Innocent Game Models of Untyped λ-Calculus

Andrew D. Ker, Hanno Nickau and C.-H. Luke Ong

Oxford University Computing Laboratory, Parks Road, Oxford. OX1 3QD

Abstract

We present a new denotational model for the untyped λ-calculus, using the techniques of game semantics. The strategies used are *innocent* in the sense of Hyland and Ong [9] and Nickau [17], but the traditional distinction between "question" and "answer" moves is removed. We first construct models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ as global sections of a reflexive object in the categories $\mathbb{A}$ and $\mathbb{A}_{\mathrm{REC}}$ of arenas and innocent and recursive innocent strategies respectively. We show that these are sensible $\lambda\eta$-algebras but are neither extensional nor universal.

We then introduce a new representation of innocent strategies in an *economical form*. We show a strong connexion between the economical form of the denotation of a term in the game models and a *variable-free form* of the Nakajima tree of the term. Using this we show that the definable elements of $\mathcal{D}_{\mathrm{REC}}$ are precisely what we call *effectively almost-everywhere copycat (EAC) strategies*. The category $\mathbb{A}_{\mathrm{EAC}}$ with these strategies as morphisms gives rise to a $\lambda\eta$-model $\mathcal{D}_{\mathrm{EAC}}$ which we show has the same expressive power as D_∞, i.e. the equational theory of $\mathcal{D}_{\mathrm{EAC}}$ is the maximal consistent sensible theory $\mathcal{H}^*$. We show that the model $\mathcal{D}_{\mathrm{EAC}}$ is sensible, order-extensional and universal (i.e. every strategy is the denotation of some λ-term). To our knowledge this is the first syntax-free model of the untyped λ-calculus with the universality property.

Key words: Game semantics, innocent strategies, untyped λ-calculus.

Contents

1 Introduction

1.1 Overview

In this paper we present a new denotational model for untyped λ-calculus, using the techniques of game semantics. A game model of the lazy λ-calculus was given by Abramsky and McCusker in [2] using the history-free strategies of [1]. The particular variety of game used here is a fairly simple modification of that in [9] for a fully abstract model of PCF – the distinctions of "question" and "answer" are not needed.

In Section 2 we present some standard definitions, rephrased to take this into account. To begin with, we define an *arena*, which details what moves may be made in the game, and the notion of *well-formed sequence*, which (by restricting the possible traces of moves) sets out some ground rules. The games are played between two imaginary people called Opponent and Proponent who must alternate moves, and each move must be *justified* by some preceding move. These ideas are familiar from the games literature, and the constructions of function and product arenas are standard.

During a game there is a notion of the *view* of the play up to that point, a subsequence of the moves played so far. This is the "computationally relevant" part, and we enforce a further rule on the game by the definition of *legal position*, which says that moves may be justified only by relevant preceding moves.

We assume that the players are operating according to some *strategy*, which we present initially as all possible traces of moves which that player will engage in. The definition of strategies allows only for deterministic play. A condition on strategies called *innocence* means that a player may not take into account any irrelevant preceding moves. Furthermore, strategies may be "composed", if they are strategies for arenas of appropriate type. All of these definitions are standard.

Another familiar idea is that of an *innocent function*, which is the presentation used in [17]. Instead of considering a strategy to be the set of all possible sequences of moves, we define a function telling a player how to react in every possible situation. Innocent strategies work particularly simply in this form because we define the function to map from views to (justified) moves and no more conditions are required, and we show how one may transfer simply from innocent functions to strategies and vice versa.

In the same fashion as other games literature, we take arenas as objects and innocent strategies on function space arenas as morphisms to make a category – we call it $\mathbb{A}$. As one would hope, this is a cartesian closed category.

In Section 3 we discover the first model for untyped λ-calculus, because there is an object U – in some sense the maximal countable arena – with the property that $U = U \Rightarrow U$ (true equality, not just isomorphism). As is well known, such an object will give rise to a $\lambda\eta$-algebra, which in this case we call $\mathcal{D}$ (or, if we restrict our attention to recursive strategies, $\mathcal{D}_{\text{REC}}$). The elements of $\mathcal{D}$ are the innocent strategies on U, and composition of the elements is defined in terms of composition of strategies. We formulate a method of approximation: the approximants to a strategy are those which only play within a specified finite part of the arena U. We show that this satisfies what Barendregt terms the "basic equations" for approximants, which lead in a totally standard way to the fact that the model is *sensible*, i.e. that all unsolvable terms have the same denotation.

However the model has some undesirable properties too. There are many undefinable elements (in particular, all the nontrivial finite approximants to terms are undefinable) and the model is not *extensional* (that is, there are distinct elements of $\mathcal{D}$ which have the same applicative behaviour). The undefinable elements are really to blame for the lack of extensionality (and a related property, not having *enough points*).

In order to improve the model we seek in Section 4 a way to characterise the definable elements. To do this we look at both strategies and terms in a different light. Strategies could be written in *economical form*, an encoding of the innocent function which deletes all redundant information. Terms are considered as *Nakajima trees* – this is the intuitive extension of a Böhm tree to an infinite η-expansion. We can encode Nakajima trees in a *variable-free form*, a somewhat technical definition but in essence just a way to write down an infinite tree (where each label is an infinite list of abstractions and a single head variable) as a function from sequence of natural numbers to pairs of integers. The major result of this section is what we call the Exact Correspondence Theorem: the denotation of a closed term, in economical form, coincides precisely with the labelling function for the variable-free form of its Nakajima tree.

In Section 5 we make use of this, using classical results which characterise those Böhm-like trees that are Böhm trees of terms, transforming these into Nakajima trees and then into the language of economical forms of innocent strategies. The result is a new class of innocent strategies, *effectively almost-everywhere copycat (EAC) strategies*, which (almost by construction) are precisely the elements of $\mathcal{D}$ which are definable. Since identity and projection strategies are easily seen to be EAC, it remains to show that arenas and EAC strategies, as a subcategory of $\mathbb{A}$, still form a CCC which we call $\mathbb{A}_{\mathrm{EAC}}$. This category still has the reflexive object U so we can identify a new λ-algebra $\mathcal{D}_{\mathrm{EAC}}$, which is both sensible and universal (every strategy is the denotation of a term). Consideration of the local structure shows that all three games models, namely $\mathcal{D}$, $\mathcal{D}_{\mathrm{REC}}$ and $\mathcal{D}_{\mathrm{EAC}}$, equate terms precisely when they are equated by Scott's D_∞ model. Thus the theory generated by denotational equality is $\mathcal{H}^*$, the maximal consistent sensible λ-theory. (A very similar result has also been obtained by Di Gianantonio, Franco and Honsell recently [7]. Using history-free strategies, in the sense of Abramsky, Jagadeesan and Malacaria, they have constructed games models of the untyped λ-calculus which all induce the same theory $\mathcal{H}^*$.) In contrast to $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, $\mathcal{D}_{\mathrm{EAC}}$ is order-extensional (from which weak extensionality follows easily, and so it is a λ-model) and universal. To our knowledge, other than term models, the universality result is the first of its kind.

Finally, in Section 6 we outline some further work which allows the model to be refined, so that η-conversion is not validated. Details of the construction, which yield a universal model whose equational theory is precisely that of Böhm-tree equality, will be presented in a sequel.

1.2 Prerequisites

We assume familiarity with the untyped λ-calculus for which the standard reference is [3]. Particularly vital topics include the notions of solvability and head normal forms, Böhm trees, standard theories and models.

Basic category theory, up to CCCs and adjunctions, is also assumed – see for example [14]. Some references are made to computability. A knowledge of the standard literature on Hyland/Ong/Nickau games ([9,17,15]) would motivate many of the definitions, but is not required.

1.3 Notation

We describe some conventions:

- The set $\{1, 2, 3, \ldots\}$ is written $\mathbb{N}$, and $\mathbb{N}_0$ is $\mathbb{N} \cup \{0\}$.
- The set of all finite sequences of elements from Σ is written Σ^*. Sequences are written $\langle s_1, s_2, \ldots, s_n \rangle$.
- The empty sequence is denoted by ε.
- Sequences are usually written $\vec{s}$ to distinguish sequences from elements. Sometimes we do not follow this convention.
- Concatenation of sequences is denoted $\vec{s} \cdot \vec{t}$. This notation is also overloaded so that, for example, $m \cdot \vec{s}$ means $\langle m \rangle \cdot \vec{s}$.
- The length of the sequence $\vec{s}$ is written $|\vec{s}|$.
- The usual order on sequences will be prefix. The subsequence pre-order is written $\preceq$.
- We will use $f; g$ for composition in categories. The terminal object will be denoted $\mathbf{1}$.
- A Σ-labelled tree is a (possibly infinitary) tree with nodes labelled with elements from Σ.
- A partially Σ-labelled tree is a tree with nodes labelled with elements from $\Sigma \cup \{\perp\}$, such that any node labelled $\perp$ has no descendants. A node labelled $\perp$ is considered to be part of the tree, but is without a label.

- A *Böhm-like tree* is, informally, a partially-labelled tree with labels of the form $\lambda x_1 \ldots x_n.y$. A formal definition can be found in [3, 10.1.12]

2 Game Arenas and Innocent Strategies

We build on the dialogue games of Hyland and Ong [9] and Hanno Nickau [17] (christened H_2O games by Girard, for **H**yland, **H**anno and **O**ng) using a variant in which there is no sense of question or answer. We present the details from scratch, and, although useful, no prior knowledge of the above references is required.

