(x\bot(\&y\bot))(x\bot\bot) : \{q_0\}$,
and by a double application of the rule (abs),
$\vdash \lambda x : o \to o \to o.\lambda y : o.x(x \bot (\& \, y \, \bot))(x \bot\bot) : ((\{q_1\} \to \emptyset^{\mathcal{O}} \to \{q_0\}) \wedge (\emptyset^{\mathcal{O}} \to \{q_2\} \to \{q_1\})) \to \{q_2\} \to \{q_0\}$.

Notice that the following rule is admissible:

$$\frac{\Gamma \vdash M : t^A \quad \phi : \mathbb{N} \to \mathbb{N}}{\Gamma\phi \vdash M : t^A\phi} \quad \text{(sub)}$$

where ϕ is a generic a function on natural numbers and $t^A\phi$ denotes the type t^A where all indexes on moves are substituted according to the function ϕ. The rule (sub) can be usefully employed on the premises of the rule (app), in order to derive premises sharing identical indexes on the corresponding types. Notice that, to obtain this result, it can be necessary to identify different indexes, and so the function ϕ, used as parameter in sub, needs to be a general function and not simply a permutation.

The fact that the types in any derivable judgement are well-formed intersection types follows from Lemma 22 below. This lemma can be easily proved by induction on derivations.

▶ **Lemma 22.** *If $x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A$ is derivable, then:*
- *if t^A is a resolved O-type, then all $t_1^{!A_1}, \ldots, t_k^{!A_k}$ are resolved O-types;*
- *if t^A is a pending O-type, then all $t_1^{!A_1}, \ldots, t_k^{!A_k}$ are O-types;*
- *if t^A is a P-type, then at most one of the types in $t_1^{!A_1}, \ldots, t_k^{!A_k}$ is a P-type.*

As a consequence, we have:

▶ **Proposition 23.** *If* $x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A$ *is derivable, then* $(t_1^{!A_1} \to (t_2^{!A_2} \to \ldots (t_k^{!A_k} \to t^A))) \in \mathit{Type}^{(A_1 \to (A_2 \to \ldots (A_k \to A)))}$.

The type assignment system immediately induces a semantics of λ-calculus based on types, whereby any term in context is interpreted by a set of tuples of types as follows:

▶ **Definition 24** (Type Semantics)**.** Let $[\![\ ]\!]^{\mathcal{T}}$ be the interpretation function defined by: $[\![x_1 : A_1, \ldots, x_k : A_k \vdash M : A]\!]^{\mathcal{T}} = \{(t_1^{!A_1}, \ldots, t_k^{!A_k}, t^A) \mid x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A\}$.

5 From Types to Games

In this section, we show that the type semantics coincides with the game semantics. This result follows from the fact that the types appearing in judgements derivable in the intersection type system correspond to partitioned positions in the strategy interpreting the term.

In order to formally state this correspondence, it is useful to introduce the notion of *indexed position*, which is a position where moves are indexed. Clearly, any indexed position determines a partitioned position, where two moves belong to the same partition if and only if they have the same index; we denote by $\mathcal{U} : IP \to PP$ the natural map from indexed to partitioned positions. Vice versa, any partitioned position determines a class of indexed positions, differing by an injective renaming of indexes. Notice that it would have been possible to use only the notion of indexed position, but we have preferred to introduce also partitioned positions, which provide canonical representatives for strategies.

One can easily define a natural map $\mathcal{E}^A : \mathit{Type}^A \to IP^A$, for any set of types Type^A:

▶ **Definition 25.** For any set of intersection types Type^A, we define $\mathcal{E}^A : \mathit{Type}^A \to IP^A$, by induction on the arena A:

- $\mathcal{E}^{\mathcal{O}}(\{q_i\}) = (q_i, \emptyset) \qquad \mathcal{E}^{\mathcal{O}}(\{q_i, a_j\}) = (q_i, \{(a_j, \emptyset)\})$.
- $\mathcal{E}^{A \to B}(t^{!A} \to t^B) = (m', \{p'_1, \ldots, p'_k, \overline{\mathcal{E}^A(t_1^A)}, \ldots, \overline{\mathcal{E}^A(t_n^A)}\})$,
  where
  - $t^{!A} = t_1^A \wedge \ldots \wedge t_n^A$,
  - $\mathcal{E}^B(t^B) = (m', \{p'_1, \ldots, p'_k\})$,
  - $\overline{\mathcal{E}^A(t_i^A)}$ denotes the position where the polarity of moves has been reversed,
  - the move names in $(m', \{p'_1, \ldots, p'_k, \overline{\mathcal{E}^A(t_1^A)}, \ldots, \overline{\mathcal{E}^A(t_n^A)}\})$ are taken up to the obvious injection in $M_A + M_B$.

