Categorical Combinatorics for Innocent Strategies

Russ Harmer, Martin Hyland, Paul-André Melliès

Categorical Combinatorics for Innocent Strategies

Russ Harmer
Équipe PPS, Université Paris 7
2, place Jussieu, Case 7014
75251 Paris Cedex 05, France
russ@pps.jussieu.fr

Martin Hyland
DPMMS, University of Cambridge,
CMS, Wilberforce Road,
Cambridge CB1 2BX, UK
m.hyland@dpmms.cam.ac.uk

Paul-André Melliès
Équipe PPS, Université Paris 7
2, place Jussieu, Case 7014
75251 Paris Cedex 05, France
mellies@pps.jussieu.fr

Abstract

We show how to construct the category of games and innocent strategies from a more primitive category of games. On that category we define a comonad and monad with the former distributing over the latter. Innocent strategies are the maps in the induced two-sided Kleisli category. Thus the problematic composition of innocent strategies reflects the use of the distributive law. The composition of simple strategies, and the combinatorics of pointers used to give the comonad and monad are themselves described in categorical terms. The notions of view and of legal play arise naturally in the explanation of the distributivity. The category-theoretic perspective provides a clear discipline for the necessary combinatorics.

1 Introduction

The notion of innocent strategy introduced in [16] and independently in [30] supports a notion of game with pointers with widespread applications. The first application of this form of game semantics to PCF was quickly followed by others: to recursive types [24], to store at ground type [3], to general store [4], to control [20], to nondeterminism [13] and to polymorphism [14]. Since then the range of applications has broadened further. There has been much work and some spectacular success in model checking and verification [2, 29, 31, 32]. The special topic of type isomorphism is treated in [21], while [1] gives a model of a λ-calculus with names. There have also been theoretical developments. For a recent close analysis of aspects of pointers which we do not cover here see [12].

The fact that innocent strategies compose is naturally essential to the many applications, but this fact is not at all trivial. In the main authors tacitly rely on the original proof in [16] or else on its reworking in [25]. Some writers purport to avoid the issue by appeal to a full abstraction result; but such a reliance on the compositionality of a corresponding syntax dilutes the interest of their semantic analysis. In this paper we revisit the definition of innocent strategy from a new point of view. We use categorical ideas both in the large to construct the category $\mathcal{INN}$ of games and innocent strategies, and in the small to control the combinatorics justifying this construction.

Our aim is to derive the original notion of innocent strategy together with its basic properties from structure on a primitive category $\mathcal{G}$ of games. We do not consider linear notions of innocence. One such is the basis for Ludics [11] (see [10] for an explanation in more standard game theoretic terms). Linear innocence is independently the subject of ongoing research. There is an abstract treatment of it in terms of homotopies [26], but we do not make any connection with that here.

The idea of innocence does not appear in our initial category $\mathcal{G}$, and the definition and the elementary properties of that category are familiar and intuitive. We give a fresh treatment in Section 3 for completeness and also to stress the combinatorial content of the category. The novelties are largely a matter of style and presentation.

Our construction of the category $\mathcal{INN}$ of games and innocent strategies involves more sophisticated categorical structure in particular a distributive law. For experts we note that the maps of $\mathcal{INN}$ appear essentially as view functions.

It is the distributive law which carries out the painful task of defining directly the composition of view functions. We maintain that one gets a good understanding of the necessary combinatorics of pointers if one sees it as justifying the relevant structure on $\mathcal{G}$. We sketch the combinatorial details in Section 4. We start however by explaining the construction of $\mathcal{INN}$ in the abstract.

2 Categorical Analysis

2.1 Distributive laws

The notion of a distributive law $\lambda : ST \to TS$ of a monad S over a monad T is due to Jon Beck [6]. Though it did not feature in Mac Lane's classic text [23] it is by now well established. An accessible concrete account in the general context of monad theory is given in [5]. An early abstract perspective is contained in [34].

In this paper we need the perhaps less familiar notion of a distributive law $\lambda : GT \to TG$ of a comonad G over a monad T, both on the same category $\mathcal{C}$.

Definition 1 *A distributive law of a comonad (G, ε, δ) over a monad (T, η, μ) on a category is a natural transformation $\lambda : GT \to TG$ satisfing the coherence conditions*

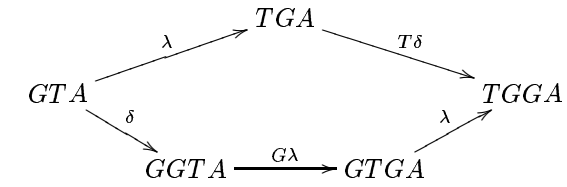
$$T\varepsilon.\lambda = \varepsilon T \quad \text{and} \quad T\delta.\lambda = \lambda G.G\lambda.\delta T$$

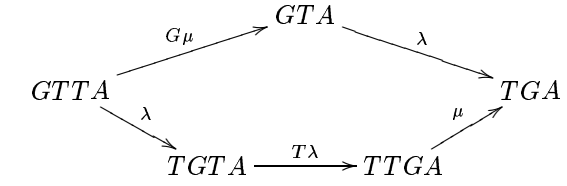
for the counit and comultipication of the comonad and

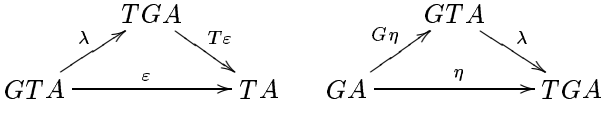
$$\lambda.G\eta = \eta G \quad \text{and} \quad \lambda.G\mu = \mu G.T\lambda.\lambda T$$

for the unit and multiplication of the monad.

So, the four diagrams below are required to commute, for every object A of the category:







Theorem 2.1 *A distributive law of a comonad G over a monad T provides both an extension $\hat{G}$ of G to the Kleisli category $\mathrm{Kl}(T)$ for T and an extension $\hat{T}$ of T to the Kleisli category $\mathrm{Kl}(G)$ for G. Moreover the Kleisli categories $\mathrm{Kl}(\hat{G})$ and $\mathrm{Kl}(\hat{T})$ are isomorphic.*

This theorem is an evident modification of Beck's ideas, but is not easy to find in the literature. We refer the reader to [33] which gives a detailed analysis of distributive laws relating monads and comonads, and treats higher-dimensional aspects of the theory. [33] also gives an indication of the range of applications in computer science. Here we need only the category $\mathrm{Kl}(\hat{G}) \cong \mathrm{Kl}((\hat{T})$, which we denote $\mathrm{Kl}(\lambda)$, and which has the following concrete and symmetric description.