To begin with we describe what we mean by a "game", defining the notion of *arena*, which details the moves of the game, and *legal position*, which sets out some rules for the game. We introduce the notion of a *strategy* for the two participants of the game, and the property of strategies called *innocence*, and thus define the categories $\mathbb{A}$ (and $\mathbb{A}_{\mathrm{REC}}$) of arenas and (recursive) innocent strategies, and show that they are cartesian closed. Both contain the same reflexive object (for which the retraction morphisms are an isomorphism), leading to $\lambda\eta$-algebras $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$. Some analysis of the models shows that they are both sensible.

2.1 Arenas, Views and Legal Positions

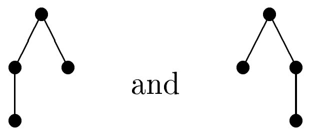
The abstract idea of a "game" is based on an arbitrary set of *moves*. The game is "played" between two players called Proponent and Opponent, and the mathematical objects we study are the possible sequences of moves. The moves come in a structure called an *arena*, which specifies that certain moves may not be played until certain others have been (that is, may not appear in a sequence of moves unless certain others appear earlier in the sequence). The arena also specifies that each move may be made only by either Proponent or Opponent, and another rule imposed on the sequences of moves is that the two players must alternate, with Opponent playing first.

This idea is as standard in [9,17,15], except that in these references each move is labelled as a question or an answer, and there are additional rules controlling the interplay of questions and answers. We think of our moves as "declarations", or answerless questions, and indeed the definitions given here do correspond with the standard ones, with all moves considered to be questions. They are sometimes presented in a slightly different manner for clarity or because simplifications can be made.

Definition 1 An *arena* is a finite tuple of nonempty trees of *moves*. The root of each tree is called an *initial move*.

We emphasise that the moves are just an arbitrary set, which are given some extra structure to become an arena. Regardless of what the moves are, we will label them in a uniform way below, in a way which reflects their arrangement in the arena, and never refer to them except by label.

Remark 2 Our trees will be considered (and illustrated) "upside-down" with the root at the top, rather more like family trees than the botanical kind. Following this analogy, we can refer to a *child* of a node, and say that one node *inherits from* another, with the obvious meanings. Moreover, we assume that the children of a node come with an ordering and that this distinguishes them, thus we can unambiguously talk about "the second child of the root node". This means that we do not consider the trees



to be the same, since in the first only the first child of the root has a child, and in the second only the second child of the root has a child.

The standard definition of [9] is in terms of a *forest* (partial order with each upper set a finite chain) and it is clear that our tree structure, ordered by inheritance and with the forests put together, determines such an order. However we wish to make sure that the trees making up the forest come in a specified order. Further we will only require finitely many trees in each forest (which allows for a simple construction of products later on).

We will only be interested in countably branching, countably deep trees. Thus we can encode each tree of the arena as a subset of $\mathbb{N}^*$ by inductively labelling the root as ε and the n^{th} child of the move $\vec{s}$ as $\vec{s} \cdot n$. Hence each move of each tree is associated uniquely with a sequence of natural numbers. Conversely, given any subset $A \subseteq \mathbb{N}^*$ which is prefix-closed and has the property that whenever $\vec{s} \cdot n \in A$ we have $\vec{s} \cdot m \in A$ for each $m \leq n$, we can form an arena of one tree where the moves are the elements of A, with tree structure given by prefix. If we represent trees by subsets of $\mathbb{N}^*$, we say that we are representing arenas in *sequence-subset* form. **In this work, we will interchangeably talk about arenas in either forest-of-trees or sequence-subset form.**

Each arena is of the form $\langle A_1, \ldots, A_n \rangle$, where each $A_i \subseteq \mathbb{N}^*$. We say that an arena is *single-tree* if $n = 1$. Most of our intuition is based on single-tree arenas, and many definitions are given only for this type of arena with the generalisation to multiple-tree arenas left to the reader.

Example 3 There are three important arenas which will be referred to in this work:

(1) E is the arena consisting of no trees at all; in sequence-subset form it is represented by $\langle\rangle$.
(2) M is the "minimal" single-tree arena, consisting of one tree of one node; in sequence-subset form it is represented by $\langle\{\varepsilon\}\rangle$.
(3) U is the "maximal" single-tree arena, consisting of one tree which is countably branching with every path countably deep; in sequence-subset form it is represented by $\langle\mathbb{N}^*\rangle$.

We say that moves at an even depth of the trees (including the roots which are at depth 0) are *O-moves*, the moves made by the Opponent, and moves at an odd depth are *P-moves*, made by Proponent. O-moves are often denoted by $\bullet$ and P-moves by $\circ$. The *polarity* of a move refers to whether it is a P- or O-move, and we may talk about *swapping the polarity* of a set of moves.

We list some properties of, and a relation between, arenas for later use.

Definition 4 (1) A *sub-arena* of an arena $A = \langle A_1, \ldots, A_n\rangle$ is an arena $B = \langle B_1, \ldots, B_n\rangle$ (which therefore has the same number of trees as A) with each $B_i \subseteq A_i$.
(2) An arena is *finitely branching* if every tree in it is finitely branching.
(3) An arena is *recursive* if, as a subset of $\mathbb{N}^*$, each tree in it is a recursive set (i.e. membership is decidable).
(4) An arena is *recursively enumerable* (or just r.e.) if, as a subset of $\mathbb{N}^*$, each tree in it is recursively enumerable (i.e. membership is semi-decidable).

Arenas can be formed from other arenas by two major constructions.

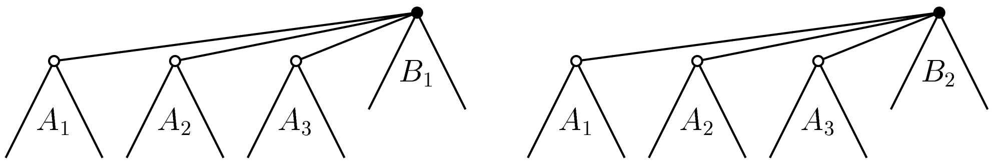
Definition 5 Suppose that $A = \langle A_1, \ldots, A_m\rangle$ and $B = \langle B_1, \ldots, B_n\rangle$ are arenas.

(1) The *product arena* $A \times B$ is the "disjoint union" of the trees of A and B, the concatenation of their tuples. Formally $A \times B = \langle A_1, \ldots, A_m, B_1, \ldots, B_n\rangle$.
(2) The *function space arena* $A \Rightarrow B$ is constructed as follows: the initial moves of $A \Rightarrow B$ are those of B; to each tree below each such initial move, we graft onto it a copy of A. More precisely, $A \Rightarrow B = \langle C_1, \ldots, C_n\rangle$ where

$$\begin{aligned} C_i = \{\varepsilon\} \ &\cup\ \{a \cdot \vec{s} \mid 1 \leq a \leq m \ \wedge\ \vec{s} \in A_a\} \\ &\cup\ \{(a+m) \cdot \vec{s} \mid a \cdot \vec{s} \in B_i\}. \end{aligned}$$

We illustrate the construction of a function space arena, when the arenas in question are not single-tree. Suppose that $A = \langle A_1, A_2, A_3\rangle$ and $B = \langle B_1, B_2\rangle$.

Then $A \Rightarrow B$ could be pictured as:



This picture shows how the arena A is duplicated in $A \Rightarrow B$, when B is not single-tree. It also illustrates how we will draw arbitrary arenas — the triangle with a single symbol $\circ$ or $\bullet$ at the top indicates some tree with the root a P- or O-move respectively.

We note that any arena can be decomposed as the product of finitely many single-tree arenas. Moves of product and function arenas may be referred to as, for example, A-moves and B-moves, depending on which part of the composite arena they lie in. Notice that the polarity of A-moves has been swapped in $A \Rightarrow B$.

We have already seen that the way the moves are arranged in the trees of the arena determines whether each is played by Proponent or Opponent. The tree structure of the arena also determines which moves are prerequisite for each move — the idea is that a move may not be played until after its immediate ancestor in the arena has been played, and we might call a move's immediate ancestor its "enabling move". It is possible that there might be more than one instance of a move's enabling move, and for reasons which will be illustrated later it is important to identify one of them as the enabling move which was "used". For this we need the following definition.

Definition 6 A *justified sequence* of an arena A is a sequence of moves of which each element except the first is equipped with a pointer to some previous move. We call the pointer a *justification pointer* and if the move m^- is pointed to by m we say that m^- *justifies* m.

We say that a move m^- in a justified sequence *hereditarily justifies* m if one can reach m^- from m by repeatedly following justification pointers.

Remark 7 For the definition it suffices to say that the justification pointers exist. To give this a proper mathematical meaning we can encode a justified sequence as a sequences of pairs, the sequence of first components being the moves and the second components being natural numbers, such that if m is justified by m^- in a sequence $\vec{s} \cdot m^- \cdot \vec{t} \cdot m \cdot \vec{u}$ then the number paired with the move m is $|\vec{t}|/2$. Note that since each move is justified by a move made by the opposite player, and players alternate, the sequence $\vec{t}$ is forced to be of even length. By convention the initial move is paired with the number zero. (Here the sequences $\vec{s}$ and $\vec{t}$ are sequences of moves, each move of which are, sometimes confusingly, represented by sequences of natural numbers.) There are other possible encodings of justification pointers, but this particular encoding matches that used in the definition of *economical form* in Section 4.1.

To avoid tedious detail, in practice we ignore this encoding, and typeset justified sequences pictorially with lines linking each move to the one move which justifies it. An example of how this looks can be found in the next section. When we define functions on well-formed sequences which involve manipulation of pointers we will not bother to explain the details of how the encodings are manipulated as a result, which in any case ought to be obvious.

We may now impose restrictions on the possible sequences of moves made in the game, as we indicated above.

Definition 8 A *well-formed sequence* over A is a justified sequence $\vec{s}$ which has the following properties:

Initial Move: The first element of $\vec{s}$ is an initial move of A, an O-move.
Alternation: Thereafter elements of $\vec{s}$ alternate between P-moves and O-moves.
Justification: If m is justified by m^- then the move m is directly beneath m^- in the tree of the arena.