The maps $\mathcal{E}^A$ can be extended to $k+1$-tuple of types $(t_1^{!A_1}, \ldots, t_k^{!A_k}, t^A)$ as follows:

▶ **Definition 26.** For all $\mathit{MType}^{!A_1}, \ldots, \mathit{MType}^{!A_k}$, Type^A, for any $k \geq 0$, we define a map $\mathcal{E}^{A_1 \times \ldots \times A_k \to A} : \mathit{MType}^{!A_1} \times \ldots \times \mathit{MType}^{!A_k} \times \mathit{Type}^A \to IP^{A_1 \times \ldots \times A_k \to A}$ by induction on the arenas $A_1, \ldots, A_k, A$ as follows:

- for $k = 0$, $\mathcal{E}^A(t_A)$ is defined as in Definition 25;
- for $k > 0$, $\mathcal{E}^{A_1 \times \ldots \times A_k \to A}(t_1^{!A_1}, \ldots, t_k^{!A_k}, t^A) =$
  $$(m', \{p'_1, \ldots, p'_h, \overline{\mathcal{E}^{A_1}(t_{11}^{A_1})}, \ldots, \overline{\mathcal{E}^{A_1}(t_{1n_1}^{A_1})}, \ldots, \overline{\mathcal{E}^{A_k}(t_{k1}^{A_k})}, \ldots, \overline{\mathcal{E}^{A_k}(t_{kn_k}^{A_k})}\}) ,$$
  where
  - $t_i^{!A_i} = t_{i1}^{A_i} \wedge \ldots \wedge t_{in_i}^{A_i}$, for all i,
  - $\mathcal{E}^A(t^A) = (m', \{p'_1, \ldots, p'_h\})$,
  - $\overline{\mathcal{E}^{A_i}(t_{ij}^{A_i})}$, for all i, j, denotes the position where the polarity of moves has been reversed,

- move names in $(m', \{p'_1, \ldots, p'_h, \overline{\mathcal{E}^{A_1}(t^{A_1}_{11})}, \ldots, \overline{\mathcal{E}^{A_1}(t^{A_1}_{1n_1})}, \ldots, \overline{\mathcal{E}^{A_k}(t^{A_k}_{k1})}, \ldots, \overline{\mathcal{E}^{A_k}(t^{A_k}_{kn_k})}\})$ are taken up to the obvious injection in $M_{A_1} + \ldots + M_{A_k} + M_A$.

The maps $\mathcal{E}^A$ and $\mathcal{U}$ determine a correspondence between the type semantics and the game semantics, namely:

▶ **Definition 27.** We define
$\mathcal{F}([\![x_1 : A_1, \ldots, x_k : A_k \vdash M : A]\!]^{\mathcal{T}}) =$
$$\{\mathcal{U} \circ \mathcal{E}^{A_1 \times \ldots \times A_k \to A}(t_1^{!A_1}, \ldots, t_k^{!A_k}, t^A) \mid x_1 : t_1^{!A_1}, \ldots, x_k : t_k^{!A_k} \vdash M : t^A\} \ .$$

Then, we have the following theorem (whose proof appears in the Appendix):

▶ **Theorem 28.**
(i) For any well-typed term $\Delta \vdash M : A$, $\quad \mathcal{F}([\![\Delta \vdash M : A]\!]^{\mathcal{T}}) = [[\![\Delta \vdash M : A]\!]^{\mathcal{G}}]^* \ .$
(ii) The type semantics and the game semantics induce the same theory.