- The objects are those of the original category $\mathcal{C}$.
- Maps from A to B are maps $GA \to TB$ in $\mathcal{C}$.
- The identity on A is $\eta_A.\varepsilon_A : GA \to TA$.
- The composition $f : GA \to TB$ and $g : GB \to TC$ is given by the composite $\mu_C.Tg.\lambda_B.Gf.\delta_A$.

One should understand the construction of $Kl(\lambda)$ as follows. At first sight maps $f : GA \to TB$ and $g : GB \to TC$ have no right to compose. The natural transformation λ negotiates that difficulty. But even then the definition of composition does not look associative, and it is exactly the coherence conditions which are used to establish associativity and the unit laws for composition.

2.2 The cartesian closed category

We are now in a position to outline our account of innocent strategies. We start with a symmetric monoidal closed category (see [23] for this notion) $\mathcal{G}$ of games. This is the simple category of the expository article [15]. $\mathcal{G}$ has products, and we equip it with a linear exponential comonad $(!, \varepsilon, \delta)$ making it a model of Intuitionistic Linear Logic (ILL). (For category theoretic background in this area see [7, 8, 27].) The comonad we need is not the simple one of [15], but another manifest possibility. (However we are perhaps sketching the details at this level for the first time.)

Next we give an evident monad $(?, \eta, \mu)$ related to the comonad. Then the nub of our analysis is that we shall construct a distributive law λ of ! over ?, and we define the category of games and innocent strategies in terms of it.

Definition 2 *The Kleisli category $Kl(\lambda)$ is the category $\mathcal{INN}$ of games and innocent strategies.*

$\mathcal{INN}$ has a lot of structure but here we simply indicate what we need by way of structure on $\mathcal{G}$, to show that $\mathcal{INN}$ itself is cartesian closed.

First we note the special feature that the monad ? commutes with products in our category $\mathcal{G}$. Thus we have isomorphisms

$$?1 \cong 1 \quad ?(A \times B) \cong ?A \times ?B$$

with the coherence conditions to the effect that $\eta \times \eta \cong \eta$ and $\mu \times \mu \cong \mu$. As a result we have the following.

Theorem 2.2 *The category $\mathcal{INN}$ has products.*

Proof. By the easy isomorphisms

$$\begin{array}{rcl} \mathcal{INN}(\mathcal{A}, \mathcal{B} \times \mathcal{C}) & = & \mathcal{G}(!\mathcal{A}, ?(\mathcal{B} \times \mathcal{C})) \\ & \cong & \mathcal{G}(!\mathcal{A}, ?\mathcal{B} \times ?\mathcal{C}) \\ & \cong & \mathcal{G}(!\mathcal{A}, ?\mathcal{B}) \times \mathcal{G}(!\mathcal{A}, ?\mathcal{C}) \\ & = & \mathcal{INN}(\mathcal{A}, \mathcal{B}) \times \mathcal{INN}(\mathcal{A}, \mathcal{C}). \end{array}$$

For the function space in $\mathcal{INN}$ we make use of a further piece of structure. As $\mathcal{INN}$ is a model for ILL, we have the familiar coherent isomorphisms $!1 \cong I$ and $!(A \times B) \cong !A \otimes !B$. But also for games A and B we have a game $A \multimap\!\!\boxplus B$, which should be regarded as an additive implication, and so is further additive structure. (We shall see that it corresponds to the construction of the function space arena in the standard presentation of innocence [16].) As a functor $A \multimap\!\!\boxplus B$ is covariant in B, and contravariant in A. The basic structure is then that we have a coherent natural isomorphism $?(A \multimap\!\!\boxplus B) \cong !A \multimap ?B$ which parallels $!(A \times B) \cong !A \otimes !B$.

Theorem 2.3 *The category $\mathcal{INN}$ is cartesian closed.*

Proof. Given the previous result this follows readily by the isomorphisms

$$\begin{array}{rcl} \mathcal{INN}(\mathcal{A} \times \mathcal{B}, \mathcal{C}) & = & \mathcal{G}(!(\mathcal{A} \times \mathcal{B}), ?\mathcal{C}) \\ & \cong & \mathcal{G}(!\mathcal{A} \otimes !\mathcal{B}, ?\mathcal{C}) \\ & \cong & \mathcal{G}(!\mathcal{A}, !\mathcal{B} \multimap ?\mathcal{C}) \\ & \cong & \mathcal{G}(!\mathcal{A}, ?(\mathcal{B} \multimap\!\!\boxplus \mathcal{C})) \\ & = & \mathcal{INN}(\mathcal{A}, \mathcal{B} \multimap\!\!\boxplus \mathcal{C}) \end{array}$$

Our task for the rest of this paper is to justify these simple arguments by showing that our category $\mathcal{G}$ has the necessary structure.

3 Games

3.1 Simple Games

Our games, played between P (Player) and O (Opponent), are essentially as in the literature, for example in [15].

Definition 3 *A* game A *is given by a diagram*

$$A(1) \xleftarrow{\pi} A(2) \xleftarrow{\pi} A(3) \xleftarrow{\pi} \cdots \qquad (1)$$

of sets and functions. The elements of the $A(n)$ are moves *or* positions *of the game. If $\pi(a_{n+1}) = a_n$, then a_n is* the move preceeding a_{n+1}, *while a_{n+1} is* a successor *of a_n. Thus the moves in $A(1)$ are the* initial moves. *For n odd, the moves $A(n)$ are* O-moves *played by O, or* O-positions, *where O has just played, and for n even they are* P-moves, *or* P-positions.

Thus informally a game is given by a forest of rooted trees, the roots being the initial moves played by O, and with the players, P and O, thereafter alternating moves. (As a result of the alternation, parity is a basic feature of the combinatorial bookkeeping which follows.) There are good reasons, which we do not go into here, why we do not bother to distinguish positions and moves in this context. Note in any case that the move $a \in A(n)$ determines the sequence $\mathbf{a} = (\pi^{n-1}(a), \pi^{n-2}(a), \cdots, a)$ of moves or positions which is its history. We call such a sequence a *play* in A. Occasionally we restrict sequences: $\mathbf{a}\restriction m$ is the result of restricting $\mathbf{a}$ to its first m members. The restriction of a play is a play.

3.2 Strategies

We need a notion of strategy for Player, a *P-strategy*, in a game A. We only consider deterministic strategies and we use the description of them in terms of the collection of P-positions reached by the strategy. (This description gives the forgetful functor from the category $\mathcal{G}$ to the category Rel of sets and relations see for example [17].)