Note that the last of these forces each move (except the first) to be justified by a move of the opposite polarity. The definitions for dialogue games involve an additional condition called *well-bracketing*. This is redundant for declaration-only arenas. Note that, as a consequence of the definition of justified sequence and the condition of justification, the first move may not be repeated in a well-formed sequence.

Well-formed sequences will be the permissible sequences of moves in the games played on the arena. One could think of these conditions as imposing some basic ground rules.

Remark 9 Since Opponent can play any of the initial moves of A, and all subsequent moves are justified (and hence cannot be roots of trees of A), Opponent has chosen which of the trees of A the rest of the sequence is to be played in. An idea presented in [17] is to label the roots of the tree A_i in A as ε_i, and then for any well-formed sequence we can see which tree it comes from by examining the subscript of the first move. In practice we almost always examine single-tree arenas (decomposing multiple-tree arenas in the product of single-tree arenas if necessary) and so we shall not need to use this convention.

Since initial moves cannot be justified by any move, and only the first move of a justified sequence is not justified, it is possible to repeat the initial moves of A in the arena $A \Rightarrow B$ but not in the arena A. This is a sort of "hidden !" of Linear Logic, which is made explicit in McCusker's presentation [15]. Our presentation is simpler but this also means that we must be aware of the possibility of moves becoming repeatable in function spaces.

In a well-formed sequence, some moves are considered "not relevant" to the player making the next move. We suppose that each time a player makes a move m he is really only interested in the next move (made by his opponent) *which is justified by* m. Once such a move is made, he is supposed to ignore any intervening moves. This will be made precise in the definition of *innocence* in the next section, but what follows is the definition of the relevant moves of a sequence.

Definition 10 The *P-view* of a justified sequence $\vec{s}$, written $\ulcorner \vec{s} \urcorner$, is given recursively by:

$$\begin{aligned}
\ulcorner \varepsilon \urcorner &= \varepsilon && \text{for initial moves } \varepsilon \\
\ulcorner \vec{s} \cdot m \urcorner &= \ulcorner \vec{s} \urcorner \cdot m && \text{for } m \text{ a P-move} \\
\ulcorner \vec{s} \cdot m^- \cdot \vec{t} \cdot m \urcorner &= \ulcorner \vec{s} \urcorner \cdot m^- \cdot m && \text{for } m \text{ an O-move justified by } m^-
\end{aligned}$$

The *O-view*, $\llcorner \vec{s} \lrcorner$, is given analogously by:

$$\begin{aligned}
\llcorner \varepsilon \lrcorner &= \varepsilon && \text{for initial moves } \varepsilon \\
\llcorner \vec{s} \cdot m \lrcorner &= \llcorner \vec{s} \lrcorner \cdot m && \text{for } m \text{ an O-move} \\
\llcorner \vec{s} \cdot m^- \cdot \vec{t} \cdot m \lrcorner &= \llcorner \vec{s} \lrcorner \cdot m^- \cdot m && \text{for } m \text{ a P-move justified by } m^-
\end{aligned}$$

It may be the case that for a P-move m justified by an O-move m^- in some justified sequence $\vec{s}$, the move m^- is deleted in $\ulcorner \vec{s} \urcorner$ (similarly if m is an O-move then its justifying move might be deleted in the O-view.) Thus the O/P-view of a justified sequence may not itself be a justified sequence. This motivates the following:

Definition 11 A *legal position* of an arena A is a justified sequence $\vec{s}$ satisfying the following *visibility condition*:

For each non-initial O-move m justified by m^-, say $\vec{s} = \vec{t_1} \cdot m^- \cdot \vec{t_2} \cdot m \cdot \vec{t_3}$, we have that $m^- \in \llcorner \vec{t_1} \cdot m^- \cdot \vec{t_2} \cdot m \lrcorner$. Similarly all P-moves are justified by O-moves appearing in the P-view up to that point.

This gives the required property:

Lemma 12 *If $\vec{s}$ is a legal position then so are $\ulcorner \vec{s} \urcorner$ and $\llcorner \vec{s} \lrcorner$.*

PROOF. Essentially a straightforward induction. See [9, 4.2.5] or [15]. □

By *a P-view* of an arena A we mean a justified sequence which is the P-view of some legal position of A.

There are other important properties of views, legal positions and function space arenas. Only the statements of the major results are given here, and proofs (in our setting when there is no sense of question or answer) carry over from those of the similar results described in [9, §4.4] or [15].

Lemma 13 (View Characterisation) *A justified sequence of an arena A is the P-view of some legal position if and only if it is well-formed and every non-initial O-move is justified by the immediately preceding P-move. The same statement is true with all polarities swapped.*

Lemma 14 (View Idempotency) *If $\vec{s}$ is a legal position then $\llcorner\llcorner\vec{s}\lrcorner\lrcorner = \llcorner\vec{s}\lrcorner$ and $\ulcorner\ulcorner\vec{s}\urcorner\urcorner = \ulcorner\vec{s}\urcorner$.*

Definition 15 If $\vec{s}$ is a legal position of a function space arena $A \Rightarrow B$, with a an initial move of A occurring in $\vec{s}$, we define the following:

The *B-component* of $\vec{s}$, written $\vec{s} \upharpoonright B$, is the projection of $\vec{s}$ onto B i.e. the subsequence formed by taking only the moves in B, together with their justification pointers.

The *(A,a)-component* of $\vec{s}$, written $\vec{s} \upharpoonright (A, a)$, is the projection of $\vec{s}$ onto the moves of A which are hereditarily justified by a, together with justification pointers.

In order to make the latter into a well-formed sequence we write $\vec{s} \upharpoonright (A, a)^+$ for $b \cdot (\vec{s} \upharpoonright (A, a))$ where b is the initial B-move justifying a.

Note that we have to identify A-components by their initial move, due to the "hidden !".

Lemma 16 (Projection Convention) *(1) $\vec{s} \upharpoonright B$ is a legal position in B. (2) $\vec{s} \upharpoonright (A, a)$ is a legal position in A.*

2.2 Strategies and Composition

Informally, a strategy is information which tells one of the players which move to make next, or not to make a next move, in any given situation. We define a strategy as the set of all possible sequences of moves which the player is prepared to see played.

Definition 17 A *P-strategy* σ for a single-tree arena A consists of a nonempty prefix-closed subset of legal positions of A subject to:

Determinacy: If $\vec{s}\cdot m \in \sigma \wedge \vec{s}\cdot m' \in \sigma$ for P-moves m and m' then $\vec{s}\cdot m = \vec{s}\cdot m'$, i.e. these moves and their justification pointers are the same.

Contingent Completeness: If $\vec{s} \cdot m \in \sigma$ for a P-move m and $\vec{s} \cdot m \cdot m'$ is a legal position of A then $\vec{s} \cdot m \cdot m' \in \sigma$.

An *O-strategy* is defined analogously. However we are more often interested in P-strategies which we will usually just refer to as strategies.

For a general arena $A = \langle A_1, \ldots, A_n \rangle$ a P-strategy is an n-tuple of P-strategies, one for each tree (and we refer to these n P-strategies as the *component* strategies). However, an O-strategy is just a single O-strategy on one of the trees, together with information which selects that tree. This difference is in view of remark 9.

The property of determinacy means that, given any legal position after which it is Proponent's turn to play (i.e. the last move was made by Opponent), either Proponent may not make a reply or their move is uniquely determined. Contingent Completeness means that all of Opponent's possible next moves are part of the strategy (although this is not to say that Proponent need reply to them).

Remark 18 The standard definitions of strategy ([9,17,15]) do not need to consider multiple-tree arenas as a special case, since usually the initial moves of the trees can be referred to under distinct names. A P-strategy is then just a prefix-closed set of legal positions, subject to the rules above, which must also contain every initial move of the arena (and an O-strategy is the same except that it contains at most one initial move of the arena).

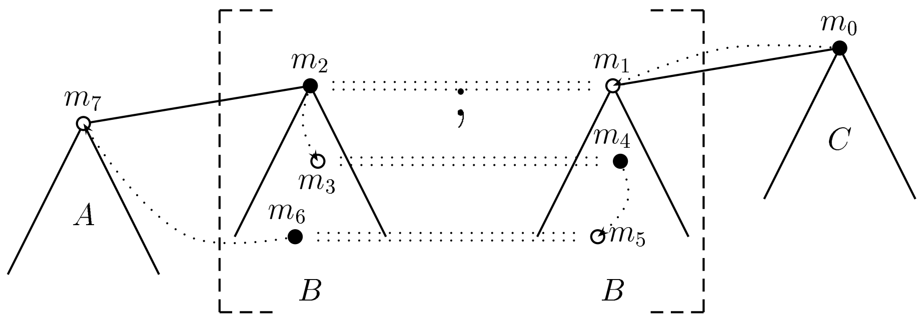
However we prefer to keep to the terminology that an initial move is labelled ε, and add some special cases to the definitions to cope with multiple-tree arenas.

Example 19 The simplest strategy is defined by the set $\{\varepsilon\}$ i.e. the only legal position in it is ε. This is a strategy on any single-tree arena, and is denoted $\perp$. We call this the *undefined strategy* or the *empty strategy* (even though it is not actually an empty set).

Definition 20 We can define the *play* of a P-strategy σ against an O-strategy τ to be the sequence of moves generated by them both. By the rules of contingent completeness and determinacy, $\sigma \cap \tau$ contains one (finite or infinite) chain of legal positions (ordered by prefix as always). The join of this chain is the play.