5.1 ITAS without Indexes

A simplified model for unary PCF can be obtained by using an alternative version of ITAS where types are without indexes. In this alternative version the type semantics of a term M defines the set of positions (and not of partitioned positions) in the strategy $[\![M]\!]^{\mathcal{G}}$.

It turns out that the simplified model does *not* provide the theory of the game model. The terms P and Q considered in the Example 16 are interpreted in the game model by two different strategies, σ_P, σ_Q, containing the same set of positions. More precisely, the theory of the simplified model is intermediate between the theory of the game model and its extensional collapse.

Intersection types without idempotency and without indexes have been considered also in [15, 16]. In these works, it has been shown that two terms having the same set of types have also the same normal form. This result is in contrast with what happen in the above sketched ITAS without indexes, where terms P and Q have different normal forms but the same set of types. This difference can be explained by the fact that in our setting the set of types without indexes contains too few elements; in particular on the Sierpinski arena just three types are definable. In contrast, in [15], the untyped lambda calculus and types built over a countable set of type variables are considered. A posteriori, one can argue that, in order to precisely characterize the normal forms of terms, it is necessary to have a sufficiently rich set of types, and the introduction of indexes on types can be seen not only as a way to encode part of the time order on moves, but also as a way to obtain a richer set of types.

6 Conclusions and Further Work

In this work we have shown how a type assignment system can be used to determine the interpretation of λ-terms in an innocent game model. An interesting aspect that has emerged is that using very similar ITAS, essentially differing among them by the use of structural rules, it is possible to capture a large variety of denotational models: various domain theoretic and game models. It will be interesting to systematically investigate the relations between structural rules and the model counterpart.

As a further result, we hope to construct, in the type semantics setting, fully abstract models of programming languages. In game semantics, fully abstract models are obtained by extensional collapse, exploiting full definability results. To repeat the same construction in the type semantics, it is necessary to obtain an analogous result of full definability. This will imply to find a concrete characterisation of semantical objects, that is to characterise

those sets of types which are interpretations of terms, or, by the full definability of innocent strategies, to characterise those sets of types corresponding to innocent strategies. In a different setting, a similar result has been obtained by Mellies, in [14], for asynchronous games. There, it has been shown that an innocent strategy can be described by the set of its positions; moreover, it has been presented a direct characterisation of those sets of positions which correspond to innocent strategies. An analogous characterisation for partitioned positions could be studied. Since in our setting positions are described by types, this is the sort of result we are looking for.

In general, there are several other aspects in game semantics that arguably can be expressed in terms of intersection types. Game semantics is a quite sophisticated theory and so far we have formulated just one part of it in the ITAS approach. Thus, it is natural to investigate what will be a suitable translation of other game semantics concepts.

References

1 S. Abramsky. Domain theory in logical form. In *Annals of Pure and Applied Logic*, volume 51, pages 1–77, 1991.

2 S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.

3 P. Baillot, V. Danos, T. Ehrhard, and L. Regnier. Timeless games. In M. Nielsen et al., editor, *CSL*, volume 1414 of *LNCS*, pages 56–77. Springer, 1997.

4 P. Boudes. Thick subtrees, games and experiments. In *Typed Lambda Calculi and Applications: Proc. 9th Int. Conf. TLCA 2009*, pages 65–79. Springer-Verlag, 2009. LNCS Vol. 5608.

5 A. Bucciarelli, B. Leperchey, and V. Padovani. Relative definability and models of unary PCF. In *TLCA'03*, volume 2701 of *LNCS*, pages 75–89. Springer, 2003.

6 A. Calderon and G. McCusker. Understanding game semantics through coherence spaces. *Electr. Notes Theor. Comput. Sci.*, 265:231–244, 2010.

7 M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic*, 21(4):685–693, 1980.

8 M. Coppo, M. Dezani-Ciancaglini, F. Honsell, and G. Longo. Extended type structure and filter lambda models. In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquium '82*, pages 241–262. Elsevier Science Publishers B.V. (North-Holland), 1984.

9 D. de Carvalho. *Execution Time of Lambda-Terms via Denotational Semantics and Intersection Types*. Mathematical Structure in Computer Science, 1991. To appear.