Definition 4 *A* P-strategy σ *in a game A consists of a collection σ of positions, being the union of $\sigma(2n) \subseteq A(2n)$ satisfying*

- *if $x \in \sigma(2n+2)$ then $\pi^2(x) \in \sigma(2n)$;*
- *if $x, y \in \sigma(2n)$ with $\pi(x) = \pi(y)$, then $x = y$.*

Intuitively, given an O-position a' reached while playing σ, the unique P-position a in σ with $\pi(a) = a'$, if it exists, is the response according to σ. Condition (i) in the definition simply means that any P-move in σ can be reached when the game is played in accord with σ. Condition (ii) is the determinacy of strategies.

3.3 The category of schedules

Maps from A to B in the category $\mathcal{G}$ will be given by P-strategies in the linear function space $A \multimap B$, which we define in the next section. Our presentation uses some combinatorial category theory, so first we give some combinatorial motivation.

A play in $A \multimap B$ is a merge of plays in A and in B: the games are played in parallel but with the parity of A reversed (O plays the P-moves and vice-versa). Alternatively

the cogame $A^\perp$ is played in parallel with B. This traditional point of view is described in [15]. By parity a play in $A \multimap B$ must begin with an initial move in B, and whenever O plays he can do so (if at all) in the game in which P has just played, but not in the other. Thus if we let 0 indicate a play in A and 1 a play in B, then the pattern of a play of length n in $A \multimap B$ is given by a $\multimap$-*scheduling function* or *schedule* $e : \{1, \cdots, n\} \to \{0, 1\}$, that is a function satisfying $e(1) = 1$ and $e(2k+1) = e(2k)$.

Schedules $e : \{1, \cdots, n\} \to \{0, 1\}$ are sequences of 0s and 1s. We write $|e|$ for the length n of e. Let $|e|_0$ be the number of 0s and $|e|_1$ the number if 1s in the sequence: so $|e| = n = |e|_0 + |e|_1$. Note that if e is a schedule and n is odd then we must have $|e|_0$ even and $|e|_1$ odd. Also if e is a schedule so is any restriction $e{\restriction}m$.

We now give the definition of a category in which schedules are maps. In the course of the definition we introduce further notation.

Definition 5 *The* category Υ of schedules *is as follows.*

- *The objects of Υ are the natural numbers. We think of an object p as representing the totally ordered set $(p) = \{1, \cdots, p\}$. Write $(p)^+$ for the set of even elements and $(p)^-$ for the odd elements of (p).*
- *The maps in $\Upsilon(p, q)$ are the schedules e with $|e|_0 = p$ and $|e|_1 = q$. A schedule $e : p \to q$ corresponds to obvious order preserving (collectively surjective) embeddings $l : (p) \to (p+q)$ and $r : (q) \to (p+q)$ and thus to order relations $l(x) < r(y)$ from $(p)^+$ to $(q)^+$ and $r(y) < l(x)$ from $(q)^-$ to $(p)^-$.*
- *The identity in $\Upsilon(p, p)$ is the* copy-cat function *c of length $2p$, the schedule with $c(2k+1) \neq c(2k+2)$. The induced orders are $\leq$ on $(p)^+$ and on $(p)^-$.*
- *Let $e : p \to q$ and $f : q \to r$ be maps in Υ. Then their composite $f.e : p \to r$ is defined by taking the corresponding order relations, composing them as relations and reconstructing the function.*

We have defined the composition in Υ as the the composition of merges sketched in the Appendix to [18]. This makes it evident that composition is associative with the copy-cat functions c as identities. (Concretely c is of form $(1, 0, 0, 1, 1, 0, 0, 1, 1, 0, \cdots)$. This is the shape of so-called copy-cat interactions in game semantics. We shall see that all basic structure maps are given by this form of interaction.) There are other approaches. In particular, Υ is related to the category theorists Δ, that is the augmented category of all finite ordinals (including 0). There is a concrete discussion of this Δ as a strict monoidal category in [23]: in fact it is the free strict monoidal category generated by a monoid. There is a (free) functor $F : \Delta \to \Delta^\top$ from Δ to $\Delta^\top$ the category of ordinals with a top element $\top$, and with maps preserving $\top$. This situation is already considered in [22]. Then Υ is one of the total categories associated with F. (With our preferred conventions it is the lax colimit (see [19] of the profunctor $F^* : \Delta^\top \to \Delta$.) Thus the category of schedules Υ is one of the range of small combinatorial categories of fundamental interest in mathematics.

3.4 The category of games

We give a simple description of the linear function space $A \multimap B$.

Definition 6 *Let $(A \multimap B)(n)$ consist of the (e, a, b) where $e : p \to q$ is a schedule, $n = p+q$, $a \in A(p)$ and $b \in B(q)$. The predecessor function π is defined by*

$$\pi(e, a, b) = \begin{cases} (e{\restriction}(n-1), \pi(a), b) & \text{if } e(n) = 0, \\ (e{\restriction}(n-1), a, \pi(b)) & \text{if } e(n) = 1. \end{cases}$$

The strategies in $A \multimap B$ are the maps $A \to B$ in our category $\mathcal{G}$. We give the formal definition.

Definition 7 *The category $\mathcal{G}$ of games is given as follows.*

- *The objects of $\mathcal{G}$ are games.*
- *The maps in $\mathcal{G}(A, B)$ are strategies in $A \multimap B$.*
- *The identity $\iota_A : A \to A$ is given by the positions in $A \multimap A$ of the form (c, a, a) with c copy-cat.*
- *The composition of $\sigma : A \to B$ and $\tau : B \to C$ is $\tau.\sigma : A \to C$ given by the positions $(f.e, a, c)$ such that there exist $(e, a, b) \in \sigma$ and $(f, b, c) \in \tau$.*

As an immediate consequence of the fact that Υ is a category one has the following.

Proposition 3.1 *Games and strategies form a category $\mathcal{G}$.*

It is worth noting at once that while the collection of maps in $\mathcal{G}$ is complicated and hard to analyse, the collection of isomorphisms is much less so. As one might hope, an isomorphism in $\mathcal{G}$ between A and B is determined by an isomorphism at the level of their defining diagrams as given in (1) above.

3.5 Symmetric monoidal closed structure

We start by exhibiting our category $\mathcal{G}$ of simple games as a symmetric monoidal closed category. We already have seen the beginnings of data for a closed structure (for this notion see [9]) on $\mathcal{G}$, but with $\mathcal{G}$ we are lucky. We can define a good tensor product $A \otimes B$ in $\mathcal{G}$ by playing A and B in parallel with no change of parity. If again we let 0 indicate a play in A and 1 a play in B, then the pattern of a play of length n in $A \otimes B$ is given by a $\otimes$-*scheduling function* $e : \{1, \cdots, n\} \to \{0, 1\}$, that is, a function e satisfying $e(2k+1) = e(2k+2)$.