If we have strategies σ and τ on arenas $A \Rightarrow B$ and $B \Rightarrow C$ respectively then we can form their composite strategy $\sigma;\tau$ on $A \Rightarrow C$. Informally we do this by identifying O/P-moves of the B component of $A \Rightarrow B$ with P/O-moves of the B component of $B \Rightarrow C$, and then hiding all the moves in B. This is reminiscent of CSP's "parallel composition" and "hiding" operators. We describe this in formal detail only briefly, as details which carry directly to declaration games can be found in [9]. We first illustrate the ideas with an example.

Example 21 Suppose that we have strategies σ on $A \Rightarrow B$ and τ on $B \Rightarrow C$, for single-tree arenas A, B and C. We picture a possible computation of the first move of the composite strategy $\sigma;\tau$ below.



This diagram illustrates the computation of the first P-move of the strategy $\sigma;\tau$. Suppose that the first move of the strategy σ in response to the initial O-move m_0 is the P-move m_1, in the arena $B \Rightarrow C$. The way composition is computed, since the move m_1 occurs in the arena B it is duplicated as the O-move m_2 in the arena $A \Rightarrow B$, where it is the initial O-move seen by the strategy τ. We suppose that τ's response to this initial move is the P-move m_3, which is the B-component of the arena $A \Rightarrow B$. This move is duplicated as the O-move m_4 in the arena $B \Rightarrow C$. The strategy σ has now seen the sequence $\langle m_0, m_1, m_4 \rangle$ (each non-initial move was justified by the immediately preceding move) and we suppose that its response is the P-move m_5, in the B-component of the arena $B \Rightarrow C$. This is duplicated as an O-move in the arena $A \Rightarrow B$. The strategy τ has now seen the sequence $\langle m_2, m_3, m_6 \rangle$ (again each non-initial move was justified by the immediately preceding move) and this time we suppose that its response is the P-move m_7, which is in the A-component of the arena $A \Rightarrow B$. Since this move is not in the hidden arena B, it is the first visible move of the composition after the initial move m_0. Thus we have calculated that the strategy $\sigma;\tau$, a strategy on the arena $A \Rightarrow C$, makes the move m_7, the root of the C-component, in response to the initial move m_0.

The formal definitions are as follows.

Definition 22 Suppose we are given two arenas $A \Rightarrow B$ and $B \Rightarrow C$. We say that $\vec{s}$ is an *interaction sequence* of A, B and C and write $\vec{s} \in \mathrm{ias}(A, B, C)$ if the following conditions hold:

(1) $\vec{s}$ is made up of moves from the arenas A, B and C, and each move except the first is equipped with a justification pointer to some previous move;
(2) $\vec{s} \upharpoonright (B, C)$, the subsequence of moves in $\vec{s}$ in B or C with the polarity of moves of B swapped to resemble $B \Rightarrow C$, is a legal position of $B \Rightarrow C$;
(3) $\vec{s} \upharpoonright (A, B, b)$, the subsequence of moves in the $A \Rightarrow B$ component hereditarily justified by b with the polarity of moves of A swapped, is a legal position of $A \Rightarrow B$, for all initial B-moves $b \in \vec{s}$.
(4) $\vec{s} \upharpoonright (A, C)$, the subsequence of all moves in A and C, with the polarity of moves in A swapped, and with pointers from A to C via B renamed as pointers directly from A to C, is a legal position of $A \Rightarrow C$.

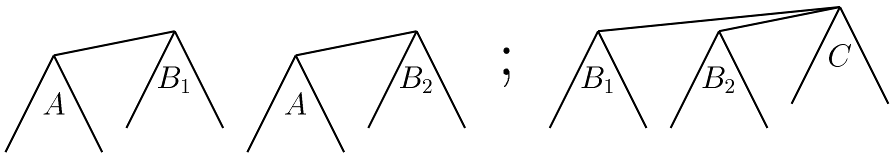
For strategies $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ define the composite strategy as follows:

$$\begin{aligned}\sigma;\tau = \{\, & \vec{s} \upharpoonright (A, C) \mid \vec{s} \in \mathrm{ias}(A, B, C) \ \wedge \\ & \vec{s} \upharpoonright (A, B, b) \in \sigma \text{ for all initial } B\text{-moves } b \in \vec{s} \ \wedge \ \vec{s} \upharpoonright (B, C) \in \tau \,\}\end{aligned}$$

An essentially straightforward result, although very technical in proof, is that composition is well-defined and associative. For a proof see [9, 5.1.3].

Lemma 23 *(1) If σ and τ are strategies on $A \Rightarrow B$ and $B \Rightarrow C$ respectively then $\sigma;\tau$ is a strategy on $A \Rightarrow C$.*
(2) If, in addition, ρ is a strategy on $C \Rightarrow D$, then $(\sigma;\tau);\rho = \sigma;(\tau;\rho)$

Remark 24 For arenas with multiple trees, one must sometimes identify copies of the same arena in the composite strategy. For example, if we have strategies σ and τ on $A \Rightarrow (B_1 \times B_2)$ and $(B_1 \times B_2) \Rightarrow C$ respectively, for single-tree arenas A, B and C, then the composition $\sigma;\tau$ will look like:



and moves made in either copy of A will appear in the single copy of A in the composite strategy on $A \Rightarrow C$.

As we commented in Section 2.1, we suppose that both players only notice the "relevant" previous moves when playing their next move. Exactly which moves were relevant to Proponent was made precise in the definition of P-view, and we now enforce the property that Proponent's strategies may only take this relevant information into account.

Definition 25 A P-strategy σ on a single-tree arena is *innocent* if for odd-length legal positions $\vec{s}$ and $\vec{t}$ and P-moves m,

$$\vec{s} \cdot m \in \sigma \ \wedge \ \vec{t} \in \sigma \ \wedge \ \ulcorner \vec{s} \urcorner = \ulcorner \vec{t} \urcorner \quad \Longrightarrow \quad \vec{t} \cdot m \in \sigma \ \wedge \ \ulcorner \vec{s} \cdot m \urcorner = \ulcorner \vec{t} \cdot m \urcorner$$

i.e. P's next move, and its justification, at each stage depends only on the P-view up to that point.

A P-strategy on a multiple-tree arena is innocent if each of its components is innocent in the above sense.

For more motivation of why innocence is important see [9, §7.5]. Put briefly, innocent strategies have a certain amount of extensionality enforced on them. A key result, which is technically difficult to prove (for details see [9, §5.3]) is that composition of innocent strategies is well-defined:

Lemma 26 *If σ is an innocent strategy on $A \Rightarrow B$ and τ an innocent strategy on $B \Rightarrow C$ then $\sigma;\tau$ is an innocent strategy on $A \Rightarrow C$.*

If we recall the definition of a strategy, the property of determinacy means that strategies can be thought of as functions mapping legal positions ending in O-moves to Proponent's next move. The rule of contingent completeness then fills in the rest of the detail, which is that positions ending in P-moves can be followed by any legal O-move.

Furthermore, the property of innocence means that the move, and its justification, made by Proponent depends only on the P-view of the legal position preceding it, so if a strategy is considered as a function its value depends only on the P-view of its argument. Recall that by *a P-view* of an arena A we mean a justified sequence which is the P-view of some legal position of A; then an innocent strategy on A could be represented by a function mapping P-views of A to *justified P-moves*, i.e. P-moves equipped with a justification pointer back into the argument of the function.

Definition 27 For an innocent strategy on a single-tree arena σ we can define the *innocent function* of σ, written f_σ, which is a partial map from P-views which end in O-moves to justified P-moves, by $f(\ulcorner\vec{s}\urcorner) = m$, with a pointer from m to the move m^- in $\vec{s}$, if $\vec{s} \cdot m \in \sigma$ and m is justified by m^-.

We can define the innocent function of an innocent strategy on a multiple-tree arena by forming a tuple of the innocent functions of the components of the strategy.

The reader will find it easy to verify that the empty strategy $\perp$ is trivially innocent, and that its innocent function is the everywhere undefined function.

Remark 28 In order to make this a proper mathematical function we need to encode the pointer of a justified P-move. We could follow the encoding in Remark 7, then the justified P-move is a pair consisting of a P-move and a natural number which says how many pairs of moves one has to go back from the end of the argument of the innocent function. Indeed, the presentations given in [9] and [17] used encodings of justified P-moves from the start. However we prefer to avoid encoding the pointers when typesetting innocent functions, leaving encodings for Section 4.1.

A function constructed in this way is called *innocent* and we can formalise such functions.

Definition 29 We say that f is *an innocent function* on a single-tree arena A if f maps P-views in A with P to move to justified P-moves, and the following conditions hold:

(1) $\mathrm{dom}(f)$ is prefix-closed,
(2) If $\vec{s}\cdot m^-\cdot\vec{t}\cdot m\cdot m' \in \mathrm{dom}(f)$, with m justified by m^-, then $f(\vec{s}\cdot m^-\cdot\vec{t}) = m$, and m points to the move m^-,
(3) If $f(\vec{s}) = m$ then m is a child in the arena A of the move which it points to in $\vec{s}$.

Obviously, an innocent function on a multiple-tree arena consists of a tuple of innocent functions, one for each tree in the arena.

We note that a function with such domain can only encode an innocent strategy. The conditions listed are required to make the function "strategic", i.e. the set of legal positions it describes are prefix-closed, deterministic and made up of properly justified sequences of moves. These conditions are sufficient to allow us to construct a unique strategy from an innocent function as follows:

Definition 30 Given an innocent function f on a single-tree arena A we can define an innocent strategy σ_f inductively by:

(1) $\varepsilon \in \sigma_f$, where ε is the initial move of A.
(2) If $\vec{s} \in \sigma_f$, $|\vec{s}|$ is even, and $\vec{s} \cdot m$ a legal position of A then $\vec{s} \cdot m \in \sigma_f$

(3) If $\vec{s} \in \sigma_f$, $|\vec{s}|$ is odd, and $f(\ulcorner\vec{s}\urcorner)$ is defined then the sequence $\vec{s} \cdot f(\ulcorner\vec{s}\urcorner)$ is in σ_f, (where the justification pointer from the move $f(\ulcorner\vec{s}\urcorner)$ points to the move in $\vec{s}$ which projects to the move specified by f in $\ulcorner\vec{s}\urcorner$).