10 P. Di Gianantonio, F. Honsell, and M. Lenisa. A type assignment system for game semantics. *Theor. Comput. Sci.*, 398(1-3):150–169, 2008.

11 J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF. *Information and Computation*, 163:285–408, 2000.

12 M. Hyland and A. Schalk. Abstract games for linear logic. *ENTCS*, 29:127–150, 1999.

13 J. Laird. A fully abstract bidomain model of unary PCF. In *TLCA'03*, volume 2701 of *LNCS*, pages 211–225. Springer, 2003.

14 P.A. Melliès. Asynchronous games 2: The true concurrency of innocence. *Theor. Comput. Sci.*, 358(2-3):200–228, 2006.

15 P. M. Neergaard and H. G. Mairson. Types, potency, and idempotency: Why nonlinearity and amnesia make a type system work. In *Proc. 9th Int. Conf. Functional Programming*. ACM Press, 2004.

16 S. Salvati. On the membership problem for non-linear abstract categorical grammars. *Journal of Logic, Language and Information*, 19(2):163 – 183, 2010.

A Proofs

Proof of Proposition 10. First we prove that the first set of partitioned positions is included in the second one. Given a play $s \in \tau \circ \sigma$, by the definition of composition of strategies, there exist a set of plays $s_1, \ldots, s_n \in \sigma$ and a play $t \in \tau$, whose interleaving composition gives s. Let $(p, E_p) = [t]^*$, $(q_i, E_{q_i}) = [s_i]^*$. Starting from the partition E_p, we build a coarser partition E'_p by considering each pair of Opponent-Proponent moves m, n belonging to the arena B and forming a partition of E_{q_i}, the corresponding instances are contained in separated partitions of E_p, the two instances are equated in E'_p. In a symmetric fashion, we build a set of partitions $E'_{q_i} \supseteqq E_{q_i}$. It is not difficult to verify that the partitioned positions (p, E'_p) and $\{(q_i, E'_{q_i}) \mid i \in I\}$ compose and their composition coincides with $[s]^*$. This fact proves that $[s]^* \in [\tau]^* \circ [\sigma]^*$. Moreover any other partitioned position $(q, E_q) \supseteqq [s]^*$, can be shown to belong to $[\tau]^* \circ [\sigma]^*$ by repeating the above construction using suitable partitioned position $(p, E''_p) \supseteqq (p, E'_p)$ and $(q_i, E''_{q_i}) \supseteqq (q_i, E'_{q_i})$.

The proof of the reverse inclusion is more complex. From the hypothesis $(p, E_p) \in [\tau]^* \circ [\sigma]^*$, by definition, it follows that there exists a set of plays $s_1, \ldots, s_n \in \sigma$, a play $t \in \tau$ and a set of partitioned positions $(q_1, E_{q_1}), \ldots, (q_n, E_{q_n}), (p', E_{p'})$ such that: $[s_i]^* \subseteqq (q_i, E_{q_i})$, $[t]^* \subseteqq (p', E_{p'})$, $(p, E) \restriction C = (p', E_{p'}) \restriction C$, $(p, E) \restriction A = \bigcup_{i \in 1..n} (q_i, E_{q_i}) \restriction A$, and $(p', E_{p'}) \restriction B = \bigcup_{i \in 1..n} (q_i, E_{q_i}) \restriction B$.

Since the plays $s_1, \ldots s_n$, and t not necessarily have an interleaving composition, using the innocence hypothesis for the strategies σ and τ, we need to construct a second set of plays $s'_1, \ldots, s'_n$ and t', that do have an interleaving composition and such that $[s'_i]^* \subseteqq (q_i, E_{q_i})$, $[t']^* \subseteqq (p', E_{p'})$. Essentially $s'_1, \ldots, s'_n$ and t' coincide with $s_1, \ldots, s_n$ and t on the components A and C, while on B the move following a move m is determined by considering the behaviour of the Proponent in either the plays $s_1, \ldots, s_n$ or in the play t.

In more detail, the plays $s'_1, \ldots, s'_n$ and t' are defined incrementally as follows. First, one considers a bijection j between the B moves in $s_1, \ldots, s_n$ and the B moves in t induced by the equality $(p', E_{p'}) \restriction B = \bigcup_{i \in 1..n} (q_i, E_{q_i}) \restriction B$. Since $(p', E_{p'}) \restriction B$ is a multiset, the bijection j is not unique.