Definition 8 *Let $(A \otimes B)(n)$ consist of the (e, a, b) where e is a $\otimes$-scheduling function, $|e| = n$, $a \in A(|e|_0)$ and $b \in B(|e|_1)$. The predecessor π is defined by*

$$\pi(e, a, b) = \begin{cases} (e\restriction(n-1), \pi(a), b) & \text{if } e(n) = 0, \\ (e\restriction(n-1), a, \pi(b)) & \text{if } e(n) = 1. \end{cases}$$

The empty game I is evidently a unit for this tensor product, and it is very simple combinatorics to check directly that $A \otimes B \multimap C \cong A \multimap (B \multimap C)$. Also it is straightforward to define a symmetry for the tensor. This gives the following.

Theorem 3.2 *With structure as described above, $\mathcal{G}$ is a symmetric monoidal closed category.*

Thus $\mathcal{G}$ models multiplicative Intuionistic Linear Logic.

3.6 The standard decomposition

The additive structure on $\mathcal{G}$ is straightforward. Given games $(A_i)_{i \in I}$ the levelwise sum or coproduct

$$\textstyle\sum_i A_i(1) \xleftarrow{\sum_i \pi_i} \sum_i A_i(2) \xleftarrow{\sum_i \pi_i} \sum_i A_i(3) \cdots$$

of the defining diagrams provides a product in $\mathcal{G}$. Thus the category $\mathcal{G}$ has products. Of course the empty product 1 is also the empty game: $1 = I$ and $\mathcal{G}$ is an affine model of multiplicative-additive Intuitionistic Linear Logic.

We can use a little fine structure. Every game A is evidently isomorphic to the product over its initial moves of games with just one initial move; and it is useful to have this decomposition in a standard form. So we note a special case of the construction $A \multimap B$. There is a game S with just one opening move for Opponent and no further moves. For any game B the game $B \multimap S$ must open with the unique move in S and after that the play is in B but with the obvious change of parity: the roles of the players reversed as the even (Player) moves of $B \multimap S$ are the odd (Opponent) moves in B. Clearly any game C with just one opening move (i.e. with $C(1)$ a singleton) is isomorphic to one of form $B \multimap S$. It follows that up to isomorphism every game A is a product $A \cong \prod_a A_a \multimap S$ of games of the form $A_a \multimap S$, the product being taken over the set $A(1)$ of the opening moves of A. We shall refer to $A \cong \prod_a (A_a \multimap S)$ as the *standard decomposition* of A.

4 The exponentials

4.1 Pointers and heaps

The exponential in $\mathcal{G}$ which we are going to introduce requires what is effectively some combinatorics of trees. We start with some general considerations. For any n, pointers on a sequence of length n are given by a function $\phi : \{1, \cdots, n\} \to \{0, \cdots, n-1\}$ such that $\phi(i) < i$ for all $1 \leq i \leq n$. (The value 0 is a dummy: one could instead deal with partial functions. In what follows if a value appears undefined it should be taken to be 0.)

For each n the collection of functions giving pointers as above is in obvious bijective correspondence with (order-reversed) heaps on the set $\{0, \cdots, n\}$. (Recall that a heap is a tree-based data structure in which elements appear in order down any branch.) To obtain the heap $\mathrm{hp}(\phi)$ corresponding to a pointer function ϕ, one places 0 at the root and attaches the is with $\phi(i) = 0$; then the js with $\phi(j) = i$ are attached to i, and so on. We introduce terminology for this tree structure. For any i there is a *ϕ-branch* $(0, \cdots, \phi^2(i), \phi(i), i)$ from the root 0 to i; if we delete the root, which is dummy and does not index a move, we get the *ϕ-thread* of i which is best thought of backwards as $(i, \phi(i), \phi^2(i), \cdots)$. Later the nodes from $\{1, \cdots, n\}$ will be decorated with moves from some game, and we shall then talk of the thread of such moves with the obvious meaning.

For ϕ, ψ functions giving pointers on n we write $\phi \succeq \psi$ just when for all k, $\psi(k)$ lies in the ϕ-thread of k: intuitively this means that ψ goes faster than ϕ.. Then $\phi \succeq \psi$ just when the ψ-thread of any k lies in the ϕ-thread of k. The point then is that in the obvious sense ψ restricts to (what is up to renumbering) a pointer function on any ϕ-thread. In terms of heaps, $\phi \succeq \psi$ holds if and only if there is an order preserving bijection $hp(\psi) \to hp(\phi)$. Evidently then $\succeq$ is a partial order. The predecessor function π is the maximal element and the constant 0 function the minimal element with respect to this order. (Our overloading of π and predecessor should cause no confusion.)

4.2 Parity pointer functions

We shall only be concerned with functions giving pointers which change parity: that is for which ϕ takes even numbers to odd and vice versa. (The obvious example is the predecessor π.) The heap perspective is so important to us that we shall refer to our most important pointer functions as heaps.

Definition 9 *A* (parity) pointer function ϕ on n *is a parity-reversing function $\phi : \{1, \cdots, n\} \to \{0, \cdots, n-1\}$ with $\phi(i) < i$ for all $1 \leq i \leq n$. ϕ is an* O-heap *just when we always have $\phi(2k) = 2k - 1$. Dually ϕ is a* P-heap *just when we always have $\phi(2k+1) = 2k$.*

If ϕ is an O-heap and ψ a P-heap, we write (ϕ, ψ) for the pointer function $(\phi, \psi)(k) = \phi(k)$ if k odd, else $\psi(k)$. Obviously any pointer function is uniquely of the form (ϕ, ψ) where ϕ is an O-heap and ψ a P-heap. ϕ is an O-heap if and only if $\phi = (\phi, \pi)$, so all the information is at the O-positions; and dually for P-heap. Only π is both an O-heap and a P-heap.

For $n \in \Upsilon$, let $\mathrm{Ohp}(n)$ be the partially ordered set of O-heaps and $\mathrm{Php}(n)$ that of P-heaps. We are going to show inter alia that $\mathrm{Ohp}(n)$ is a contravariant functor and $\mathrm{Php}(n)$ a covariant functor in n.