Further, the construction of innocent strategy from innocent function and vice versa are mutually inverse:

Lemma 31 *(1)* $f_{(\sigma_f)} = f$
(2) $\sigma_{(f_\sigma)} = \sigma$
(3) $f \subseteq f' \iff \sigma_f \subseteq \sigma_{f'}$
(4) $\sigma \subseteq \sigma' \iff f_\sigma \subseteq f_{\sigma'}$

In view of this very strong correspondence between innocent functions on A and innocent strategies on A, we use the representations of strategies as innocent functions and subsets of legal positions interchangeably. We draw attention to the difference between "the innocent function" of an innocent strategy, and "an innocent function" on an arena. Each of the former must be one of the latter.

The representation as an innocent function allows us to define the following properties of strategies cleanly.

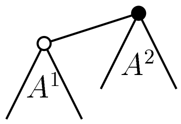
Definition 32 (1) An innocent strategy is said to be *compact* if the domain of definition of its innocent function is finite.
(2) We say that an innocent strategy is *recursive* if the innocent function representing it is (partial) recursive, after some encoding of P-views and justified P-moves.

It does not make much sense to talk about a recursive function on a domain which is not at least r.e., and so we only consider recursive strategies on r.e. arenas. This also guarantees that as a set of legal positions a recursive innocent strategy is an r.e. set. Note that the innocent function representation of an innocent strategy can be found effectively, and vice versa. Also it is clear that composition of strategies is given effectively. Hence,

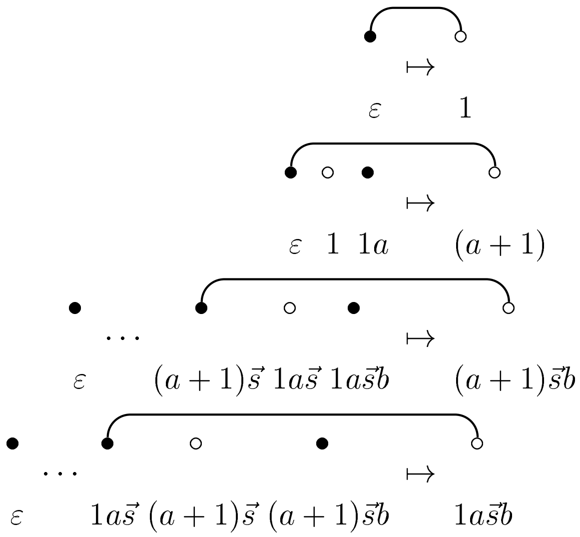
Lemma 33 *The composition of two recursive innocent strategies is itself recursive.*

The technology of innocent functions, and the representation of justification pointers as numbers, allows us to define strategies in an intuitive and typographically sensible way. (Note: This is not the only reason for their introduction!) We give an example of some innocent strategies to illustrate some of the recurring themes, to introduce our notation for innocent functions, and for use in the next section.

For any single-tree arena A we can define the *identity strategy* on $A \Rightarrow A$, which can be pictured as below.



Both A^1 and A^2 are copies of the arena A. We write id_A for the strategy which plays the P-move $\langle 1 \rangle$ in response to the initial move ε, and then in response to any O-move in the component A^1 (respectively A^2) it plays the identical move (which will be a P-move) in the component A^2 (respectively A^1). Formally, the innocent function of id_A is:

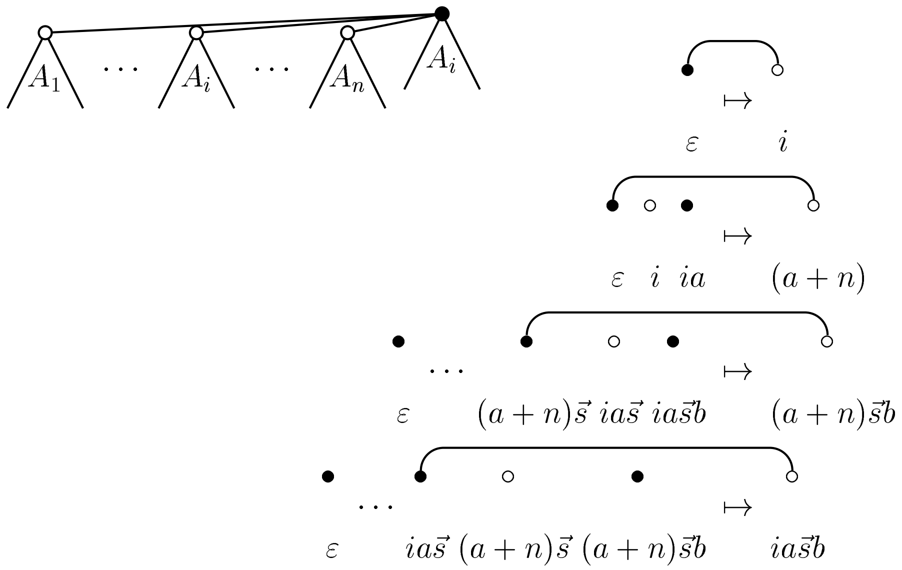


for any sequence of numbers $\vec{s}$ which is of appropriate parity of length (odd in the third clause, even in the fourth) and such that the moves $1a\vec{s}$, $1a\vec{s}b$, etc. do actually exist in the arena $A \Rightarrow A$.

We have omitted some justification pointers in the innocent function, but they can be reconstructed from the action of the function on shorter P-view, so this is unambiguous.

This is an example of a *copycat* strategy, in which P's response to any non-initial O's move is just to copy it into another tree. To generalise to multiple-tree arenas we first look at another example of a copycat strategy, which we will need in order to construct the categories we use.

For an arena $A = \langle A_1, \ldots, A_n \rangle$ and for $i = 1, \ldots, n$ we can define the *projection strategy* on the single-tree arena $A \Rightarrow A_i$ (shown below), written $\pi^A_{A_i}$, as a copycat strategy with innocent function given by:



Here $\vec{s}$ ranges over sequences of appropriate parity, a and b over positive natural numbers.

Then the *identity strategy* on any arena A is

$$\mathrm{id}_A = \langle \pi^A_{A_1}, \ldots, \pi^A_{A_n} \rangle$$

The reader is invited to verify that identities and projections work as expected.

2.4 Categories of Innocent Strategies

In this section we show that arenas and innocent strategies form a category, which is cartesian closed. We also make the corresponding definition when we restrict our attention to recursive innocent strategies.

Definition 34 The *Category of Arenas and Innocent Strategies*, $\mathbb{A}$, is defined as follows:

(1) Objects are arenas;
(2) Morphisms $f : A \to B$ are innocent strategies on the function space arena $A \Rightarrow B$;
(3) Composition of morphisms is composition as strategies;
(4) The identity morphism on A is the copycat strategy id_A.

Lemmas 23, 26 and the properties of identities show that $\mathbb{A}$ is indeed a category.

Theorem 35 $\mathbb{A}$ *is cartesian closed.*

PROOF. We define products and exponentials as follows:

(1) The terminal object $\mathbf{1}$ is the arena E (defined in Example 3, the arena consisting of no trees);
(2) The product of A and B is the product arena $A \times B$;
(3) Projections are copycat strategies. If $A = \langle A_1, \ldots, A_m \rangle$ and $B = \langle B_1, \ldots, B_n \rangle$ then define

$$\pi_A^{A\times B} : (A \times B) \to A = \langle \pi_{A_i}^{\langle A_1,\ldots,A_m,B_1,\ldots,B_n\rangle} | i \in \{1, \ldots, m\} \rangle$$
$$\pi_B^{A\times B} : (A \times B) \to B = \langle \pi_{B_i}^{\langle A_1,\ldots,A_m,B_1,\ldots,B_n\rangle} | i \in \{1, \ldots, n\} \rangle$$

(4) The exponential object $(A \Rightarrow B)$ is the function space arena $A \Rightarrow B$;
(5) Note that the arenas $(A \Rightarrow B) \times A \Rightarrow B$ and $(A \Rightarrow B) \Rightarrow (A \Rightarrow B)$ are identical. Take the evaluation morphism $\mathrm{eval}_{A,B} : (A \Rightarrow B) \times A \to B$ to be the same strategy as $\mathrm{id}_{A\Rightarrow B} : (A \Rightarrow B) \to (A \Rightarrow B)$.

We draw attention to the distinction between $\mathrm{id}_{A\Rightarrow B}$ and $\mathrm{eval}_{A,B}$ as morphisms. They are given by the same strategy, but have different domains and codomains and hence are different morphisms.

It is clear that for any arena A the arena $A \Rightarrow \mathbf{1}$ is the empty arena E and hence that there is a unique morphism $!_A : A \Rightarrow \mathbf{1}$, the "empty" strategy $\perp$ defined in Example 19. Note that $\mathbf{1} \Rightarrow A = A$.

To show that $\mathbb{A}$ is cartesian we need that for arenas A, B, C and morphisms $\sigma : A \Rightarrow B$ and $\tau : A \Rightarrow C$ we have a unique morphism $\langle \sigma, \tau \rangle$ such that $\langle \sigma, \tau \rangle \, ; \, \pi_B^{B\times C} = \sigma$ and $\langle \sigma, \tau \rangle \, ; \, \pi_C^{B\times C} = \tau$. Given the above conditions and the properties of the strategies π we can easily show that the morphism

$$\langle \sigma, \tau \rangle : A \to (B \times C) = \sigma \cdot \tau$$

works as required (recall that σ and τ will be tuples of strategies on single-tree arenas — the operation $\cdot$ is simply concatenation).