The initial sequence of t' coincides with the initial sequence t_1 of t till the first instance of a move b_1 in the arena B; b_1 must be a Proponent move. Then one considers the move $\overline{b_1}$ associated to b_1 by j; assume that $\overline{b_1}$ lies in the plays s_i, next, one considers the subsequence $s_{i,1}$ of s_i starting from $\overline{b_1}$ till the next move $\overline{b_2}$ in the arena B. The sequence $s_{i,1}$ forms the initial sequence of s'_i. Notice that $s_{i,1}$ can be composed by just two moves, but can also contain moves in A. Notice moreover that $\overline{b_1}$ is an Opponent move, while $\overline{b_2}$ must be a Proponent move. The construction goes on considering the move b_2 in t associated to $\overline{b_2}$ by j and the subsequence t_2 of t starting from b_2 till the next move b_3 in the arena B. The concatenation $t_1 t_2$ defines the initial sequence of t'. Notice that b_2 and b_3 are, respectively, Opponent and Proponent moves.

Repeating the steps presented above, one considers the move $\overline{b_3}$, associated to b_3 by j. If, by chance, $\overline{b_3}$ is contained in the play s_i, one considers the subsequence $s_{i,2}$ of s_1 starting from the move $\overline{b_3}$ to the next move $\overline{b_4}$ in the arena B, the concatenation $s_{i,1} s_{i,2}$ forms the initial sequence of s'_i. If $\overline{b_3}$ lies in a different sequence s_h, the subsequence from $\overline{b_3}$ to $\overline{b_4}$ defines the initial sequence of s'_h. The construction carries on in this way, moving continuously from the play t to the plays $s_1, \ldots, s_n$, till all moves have been considered. It is immediate to check that the constructed plays t' and $s'_1, \ldots, s'_n$ compose. It remains to prove that the the plays t' and $s'_1, \ldots, s'_n$ satisfy the visibility condition and belong to the innocent strategies τ, σ. The plays t' and $s'_1, \ldots, s'_n$ belong to the innocent strategies since

the Proponent view of a move in t' and $s'_1, \ldots, s'_n$ coincides with the Proponent view of the corresponding moves in t and $s_1, \ldots, s_n$. The visibility condition is satisfied since, for any B move, the Proponent view in t' coincides with the Opponent view in $s'_1, \ldots, s'_n$, and vice versa. To conclude the proof, from $t', s'_1, \ldots, s'_n$ one constructs a play s in the strategy $\tau \circ \sigma$ such that $(p, E_p) \supseteq [s]^*$. ◀