It seems best (and will be useful later) to consider at once a quite general situation. Suppose that $e : p \to q$ is a schedule, ϕ an O-heap on q and ψ a P-heap on p. Recall that e gives injections $l : (p) \to (p+q)$, $r : (q) \to (p+q)$. We define the O-heap (ϕ, e, ψ) on $p+q$ by setting

$$(\phi, e, \psi)(k) = \begin{cases} r(\phi(j)) & \text{if } k = r(j) \text{ is odd,} \\ l(\psi(i)) & \text{if } k = l(i) \text{ is odd,} \\ k-1 & \text{otherwise.} \end{cases}$$

Now for a schedule $e : p \to q$ in Υ and O-heap ϕ on q, we define the O-heap $e^*\phi$ on p, using the threads of the O-heap (ϕ, e, π) on $p+q$. We set $e^*\phi(k) = j$ just when j is maximal with $j < k$ and $l(j)$ in the (ϕ, e, π)-thread of $l(k)$, if there is such, 0 otherwise. It is evident that (with the obvious disambiguation) (π, e, π) is π so that $e^*\pi = \pi$. Also if $\phi \succeq \phi'$, then then $(\phi, e, \pi) \succeq (\phi', e, \pi)$ and it follows that $e^*\phi \succeq e^*\phi'$. Thus e^* is a map of posets preserving the top element. Similarly given $e : p \to q$ and ψ a P-heap on p, we define $e_*\psi$ using (π, e, ψ). We set $e_*\psi(k) = j$ just when j is maximal with $j < k$ and $r(j)$ in the (π, e, ψ)-thread of $r(k)$, if there is such, 0 otherwise. Again we have a map of posets preserving the top element. We write Pos for the usual category of posets and order preserving maps.

Proposition 4.1 *With the structure indicated above, both* $\mathrm{Ohp} : (\Upsilon)^{\mathrm{op}} \to \mathrm{Pos}$ *and* $\mathrm{Php} : (\Upsilon) \to \mathrm{Pos}$ *are functorial: for copycat* $c : p \to p$ *in* Υ, $c^*\phi = \phi$ *and* $c_*\psi = \psi$*; for* $e : p \to q$ *and* $f : q \to r$ *in* Υ, $e^*f^*\phi = (f.e)^*\phi$ *and* $f_*e_*\psi = (f.e)_*\psi$.

We also need some information concerning threads of O-heaps of the form (ϕ, e, ψ). Schedules $e : p \to q$ and $f : q \to r$ give identifications of (q) in $(p+q)$ and in $(q+r)$ with parity preserved in the first case and reversed in the second. Given subsequences (threads) in $(p+q)$ and $(q+r)$, which agree on the image of (q) we can extend the notion of composition of schedules to give a composed subsequence in $(p+r)$.

Proposition 4.2 *Suppose* $e : p \to q$ *and* $f : q \to r$ *are schedules,* ϕ *an O-heap on* r *and* ψ *a P-heap on* p*. Then threads for* $(\phi, f.e, \psi)$ *on* $p+r$ *are composites of unique threads for* $(\phi, f, e_*\psi)$ *on* $q+r$ *and for* $(f^*\phi, e, \psi)$ *on* $p+q$.

4.3 The exponential functor

In effect $!A$ is a game in which O can perform backtracking with repetitions. The definition is as follows.

Definition 10 *Given a game* A*, the game* $!A$ *is determined as follows. The positions of* $!A$ *are of the form* $(\phi, \mathbf{a})$ *where*

- ϕ *is an O-heap on* n *and* $\mathbf{a}$ *is a sequence of moves from the game* A *of length* n*;*
- *for any thread of* ϕ*, the moves in* $\mathbf{a}$ *corresponding to the thread form a play in* A*.*

The predecessor is $\pi(\phi, \mathbf{a}) = (\phi\restriction(n-1), \mathbf{a}\restriction(n-1))$.

The first non-trivial issue is to show that ! extends to a functor: given $\sigma : A \to B$ we have to define $!\sigma : !A \to !B$. The rough intuition is that in playing $!A \multimap !B$ according to $!\sigma$, there is a current thread in $A \multimap B$ determined by the fact that O can backtrack in B, and P responds according to σ in this thread. As a result P backtracks in A simply in response to O's doing so in B.

Definition 11 *Let* σ *be a strategy in* $A \multimap B$*. A position* $(e, (\phi, a), (\psi, b))$ *of* $!A \multimap !B$ *is in* $!\sigma$ *just when* $\phi = e^*\psi$ *and the moves in any thread of* (ϕ, e, π) *are moves played in accord with* σ.

On can understand this definition as follows. I is a coalgebra for the comonad !, and so we can promote σ to give a strategy $\sigma^\dagger$ in $!(A \multimap B)$. (That is where the O-heap (ϕ, e, π) lives.) Now it follows from the isomorphisms (2) that ! is a monoidal functor in the lax sense. This gives us a map $!(A \multimap B) \to !A \multimap !B$, and composing that with $\sigma^\dagger$ gives $!\sigma$.

The main result relying on the composition of threads is the functoriality in the following.

Theorem 4.3 *If* σ *is a P-strategy in* $A \multimap B$*, then* $!\sigma$ *is a P-strategy in* $!A \multimap !B$*. Further* ! *is functorial:* $!\iota_A = \iota_{!A}$ *and* $!(\tau.\sigma) = !\tau.!\sigma$.

Proof: The functoriality follows from Proposition 4.2 and the fact that $e^*\pi = \pi$.

4.4 The linear exponential comonad

We claim that ! is the functor part of a linear exponential comonad (in the terminology of [17]). The main issue is the comonad structure as the tight form of Seely's conditions (see [8]) are easy to check. To understand the situation, we need a concrete understanding of plays in games like $!!A$ and even $!!!A$. Clearly in $!!A$ there must be two pointers corresponding to the two !s; and in fact that together with a sequence from A is all the data needed. The issue is the relation between the pointers.

Now we give a representation of the plays in $!!A$. They are given by $(\phi, \psi, \mathbf{a})$ where

- ϕ and ψ are O-heaps and $\phi \succeq \psi$;

- the moves along the ψ-threads are plays from A.

(It is then automatic that the restriction of ψ makes each ϕ-thread a $!A$-play.) In terms of this representation we need a description for $\sigma : A \to B$ of $!!\sigma : !!A \to !!B$. With the above conventions, the strategy $!!\sigma$ consists of the

$$(e, (\phi, \psi, \mathbf{a}), (\phi', \psi', \mathbf{b}))$$

with $e^*\phi' = \phi$, $e^*\psi' = \psi$ and each (ψ', e, π)-thread a play in σ.

We now give the comonad structure.