To show cartesian closure we need that for each $f : A \times B \to C$ there is a unique $\Lambda(f) : A \to (B \Rightarrow C)$ such that $f = (\Lambda(f) \times \mathrm{id}_B) ; \mathrm{eval}_{B,C}$. Note that the arenas $(A \times B) \Rightarrow C$ and $A \Rightarrow (B \Rightarrow C)$ are identical. Since $\mathrm{eval}_{B,C} = \mathrm{id}_{B\Rightarrow C}$, as strategies, it is easy to check that taking $\Lambda(f)$ as the same strategy as f is such a unique morphism. $\square$

Definition 36 The *Category of Arenas and Recursive Innocent Strategies*, $\mathbb{A}_{\mathrm{REC}}$, is the subcategory of $\mathbb{A}$ with objects restricted to r.e. arenas and morphisms restricted to recursive innocent strategies.

Lemma 33, and the trivial observation that identity and projection strategies are recursive, show that $\mathbb{A}_{\mathrm{REC}}$ is also a cartesian closed category with the same cartesian closed structure as $\mathbb{A}$.

3 The $\lambda\eta$-Algebras $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$

In this section we use the fact that $\mathbb{A}$ and $\mathbb{A}_{\mathrm{REC}}$ are cartesian closed categories to construct the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$. We show that they are $\lambda\eta$-algebras and that they are sensible (all unsolvable terms are equated).

3.1 λ-Algebras from $\mathbb{A}$ and $\mathbb{A}_{\mathrm{REC}}$

The first model of the untyped λ-calculus was given by Scott in 1969 and a variety of other models followed in the 1970's. It was not until rather later that the general idea of what a model of the λ-calculus should be took shape.

We consider that a model of the λ-calculus must be a λ-*algebra*, a combinatory algebra which also satisfies every equation provable in the λ-calculus (i.e. the formal theory $\lambda\beta$). If in addition all equations provable in the $\lambda\eta$-calculus (the formal theory $\lambda\beta\eta$) are satisfied, we call it a $\lambda\eta$-algebra. We describe a λ-algebra by a tuple $\langle \mathcal{A}, \bullet, [\![-]\!]_{\_} \rangle$ where $\mathcal{A}$ is a combinatory algebra with application operation $\bullet$, and for each valuation ρ, $[\![-]\!]_\rho$ is a map from λ-terms to elements of $\mathcal{A}$. For a precise definition, the reader is referred to [3, §5.2].

Scott showed how λ-algebras arise in cartesian closed categories, as follows.

For any category we say that an object R is *reflexive* if there are morphisms $\mathit{Fun} : R \to (R \Rightarrow R)$ and $\mathit{Gr} : (R \Rightarrow R) \to R$ satisfying $\mathit{Gr}; \mathit{Fun} = \mathrm{id}_{R\Rightarrow R}$. In this case we say that $R \Rightarrow R$ is a *retract* of R, and that Fun and Gr are the *retraction morphisms*. A CCC $\mathbb{C}$ with reflexive object R, together with the retraction morphisms Fun and Gr, define a λ-algebra.

First, some notation: we write B^n for the n-fold product $(\cdots((B \times B) \times B)\cdots) \times B$, with the intention that $B^0 = \mathbf{1}$. Given $f_1, \ldots, f_n : A \to B$ we define the n-tuple $\langle f_1, \ldots, f_n \rangle : A \to B^n$ by $\langle\rangle = !_A$, $\langle f_1, \ldots f_{r+1} \rangle = \langle \langle f_1, \ldots, f_r \rangle, f_{r+1} \rangle$. Then if $\Delta = \langle x_1, \ldots, x_n \rangle$, write $\Pi^{\Delta}_{x_i}$ for the obvious projection onto the i^{th} component. Then define a λ-algebra $\langle \mathcal{A}, \bullet, [\![-]\!]_{\_} \rangle$ as follows.

(1) $\mathcal{A}$ is the homset $\mathrm{Hom}_{\mathbb{C}}(\mathbf{1}, R)$.
(2) For any object A with $f, g : A \to R$ define $f \bullet g = \langle f; \mathit{Fun}, g \rangle; \mathrm{eval}_{R,R}$. In particular this defines a binary operation on $\mathcal{A}$.
(3) If $\{x_1, \ldots, x_n\} \supseteq \mathrm{FV}(s)$ define inductively the morphism $[\![s]\!]_\Delta : R^n \to R$, where $\Delta = \langle x_1, \ldots x_n \rangle$, as follows:

$$\begin{aligned} [\![x]\!]_\Delta &= \Pi^\Delta_x, \\ [\![st]\!]_\Delta &= [\![s]\!]_\Delta \bullet [\![t]\!]_\Delta, \\ [\![\lambda x.s]\!]_\Delta &= \Lambda([\![s]\!]_{\Delta \cdot x}); \mathit{Gr}. \end{aligned}$$

In the last clause we may assume that x does not appear in Δ (by renaming if necessary).
(4) If ρ is a valuation mapping variables to elements of $\mathcal{A}$, and Δ is as above, define the morphism $\rho^\Delta : \mathbf{1} \to R^n$ by $\rho^\Delta = \langle \rho(x_1), \ldots, \rho(x_n) \rangle$. Then set

$$[\![s]\!]_\rho = \rho^\Delta; [\![s]\!]_\Delta.$$

The following result, along with a proof, can be found for example in [3, §5.5].

Proposition 37 *With the above construction $\langle \mathcal{A}, \bullet, [\![-]\!]_{\_} \rangle$ is a λ-algebra. We denote this model $\mathcal{M}(\mathbb{C}, R, \mathrm{Fun}, \mathrm{Gr})$. This is a $\lambda\eta$-algebra if and only if the retraction morphisms are iso (i.e. $\mathrm{Fun}; \mathrm{Gr} = id_R$).*

In fact, every λ-algebra can be obtained in this way, from some CCC with a reflexive object.

Recall that the arena U, the "maximal" single-tree arena defined in Example 3, is given in sequence-subset form by $\langle \mathbb{N}^* \rangle$. Observe that, as arenas, $U = U \Rightarrow U$, so that the strategy id_U defines both the morphisms $\mathit{Fun} : U \to (U \Rightarrow U)$ and $\mathit{Gr} : (U \Rightarrow U) \to U$, and these morphisms satisfy $\mathit{Gr}; \mathit{Fun} = \mathrm{id}_{U \Rightarrow U}$ and $\mathit{Fun}; \mathit{Gr} = \mathrm{id}_U$. Also U is r.e., and Fun and Gr are recursive strategies, so U is a reflexive object of both $\mathbb{A}$ and $\mathbb{A}_{\mathrm{REC}}$, for which the retraction maps form an isomorphism.

Definition 38 We write $\mathcal{D}$ for the $\lambda\eta$-algebra $\mathcal{M}(\mathbb{A}, U, \mathit{Fun}, \mathit{Gr})$, and $\mathcal{D}_{\mathrm{REC}}$ for the $\lambda\eta$-algebra $\mathcal{M}(\mathbb{A}_{\mathrm{REC}}, U, \mathit{Fun}, \mathit{Gr})$.

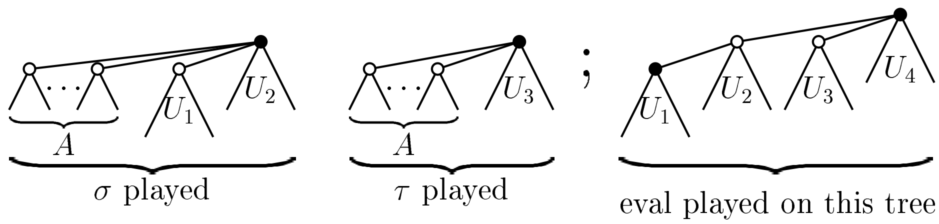
The only difference between $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ is the ambient categories from which they arose, and there is no difference in the strategies used for identities, projections and retractions in those categories. Thus the interpreting function $[\![-]\!]_{\_}$ is the same in both $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, so we write $[\![s]\!]$ for the strategy which denotes the term s in either.

For future use we lift the notion of application in $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ directly to strategies on U:

Definition 39 For innocent strategies σ and τ on $U^n \Rightarrow U$ (for any $n \in \mathbb{N}_0$) we write $\sigma \bullet \tau$ for $\langle \sigma; \mathit{Fun}, \tau \rangle; \mathrm{eval}_{U,U}$.

Thus for terms s and t, not necessarily closed, $[\![st]\!]_\Delta = [\![s]\!]_\Delta \bullet [\![t]\!]_\Delta$.

Since some later results will require a fairly delicate analysis of application as strategies, we investigate what happens. Firstly note that since the morphism *Fun* is given by the strategy id_U it does not affect the moves of the strategy σ. The arenas of the outer composition are shown below, with each subtree being a copy of the arena U:



The composition identifies moves made in the U arenas marked U_1, U_2 and U_3 and all are hidden, with the moves being made in U_4 visible. All the little trees labelled A — which represent the free variables in σ and τ, so we could call them *context subtrees* — are merged into one set of visible trees (as in Remark 24).

The strategy eval plays in response to the initial move, the root of U_4, the first move the root of the tree U_2. Thereafter moves played in the tree U_2 are copied across into U_4 and vice-versa. Similarly moves are copied between U_1 and U_3. That is, the composition first plays across into the root node of the tree $U_1 \Rightarrow U_2$, on which σ is played. Moves made by this strategy σ in the tree U_2 are visible, and moves made in the tree U_1 are copied across and played against by τ.