Proof of Theorem 15. Clearly, two $\beta\eta$-equivalent terms have the same interpretation in the model. Vice versa, if two terms at a given type have different $\beta\eta$-normal forms, then, by induction on the structure of them, one can show that the corresponding strategies are different. First of all notice that any normal form $\lambda\vec{x} : \vec{A}.\, \& MM'$ must be of the shape $\lambda\vec{x} : \vec{A}.\, \&(\ldots(\&(x_i\vec{M})M'_1)\ldots)M'_k$, *i.e.* there is a subterm $x_i\vec{M}$, and hence the strategy interpreting the whole normal form interrogates the variable x_i, and it is *not* constant. Therefore, the strategies interpreting the normal forms $\lambda\vec{x}{:}\vec{A}.\, \bot$ and $\lambda\vec{x}{:}\vec{A}.\, \top$, being constant, are different from all strategies interpreting other normal forms. Moreover, if the normal forms are of the shape $\lambda\vec{x}{:}\vec{A}.\, \&(\ldots(\&(x_i\vec{M})M'_1)\ldots)M'_k$ and $\lambda\vec{x}{:}\vec{A}.\, x_j\vec{N}$, then, if $i \neq j$, the corresponding strategies are extensionally different (*e.g.*, when x_i is $\bot$ and x_j is $\top$ of the appropriate types, they provide different results). If $i = j$, then, when x_i is $\top$, the strategy corresponding to the second term yields immediately $\top$, while the strategy corresponding to the first term would yield $\top$ immediately only if the strategy interpreting the second argument would be the $\top$-strategy. But, by induction hypothesis, this means that the second argument is $\top$, which cannot be by hypothesis. Now, if the two normal forms are of the shape $\lambda\vec{x}{:}\vec{A}.\, \&M_1M_2$ and $\lambda\vec{x}{:}\vec{A}.\, \&N_1N_2$, then the strategies interpreting them would be $\Lambda \circ ev \circ \langle ev \circ \langle [\![\&]\!]^{\mathcal{G}}, f_1\rangle, f_2\rangle$ and $\Lambda \circ ev \circ \langle ev \circ \langle [\![\&]\!]^{\mathcal{G}}, g_1\rangle, g_2\rangle$, where f_1, f_2, g_1, g_2 are the interpretations of M_1, M_2, N_1, N_2 in the appropriate environments. By induction hypothesis, $f_1 \neq g_1$ or $f_2 \neq g_2$. Notice that the strategy interpreting & starts by interrogating the first argument and, if this provides an answer, it interrogates the second one. But, since the first argument is different from $\bot$, it must provide an answer. Hence, from the fact that $f_1 \neq g_1$ or $f_2 \neq g_2$, we can conclude that the strategies interpreting the the two normal forms are different. Finally, if the normal forms are of the shape $\lambda\vec{x}{:}\vec{A}.x_iM_1 \ldots M_{q_i}$ and $\lambda\vec{x}{:}\vec{A}.x_jM'_1 \ldots M'_{q_j}$, and the head variable is different, then the strategies are different, because the first starts by interrogating the i-th argument, while the second starts by interrogating the j-th argument. If the head variable is the same in the two terms, but the strategies interpreting one of the arguments are different, *i.e.* $[\![\Delta \vdash \lambda\vec{x}{:}\vec{A}.x_iM_1 \ldots M_{q_i} : B]\!]^{\mathcal{G}} = \Lambda^n \circ ev \circ \langle \ldots \langle ev \circ \langle \pi^n_i, g_1\rangle, g_2\rangle \ldots, g_{q_i}\rangle$ and $[\![\Delta \vdash \lambda\vec{x}{:}\vec{A}.x_iM'_1 \ldots M'_{q_i} : B]\!]^{\mathcal{G}} = \Lambda^n \circ ev \circ \langle \ldots \langle ev \circ \langle \pi^n_i, g'_1\rangle, g'_2\rangle \ldots, g_{q'_i}\rangle$ with $g_j \neq g'_j$ for some j, then, by definition of ev, π^n_i and $\langle\ ,\ \rangle$, one can easily check that the overall strategies are also different. ◀

Proof of Theorem 28.
(i) The proof proceeds by induction on the derivation of $\Delta \vdash M : A$, by showing that $\mathcal{F}([\![\Delta \vdash M : A]\!]^{\mathcal{T}})$ is the set of partitioned positions of the strategy $[\![\Delta \vdash M : A]\!]^{\mathcal{G}}$. For M the ground constants $\bot, \top, \&$ or a variable, the thesis directly follows from the definitions of type assignment system and game semantics. For $\Delta \vdash \lambda x : A.M : A \to B$, the thesis easily follows by induction hypothesis. For $\Delta \vdash MN : B$, applying the induction hypothesis, we have that $\mathcal{F}([\![\Delta \vdash M : A \to B]\!]^{\mathcal{T}})$ is the set of partitioned positions of the strategy $[\![\Delta \vdash M : A \to B]\!]^{\mathcal{G}}$, while $\mathcal{F}([\![\Delta \vdash N : A]\!]^{\mathcal{T}})$ is the set of partitioned positions of the strategy $[\![\Delta \vdash N : A]\!]^{\mathcal{G}}$. Then, the thesis follows by rule (*app*) of the type assignment system, and by the characterisation of strategy application when strategies are viewed as sets of partitioned positions (see Section 2.1).
(ii) By item (i), since both $[\ ]^*$ and $\mathcal{F}$ are injective maps, $Th^{\mathcal{T}} = Th^{\mathcal{G}}$. ◀