- The P-strategy ε_A in $!A \multimap A$ is given by positions of the form $(c, (\pi, \mathbf{a}), a)$ with c copy-cat, π as ever predecessor and a the last move of the play $\mathbf{a}$.
- The P-strategy δ_A in $!A \multimap !!A$ is given by positions of the form $(c, (\psi, \mathbf{a}), (\phi, \psi, \mathbf{a}))$ again with c copy-cat, and with $\phi \succeq \psi$.

Proposition 4.4 *ε_A and δ_A are natural in A.*

Proof: We give some flavour of the arguments at this stage of the theory.
First for ε, take $\sigma : A \to B$. To get $\sigma.\varepsilon_A$ we compose positions of form $(c, (\pi, \mathbf{a}), a)$ with those of form (e, a, b) from σ: that gives positions $(e, (\pi, \mathbf{a}), b)$ where if a is the last move in $\mathbf{a}$, then (e, a, b) is in σ. For $\varepsilon_B.!\sigma$ we compose positions of the form $(e, (\phi, \mathbf{a}), (\psi, \mathbf{b}))$ above with $(c, (\pi, \mathbf{b}), b)$. So we have to identify ψ with π. But then $e^*\pi = \pi$ so we get the same result as for $\sigma.\varepsilon_A$.
The argument for δ is similar. Both $!!\sigma.\delta_A$ and $\delta_B.!\sigma$ give positions $(e, (\phi, \mathbf{a}), (\psi, \chi, \mathbf{b}))$, where $e^*\chi = \phi$ and each (χ, e, π)-thread gives a play in σ.

The comonad equations are equally straightforward checks.

One sees that $!\varepsilon_A$ in $!A \multimap !!A$ consists of the $(c, (\phi, \mathbf{a}), (\phi, \phi, \mathbf{a}))$, while clearly $\varepsilon_{!A}$ consists of the $(c, (\phi, \mathbf{a}), (\pi, \phi, \mathbf{a}))$ in each case with c copycat. One immediately deduces $!\varepsilon_A.\delta_A = \iota_{!A} = \varepsilon_{!A}.\delta_A$.

Plays in $!!!A$ are given by $(\rho, \phi, \psi, \mathbf{a})$ satisfying the obvious extension of the condition for $!!A$. Then again with a little work $!\delta_A$ consists of the $(c, (\phi, \psi, \mathbf{a}), (\rho, \phi, \psi, \mathbf{a}))$ while clearly $\delta_{!A}$ consists of the $(c, (\rho, \psi, \mathbf{a}), (\rho, \phi, \psi, \mathbf{a}))$. Hence we have $!\delta_A.\delta_A = \delta_{!A}.\delta_A$, both composites consisting of $(c, (\psi, \mathbf{a}), (\rho, \phi, \psi, \mathbf{a}))$.

Finally the two coherent isomorphisms

$$!1 \cong I \quad \text{and} \quad !(A \times B) \cong !A \otimes !B \tag{2}$$

are easy. So we get the expected result.

Theorem 4.5 *$(!, \varepsilon, \delta)$ is a linear exponential comonad.*

At this point we have given $\mathcal{G}$ the structure of a model for full Intuitionistic Linear Logic. So we stress that we are not here concerned with the usual Kleisli category for this situation.

4.5 The monad

The monad $(?, \eta, \mu)$ arises by playing games but now allowing P to backtrack. We shall need an account paralleling that for ! and we sketch that at once.

Definition 12 *Given a game A, the game $?A$ is determined as follows. The positions of $!A$ are the $(\phi, \mathbf{a})$ where*

- *ϕ is a P-heap on n and $\mathbf{a}$ is a sequence of moves from A of length n;*
- *for any thread of ϕ, the moves in $\mathbf{a}$ corresponding to the thread form a play in A.*

The predecessor is $\pi(\phi, \mathbf{a}) = (\phi\restriction(n-1), \mathbf{a}\restriction(n-1))$.

Experts will recognise at once that if they regard A as an arena, then strategies in $?A$ are essentially view functions. That is, they are representing functions for innocent strategies over A in the old terminology of [16]. Since $?(A \multimap\!\!\boxplus B) \cong !A \multimap ?B$, it is exactly these view functions which give the maps in $\mathcal{INN}$.

Functoriality of ? needs attention. Again there are intuitions about threads, but now O forces the backtracking in the (dual) version $A^\perp$ of A.

Definition 13 *Let σ be a strategy in $A \multimap B$. A position $(e, (\phi, a), (\psi, b))$ of $?A \multimap ?B$ is in $?\sigma$ just when $\psi = e_*\psi$ and the moves in any thread of (π, e, ψ) are moves played in accord with σ.*

Again a little work is needed to establish the following.

Theorem 4.6 *If σ is a P-strategy in $A \multimap B$, then $?\sigma$ is a P-strategy in $?A \multimap ?B$. Furthermore $?$ is functorial: $?\iota_A = \iota_{?A}$ and $?(\tau.\sigma) = ?\tau.?\sigma$.*

We now give the monad structure.

- The P-strategy η_A in $A \multimap ?A$ is given by positions of the form $(c, a, (\pi, \mathbf{a}))$ with c copy-cat, a the last move of the play $\mathbf{a}$ and π predecessor.
- The P-strategy μ_A in $??A \multimap ?A$ is given by positions of the form $(c, (\phi, \psi, \mathbf{a}), (\psi, \mathbf{a}))$ again with c copy-cat, and with $\phi \succeq \psi$.

The arguments for naturality and the monad laws are straightforward variants of earlier arguments

Theorem 4.7 *$(?, \eta, \mu)$ is a monad on $\mathcal{G}$.*

At this point we observe that there is a definition of $?$ on objects in terms of the standard decomposition. We have $?(\prod_a A_a \multimap S) \cong \prod_a !A_a \multimap S$. In the same vein, up to isomorphism,

$$\eta_A = \prod_{a \in A(1)} \varepsilon_{A_a} \multimap S \quad \text{and} \quad \mu_A = \prod_{a \in A(1)} \delta_{A_a} \multimap S\,.$$

From this we get a quick treatment of the further special structure relating to ?.

From the internal definition of ? and the standard decomposition we see that

$$\begin{array}{rcl} ?(A \times B) & \cong & ?(\prod_a A_a \multimap S \times \prod_b B_b \multimap S) \\ & = & \prod_a (!A_a \multimap S) \times \prod_b (!B_b \multimap S) \\ & = & ?(\prod_a A_a \multimap S) \times ?(\prod_b B_b \multimap S) \\ & \cong & ?A \times ?B \end{array}$$

as required: the isomorphisms $\eta_{A\times B} \cong \eta_A \times \eta_B$ and $\mu_{A\times B} \cong \mu_A \times \mu_B$ are immediate.