Thus the net effect is that $\sigma \bullet \tau$ is the strategy whose visible moves are those of σ in all except the leftmost non-context subtree, in which moves are played and hidden with τ playing as the opponent.

Note that when there are no free variables, and hence no context subtrees, this is exactly the same as the strategy $\tau; \sigma$.

We now examine the properties of the λ-algebras we have defined. We are able to show that they are sensible (i.e. the equational theory induced by the models is a sensible λ-theory), but in other respects there are rather unsatisfactory.

A standard method to show that a model is sensible is to use an approximation on the elements of the model. The approximation we will use is based on the approximation of *arenas* which then induces an approximation on strategies.

Definition 40 If σ is an innocent strategy on an arena A and B is a sub-arena of A we define $\sigma\|_B : \vec{v} \to m$, with m justified by a move m^- in $\vec{v}$, if $\sigma : \vec{v} \to m$ with m justified by the same move m^- in $\vec{v}$, and all of the moves in $\vec{v}$ and m are in the sub-arena B.

Here we are considering σ as an innocent function (and the justification pointers of the P-view $\vec{v}$ and the move m do not make any difference as to whether $\sigma\|_B : \vec{v} \to m$).

Note that $\sigma\|_B$ is still a strategy on the arena A, but is undefined at some P-views. An alternative definition is to consider the strategies as subsets of legal positions, then $\sigma\|_B = (\sigma \cap \{\vec{s} \mid \vec{s} \text{ is a legal position of } B\})^+$, if S^+ means the closure of the set S of legal positions under the rule of contingent completeness.

The intuition behind this is that parts of the arena A other than B are "out of bounds" and $\sigma\|_B$ will neither play there nor respond to moves played there, but will otherwise behave as σ.

Lemma 41 *If $\sigma : A \Rightarrow B$, A' is a sub-arena of A and B' is a sub-arena of B then*

$$\sigma\|_{A \Rightarrow B'} = \sigma; \iota_{B'}$$
$$\sigma\|_{A' \Rightarrow B} = \iota_{A'}; \sigma$$

where $\iota_{A'} : A \to A$ and $\iota_{B'} : B \to B$ are the obvious subsets of the identity strategies on A and B respectively.

PROOF. A straightforward modification of the proof that $\mathrm{id}_A ; \sigma = \sigma = \sigma ; \mathrm{id}_B$ $\square$

Corollary 42 *If $\sigma : A \Rightarrow B, \tau : B \Rightarrow C$, B' is a sub-arena of B and C' a* *sub-arena of C then*

$$(\sigma\|_{A\Rightarrow B'})\,;\tau = \sigma\,;(\tau\|_{B'\Rightarrow C})$$
$$\sigma\,;(\tau\|_{B\Rightarrow C'}) = (\sigma\,;\tau)\|_{A\Rightarrow C'}$$

With this definition of "restriction" we introduce arenas which approximate the arena U, and an induced approximation on innocent strategies over U.

Definition 43 (1) $U_0 = M$, the "minimal" single-tree arena defined in Example 3,
(2) $U_{n+1} = U_n \Rightarrow U_n$,
(3) If σ is an innocent strategy on U then $\sigma_n = \sigma\|_{U_n}$.

This particular definition of U_n has the following key property:

Lemma 44 $eval_{U,U}\|_{(U_{n+1}\times U)\Rightarrow U} \;=\; eval_{U,U}\|_{(U\times U_n)\Rightarrow U_n}$

PROOF. As strategies we know that $\mathrm{eval}_{U,U}$ and $\mathrm{id}_{U\Rightarrow U}$ are identical. Further, from lemma 41 it is clear that for any sub-arena A' of A we have $\mathrm{id}_A\|_{A'\Rightarrow A} = \mathrm{id}_A\|_{A\Rightarrow A'}$. Hence taking $A' = U_n \Rightarrow U_n$ and $A = (U \Rightarrow U) = U$ we have that, as strategies

$$\begin{aligned}\mathrm{eval}_{U,U}\|_{(U_{n+1}\times U)\Rightarrow U} &= \mathrm{id}_{U\Rightarrow U}\|_{(U_n\Rightarrow U_n)\Rightarrow(U\Rightarrow U)}\\ &= \mathrm{id}_{U\Rightarrow U}\|_{(U\Rightarrow U)\Rightarrow(U_n\Rightarrow U_n)}\\ &= \mathrm{id}_{U\Rightarrow U}\|_{((U\Rightarrow U)\times U_n)\Rightarrow U_n}\\ &= \mathrm{eval}_{U,U}\|_{(U\times U_n)\Rightarrow U_n} \qquad \square\end{aligned}$$

Thus σ_n satisfies these properties, which we need for an approximation:

Lemma 45 *For $\sigma, \tau : \mathbf{1} \to U$ in $\mathbb{A}$ (i.e. elements of the models $\mathcal{D}$ or $\mathcal{D}_{\mathrm{REC}}$),*

(i) $\sigma = \bigcup_{n\in\omega} \sigma_n$,

(ii) $\sigma_0 = \perp$,

(iii) $(\sigma_m)_n = \sigma_{\min(m,n)}$,

(iv) $\sigma_{n+1} \bullet \tau = (\sigma \bullet \tau_n)_n$.

PROOF. (i)–(iii) follow immediately from the definition. For (iv) use Lemma 44. $\square$

Theorem 46 *The λ-algebras $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ are sensible. Any term s is unsolvable if and only if $[\![s]\!] = \bot$.*

PROOF. This now follows a standard argument, which we do not reproduce in entirety. The full details of this fact for the model D_∞ can be found in [3, §19.2], and uses only properties of D_∞ which we have proved for $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$.

The argument uses the technique of *labelled β-reduction* introduced by Hyland in [8] and Wadsworth in [19]. The properties of the approximation are used to show that labelled reduction is monotone in the model, and hence that labelled β-normal forms are maximal. This gives rise to an approximation theorem — in the model any term is the union of its *approximate normal forms* (introduced by Wadsworth in [18]). Finally, it is simple to show that the only approximate normal forms of unsolvable terms have denotation $\bot$. $\square$

In addition to being sensible, there are other desirable properties of models which we might aim for:

Definition 47 Let $\mathcal{M} = \langle \mathcal{A}, \bullet, [\![-]\!]_{-} \rangle$ be a λ-algebra.

(1) $\mathcal{M}$ is a *λ-model* if it is *weakly extensional*, that is,

$$(\forall a \in \mathcal{A}.\ [\![s]\!]_{\rho(x:=a)} = [\![t]\!]_{\rho(x:=a)}) \implies [\![\lambda x.s]\!]_\rho = [\![\lambda x.t]\!]_\rho.$$

(The principle is that, after interpretation, $(\forall x.s = t) \implies \lambda x.s = \lambda x.t.$) [1]

(2) $\mathcal{M}$ is *extensional* if for all a and b in $\mathcal{A}$,

$$(\forall x \in \mathcal{A}.\ a \bullet x = b \bullet x) \implies a = b.$$

(3) $\mathcal{M}$ is *universal* if every element of $\mathcal{A}$ is the denotation of some term, that is,

$$\forall a \in \mathcal{A}.\ \exists s \in \Lambda^0.\ [\![s]\!] = a.$$

Note that for any function on the elements of a λ-algebra, say $f(x_1, \ldots, x_n)$, there is an element of the λ-algebra representing that function, in this case $[\![\lambda x_1 \ldots x_n.f]\!]$. The principle of weak extensionality ensures that this element is unique. Furthermore, the λ-models have a pleasing categorical structure.

[1] Recall that every λ-algebra arises as $\mathcal{M}(\mathbb{C}, R, \mathit{Fun}, \mathit{Gr})$ as described in Section 3.1, where $\mathbb{C}$ is a CCC with an object R which is reflexive, via the retraction morphisms Fun and Gr. The λ-algebra $\mathcal{M}(\mathbb{C}, R, \mathit{Fun}, \mathit{Gr})$ is a λ-model if and only if the object *R has enough points* (i.e. $\forall f, g : R \to R.\ (\forall r : \mathbf{1} \to R.\ r; f = r; g) \implies f = g$).

An extensional model has the property that every element is determined by its applicative behaviour — this is what one might expect in a "functional" setting. However, the λ-calculus itself does not have this property, since $Iab = ab = 1ab$ for all terms a, b, but $I \neq 1$. However, if η-conversion is included the calculus will have this property (for example $I = 1$ in the $\lambda\eta$-calculus).

A universal model is a very powerful structure, as one can be sure of a 1-1 correspondence between the elements of the model and the equivalence classes of the terms of the λ-calculus modulo whatever theory the model imposes. In particular, one might hope to accomplish existence proofs for the λ-calculus by working on the model. A universal model could even be considered an alternative presentation of the λ-calculus, with respect to some λ-theory.

However, the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ do not enjoy any of these properties.

Theorem 48 *$\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ are not universal, not extensional and not even weakly extensional (thus they are not λ-models).*

PROOF. Recall that $\perp$ is the "empty" innocent strategy with everywhere undefined innocent function. Define the strategy $\perp'$ on the arena U by the following innocent function:

$$\varepsilon \mapsto 1$$

That is, $\perp'$ is the set $\{\varepsilon, \varepsilon \cdot 1\} \ \cup \ \{\varepsilon \cdot 1 \cdot n \mid n \in \mathbb{N}\}$

We may consider $\perp$ and $\perp'$ as defining morphisms $\mathbf{1} \to U$, and so as elements of the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$. Certainly we have $\perp \neq \perp'$. However, the fact that neither $\perp$ nor $\perp'$ ever moves outside the first subtree means that they cannot make moves that are visible in the composition which arises in application. Hence for all strategies $\sigma : \mathbf{1} \to U$,

$$\perp \bullet \sigma = \perp' \bullet \sigma = \perp.$$

Hence $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ are not extensional.