We give an easy definition of $A \multimap\!\!\boxplus B$ using the standard decomposition. We set $A \multimap\!\!\boxplus B = \prod_b (A \times B_b \multimap S)$. (A moment's reflection shows that this is essentially the construction of the function space arena from the original treatment [16]. (Functoriality in B needs a moments thought, but we do not really need it.) Now we can readily calculate

$$\begin{array}{rcl} ?(A \multimap\!\!\boxplus B) & = & ?(\prod_b A \times B_b \multimap S \\ & \cong & \prod_b !(A \times B_b) \multimap S \\ & \cong & \prod_b !A \otimes !B_b \multimap S \\ & \cong & \prod_b !A \multimap (!B_b \multimap S) \\ & \cong & !A \multimap \prod_b (!B_b \multimap S) \\ & = & !A \multimap ?B \end{array}$$

as required.

4.6 The distributive law

Now at last we come to an explanation of the distributive law. We need first to understand plays in the games $!?A$ and $?!A$, along the lines of the analysis of $!!A$ above. Again they will be given by positions $(\phi, \chi, \mathbf{a})$ where $\phi \succeq \chi$, but with general mixed pointer functions we have to say exactly what χ is. Recall that any pointer function χ is uniquely of the form (ϕ, ψ) with ϕ an O-heap and ψ a P-heap.

The plays in $!?A$ are given by $(\phi, \chi, \mathbf{a})$ where

- ϕ is an O-heap, $\chi = (\phi, \psi)$ for some P-heap ψ and $\phi \succeq \chi$;
- the moves along the χ-threads are plays from A.

The plays in $?!A$ are given dually by $(\psi, \chi, \mathbf{a})$ where

- ψ is an P-heap, $\chi = (\phi, \psi)$ for some O-heap ϕ and $\psi \succeq \chi$;
- the moves along the χ-threads are plays from A.

The correctness of this analysis depends on some abstract results which we record here.

Proposition 4.8 *Let $e : p \to q$ in Υ, let ϕ be an O-pointer function on q, and ψ a P-pointer function on p. Then*

- *If $e^*\phi \succeq (e^*\phi, \psi)$ then $\phi \succeq (\phi, e_*\psi)$.*
- *If $e_*\psi \succeq (\phi, e_*\psi)$ then $\psi \succeq (e^*\phi, \psi)$.*

We explain the connection with the language of [16]. In the case of $!?A$, the condition $\phi \succeq (\phi, \psi)$ means that in the game played with the pointers (ϕ, ψ), P plays innocently, that is, his pointers are all in his view. This corresponds to the fact that the restriction of $\chi = (\phi, \psi)$ makes each ϕ-thread a $?A$-play. Dually for $?!A$ it is O who is constrained to play innocently and then, the restriction of $\chi = (\phi, \psi)$ makes each ψ-thread a $!A$ play.

Now we give our main definition.

Definition 14 *The strategy λ_A in $!?A \multimap ?!A$ consists of all positions of the form $(c, (\phi, (\phi, \psi), \mathbf{a}), (\psi, (\phi, \psi), \mathbf{a}))$.*

In λ_A we have $\chi = (\phi, \psi)$ with $\phi \succeq \chi$ and $\psi \succeq \chi$. In the standard terminology of [16], the plays with pointers from the pointer function χ are then exactly the ones in which both players play innocently. Thus λ_A identifies what are called in [16] the legal plays in the two games $!?A$ and $?!A$.

To establish naturality of λ, we need descriptions in terms of our representations of maps $!?\sigma : !?A \to !?B$, and $?!\sigma : ?!A \to ?!B$. Recall that for a schedule $e : p \to q$, ϕ a O-heap on q and ψ a P-heap on p, we defined the O-heap (ϕ, e, ψ) on $p + q$. Then using our conventions, the strategy $!?\sigma$ consists of

$$(e, (\phi, (\phi, \psi), \mathbf{a}), (\phi', (\phi', \psi'), \mathbf{b}))$$

with $e^*\phi' = \phi$, $e_*\psi = \psi'$ and each (ϕ', e, ψ) thread giving a play in σ. The strategy $?!\sigma$ consists of

$$(e, (\psi, (\phi, \psi), \mathbf{a}), (\psi', (\phi', \psi'), \mathbf{b}))$$

with the same conditions.

We now give the final piece of combinatorial information needed to justify the analysis of Section 2.

Theorem 4.9 *λ_A gives a distributive law $\lambda : !? \to ?!$.*

Proof: The arguments are similar to those given earlier. By way of example we indicate why we have

$$?\delta.\lambda = \lambda!.!\lambda.\delta? \,.$$

We use presentations of the plays in complex games as follows.

- $!!?A$ has as positions the $(\chi, \phi, (\phi, \psi), \mathbf{a})$ with both χ, ϕ O-heaps, ψ a P-heap and with $\chi \succeq \phi \succeq (\phi, \psi)$;
- $!?!A$ has as positions the $(\chi, (\chi, \psi), (\phi, \psi), \mathbf{a})$ with χ , ϕ O-heaps, ψ a P-heap and $\chi \succeq (\chi, \psi) \succeq (\phi, \psi)$;
- $?!!A$ has as positions the $(\psi, (\chi, \psi), (\phi, \psi), \mathbf{a})$ with χ , ϕ O-heaps, ψ a P-heaps and $\psi \succeq (\chi, \psi) \succeq (\phi, \psi)$.

Then we confirm that both $?\delta_A.\lambda_A$ and $\lambda_{!A}.!\lambda_A.\delta_{?A}$ have as positions all the

$$(c, (\phi, (\phi, \psi), \mathbf{a}), (\psi, (\chi, \psi), (\phi, \psi), \mathbf{a})) \,.$$

5 Conclusions

We have sketched an approach to the most basic category $\mathcal{HON}$ of pointer games, disciplined by some serious category theory. While our approach is not shorter than the original hands on approach, it is less ad hoc. In particular what we do demystifies the nature of arenas. After all concretely they are themselves games, albeit games which are not that interesting in their own right. In particular we backwards engineer the function space arena as exactly what is required to produce a function space in the Kleisli category.

We have demonstrated that the definition of composition of innocent strategies and the proof that this composition is associative have two components. There is some categorical combinatorics to establish a distributive law λ and there is the categorical construction of the Kleisli category $\mathrm{Kl}(\lambda)$. The distributive law is an entirely new feature, not part of Linear Logic.

Establishing the distributive law demands a systematic approach to the elementary combinatorics of pointers, and we use categorical ideas also there. This treatment captures computational ideas in a clean way. For example the contravariant transport of O-heaps by the functor e^* has to do with properties of independence. As such it is related to recent work [28] of Melliès and his student Samuel Mimram in his asynchronous setting.

Overall our approach is an interplay of categorical combinatorics. While we have only given the basic structure of $\mathcal{INN}$, we believe that our treatment extends readily to further structure and related categories of pointer games.

References

[1] S. Abramsky, D. R. Ghica, A. S. Murawski, C.-H. L. Ong and I. D. R. Stark. Nominal games and full abstraction for the nu-calculus. In: *Proceedings of Nineteenth Annual Symposium in Logic in Computer Science*, IEEE Computer Society Press, 2004, 150–159.

[2] S. Abramsky, D. R. Ghica, A. S. Murawski and C.-H. L. Ong. Applying Game Semantics to Compositional Software Modeling and Verification. In *Proceedings of 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 2988, Springer-Verlag, 2004, 421–435.

[3] S. Abramsky and G. McCusker. Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions. In P. W. O'Hearn and R. D. Tennent, eds., *Algol-like languages Vol. 2*. Birkhaüser, 1997, 297-330.

[4] S. Abramsky, K. Honda, and G. McCusker. A fully abstract game semantics for general references. In *Proceedingsof Thirteenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1998, 334–344.

[5] M. Barr and C. Wells. *Toposes, Triples, and Theories*. Grundlehren der math. Wissenschaften 278, Springer-Verlag, 1985.

[6] J. Beck. Distributive laws. In: B. Eckman ed., *Seminar on Triples and Categorical Homolgy Theory*, Lecture Notes in Mathematics, 80, Springer, Berlin, 1969, 119–140.

[7] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. Linear λ-Calculus and Categorical Models Revisited. In: *Proceedings Computer Science Logic 1992*, Lecture Notes in Computer Science 702, Springer-Verlag, 1993, 61–84.

[8] G. Bierman. What is a categorical model of intuitionistic linear logic? In: *Proceedings of the Second International Conference on Typed Lambda Calculus and Applications*, Lecture Notes in Computer Science, 902, Springer Verlag, 1995, 73–93.

[9] S. Eilenberg and G.M. Kelly. Closed Categories. In: *Proceedings of Conference on Categorical Algebra, La Jolla 1965*, 1966, 421–562.

[10] C. Faggian and M. Hyland. Designs, Disputes and Strategies. In: J. Bradfield, ed., *Computer Science Logic*, Lecture Notes in Computer Science 2471, Springer-Verlag, 2002, 442-457.

[11] J.-Y. Girard. Locus Solum. *Mathematical Structures in Computer Science*, 11, 2001, 301-506.

[12] R. Harmer and O. Laurent. The anatomy of innocence revisited. In: S. Arun-Kumar and N. Garg, eds., *Foundations of Software Technology and Theoretical Computer Science (FSTTCS '06)*, Lecture Notes in Computer Science, 4337, Springer-Verlag, 2006, 224-235. .

[13] R. Harmer and G. McCusker. A fully abstract game semantics for finite nondeterminism. In: *Proceedingsof Fourteenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1999.

[14] D. Hughes. *Hypergame Semantics: Full Completeness for System F*. D. Phil. Dissertation, Oxford University, 2000.

[15] M. Hyland. Game semantics. In: A. Pitts, P. Dybjer eds., *Semantics and Logics of Computation*, Publications of the Newton Institute, Cambridge University Press, Cambridge, 1997, 131-184.

[16] M. Hyland and L. Ong. On full abstraction for PCF, I, II and III. *Information and Computation*, 163, 2000, 285-408.

[17] M. Hyland and A. Schalk. Abstract Games for Linear Logic. Extended Abstract. In *Proceedings of Category Theory and Computer Science '99*, Electronic Notes in Theoretical Computer Science, 29, 1999, 26 pp.

[18] M. Hyland and A. Schalk. Games on Graphs and Sequentially Realizable Functionals. Extended Abstract. In: *Proceedingsof Seventeenth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 2002, 257–264.

[19] G. M. Kelly. Elementary observations on 2-categorical limits. *Bulletin of the Australian Mathematical Society*, 39, 1989, 301-317.

[20] J. Laird. Full abstraction for functional languages with control. In: *Proceedings of Twelth Annual Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1997, 58–67 .

[21] O. Laurent. Classical isomorphisms of types. *Mathematical Structures in Computer Science*, 15, 2005, 969-1004.

[22] F. W. Lawvere. Ordinal sums and equational doctrines. In: *Seminar on Triples and Categorical Homology Theory*, Lecture Notes in Mathematics 80, Springer-Verlag, 1969, 141-155.

[23] S. Mac Lane. *Categories for the Working Mathematician*, Graduate Texts in Mathematics, 2nd ed., Springer Verlag, 1998.

[24] G. McCusker. Games and definability for **FPC**. *The Bulletin of Symbolic Logic*, 3, 1997, 347–362.

[25] G. McCusker. *Games for Recursive Types*. BCS Distinguished Dissertation Series, Cambridge University Press, Cambridge 1999.

[26] P.-A. Melliès. Asynchronous games 2. The true concurrency of innocence. Extended abstract in: *Proceedings of the 15th International Conference on Concurrency Theory*, Lecture Notes in Computer Science 3170, Springer-Verlag 2004, 448-465.

[27] P.-A. Melliès. Categorical models for linear logic revisited. To appear in *Theoretical Computer Science*.

[28] P.-A. Melliès and S. Mimram, Asynchronous games: innocence without alternation. Submitted, April 2007.

[29] A. S. Murawski, C.-H. L. Ong and I. Walukiewicz. Idealized Algol with Ground Recursion, and DPDA Equivalence. In *Proceedings 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)*, Lecture Notes in Computer Science, 3580, Springer-Verlag, 2005, 917-929.

[30] H. Nickau. *Hereditarily Sequential Functionals: A Game-Theoretic Approach to Sequentiality*. Ph.D. Thesis, Universität-Gesamthochschule-Siegen, 1996.

[31] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes (extended abstract). In: *Proceedings of 21st Symposium on Logic in Computer Science*, IEEE Society Press, 2006, 81-90.

[32] C.-H. L. Ong. Some results on a game-semantic approach to the verification of infinite structures. In: Z. Ésik, ed., *Proceedings of 20th International Workshop and 15th Annual Conference of the EACSL*, Lecture Notes in Computer Science, 4207, 2006, 31-40.

[33] A.J. Power and H. Watanabe. Combining a Monad and a Comonad. *Theoretical Computer Science*, 280, 2002, 137–162.

[34] R. Street. The Formal Theory of Monads. *Journal of Pure and Applied Algebra*, 2, 1972, 149–168.