Now it is a standard result that any λ-algebra is extensional if and only if it is weakly extensional and a $\lambda\eta$-algebra. For a proof see, for example, [3, §5.2]. Since $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ are $\lambda\eta$-algebras but are not extensional, they cannot be weakly extensional either.

We show that $\bot'$ is not the denotation of any term as follows: Suppose there is a term s such that $[\![s]\!] = \bot'$. We have shown that $[\![s]\!] \bullet \sigma = \bot$ for all strategies $\sigma : \mathbf{1} \to U$, so $[\![st]\!] = \bot$ for all terms t. Hence st is unsolvable for all t, so s is unsolvable and hence $[\![s]\!] = \bot$. $\square$

Finding an improved version of these models is the subject of the next sections.

4 Economical Form and Exact Correspondence

Our aim in this section is to gain more information on the strategies which denote terms of the λ-calculus in the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$.

To do so we first present an *economical* way to encode strategies. We then recall the definition of the *Nakajima tree* of a term, an infinitary syntax tree deriving from the Böhm tree after infinite η-expansion. We give a representation of Nakajima trees which is *variable-free* (i.e. all bound variables are replaced by syntax-free encoded "pointers", which can be seen as two-dimensional de Bruijn indices). The powerful result linking strategies and Nakajima trees, which we call an *Exact Correspondence Theorem*, is that the denotation of a term, in the new economical representation, is precisely the same as the labelling function of the variable-free Nakajima tree of that term. This allows us to examine the local structure of the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ in more detail than we were able to in the previous chapter, and show that they induce the equational theory $\mathcal{H}^*$.

4.1 Innocent Strategies in Economical Form

Recall the representation of innocent strategies as innocent functions, mapping P-views to justified P-moves. We may encode any arena as a subset of $\mathbb{N}^*$, so that a P-view is a (justified) sequence of elements of $\mathbb{N}^*$, and also encode the justification pointer as a member of $\mathbb{N}_0$ by counting the number of pairs of moves it points backwards over, as in Remarks 7 and 28.

There are three properties which allow for a more compact representation of an innocent strategy:

(1) We know that the domain of an innocent function is made up of P-views. Lemma 13 shows that each non-initial O-move in any P-view must be a child of the previous move, and the initial move must be ε.
(2) The conditions on an innocent function mean that, for a P-view which is in the domain of the function, given only the O-moves in this P-view

and the value of the innocent function on strictly shorter P-views we can reconstruct the P-view entirely.

(3) The condition of Justification on well-formed sequences means that the P-move to which this P-view is mapped must be a child of the move which it is justified by.

In view of these redundancies, we encode innocent strategies over any single-tree arena as (partial) maps from $\mathbb{N}^*$ to $\mathbb{N} \times \mathbb{N}_0$. We call this encoding the *economical form* of σ and sometimes write it e_σ (but usually we abuse notation and write it f_σ too). It is defined as follows:

$$e_\sigma : \langle v_1, \ldots, v_n \rangle \mapsto (i,p) \text{ iff}$$

$$f_\sigma : \begin{array}{ccccccccccccc} \bullet & \circ & \bullet & \circ & \bullet & & \circ & \bullet & & \circ & \bullet & & \circ \\ & & & & & \cdots & & & \cdots & & & \mapsto & \\ \varepsilon & \vec{s}_1 & \vec{s}_1 v_1 & \vec{s}_2 & \vec{s}_2 v_2 & & \vec{s}_{n-p} & \vec{s}_{n-p} v_{n-p} & & \vec{s}_n & \vec{s}_n v_n & & \vec{s}_{n-p} v_{n-p} i \end{array}$$

(We set $p = n$ when the resulting move in the clause of f_σ is a child of the initial move, intending "$\vec{s}_0 v_0$" to mean the sequence $\langle \varepsilon \rangle$.)

Justification pointers in the P-view can be deduced from the behaviour of f_σ on shorter P-views, and so have been omitted. The sequences $\vec{s}_i$ are sequences of natural numbers encoding moves in the way we use for sequence-subset form of arenas. By the move "$\vec{s}_i v_i$" we mean the move which encoded by the sequence $\vec{s}_i \cdot v_i$, which by definition is the v_i^{th} child of the move coded by $\vec{s}_i$.

Furthermore, we can expand any partial function $f : \mathbb{N}^* \rightharpoonup \mathbb{N} \times \mathbb{N}_0$ which has prefix-closed domain and satisfies $f(\vec{v}) = (i,p) \implies 0 \leq p \leq |\vec{v}|$ into an innocent strategy on U. Depending on the function, we might not need the whole of U to contain the strategy.

A strategy on a multiple-tree arena will be encoded as a tuple of such functions, one for each component. In practice we will only be interested in using this encoding for the arena U.

Example 49 Recall that the following is the innocent function of the copycat strategy id_U:

$$\begin{array}{ccc} \bullet & \mapsto & \circ \\ \varepsilon & & 1 \end{array} \qquad \begin{array}{cccccccc} \bullet & \cdots & \bullet & \circ & \bullet & \mapsto & \circ \\ \varepsilon & & (a{+}1)\vec{s} & 1a\vec{s} & 1a\vec{s}b & & (a{+}1)\vec{s}b \end{array}$$

$$\begin{array}{ccccc} \bullet & \circ & \bullet & \mapsto & \circ \\ \varepsilon & 1 & 1a & & (a{+}1) \end{array} \qquad \begin{array}{cccccccc} \bullet & \cdots & \bullet & \circ & \bullet & \mapsto & \circ \\ \varepsilon & & 1a\vec{s} & (a{+}1)\vec{s} & (a{+}1)\vec{s}b & & 1a\vec{s}b \end{array}$$

Here $\vec{s}$ range over sequences of appropriate parity, a and b over positive natural numbers. The reader is invited to check that the economical form of this strategy is given by: $\varepsilon \mapsto (1,0), i \mapsto (i+1,1)$ and for nonempty sequences $\vec{v}$, $\vec{v}i \mapsto (i,1)$.

4.2 Nakajima Trees and Variable-Free Form

The following presentation of terms as trees was first proposed by Nakajima in [16]. The principle that a term of the untyped λ-calculus may be applied to any number of other terms suggests that, for example, the term $I = \lambda x.x$ might be better expanded infinitely many times by the rule η, and represented by the pseudo-syntax $\lambda x z_0 z_1 z_2 \ldots \bullet x z_0 z_1 z_2 \ldots$. The large dot $\bullet$ is used to make clear the "end" of the infinite chain of abstractions and the start of the infinite chain of applications in the term. Combining this idea into a presentation of terms in the style of Böhm trees leads to the following definition.

Definition 50 Let Σ be the set $\{\lambda x_1 x_2 \ldots \bullet\, y \mid x_1, x_2, \ldots, y \text{ variables}\}$.

For a λ-term s the *Nakajima tree* of s, written $\mathrm{NT}(s)$, is the countably branching, countably deep partially Σ-labelled tree defined inductively as follows.

$$\mathrm{NT}(s) = \perp \qquad \text{if } s \text{ is unsolvable}$$

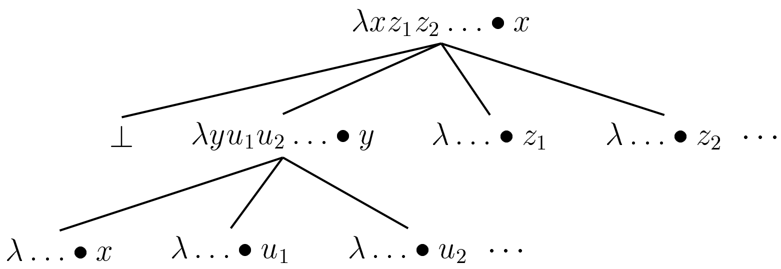
$$\begin{array}{lc}
\mathrm{NT}(s) = & \lambda x_1 \ldots x_n z_0 z_1 \ldots \bullet\, y \qquad \text{if } s \text{ has HNF } \lambda x_1 \ldots x_n . y s_1 \ldots s_m \\
 & \mathrm{NT}(s_1) \ \cdots \ \mathrm{NT}(s_m) \quad \mathrm{NT}(z_0) \quad \mathrm{NT}(z_1) \ \cdots
\end{array}$$

where $z_0, z_1, \ldots$ are countably many fresh variables.

A formalization of the process of finding fresh variables at each stage is given in [16].

In order to work modulo α-conversion, we would like to follow de Bruijn (as in [6]) and construct a variable-free representation of Nakajima trees. We know that each node which has a label has one of the form $\lambda x_1 x_2 \ldots \bullet\, y$ and always has countably many children, so all we need to encode are the head variables at each node. Assuming that the term whose Nakajima tree we are encoding is closed, each head variable is an instance of an abstraction from one of its ancestors in the tree. We thus encode each labelled node's head variable as a pair, the second component of which counts how many levels up the tree we go to find the abstraction which introduced this head variable, and the first component counts how far along the infinite chain of abstractions this variable appears.

Example 51 We illustrate with an example. The term $\lambda x.x\Omega(\lambda y.yx)$ has Nakajima tree of which the following is a part (some subtrees are missing):



The node labelled $\lambda \ldots \bullet\, z_1$ should be encoded as the pair $(2,1)$ corresponding to the fact that we go up one level of the tree and take the second abstracted variable to get z_1. Similarly the node in the lowest pictured level labelled $\lambda \ldots \bullet\, x$ should be encoded $(1,2)$.

A precise definition of this encoding is tricky, because we have to deal with free variables, of which there are countably many in Nakajima trees. The following definition is quite technical, and we have to do a bit of work to show that this definition matches up with the informal notion just described, but this form is useful for the characterisation proof which follows.

Definition 52 For a partially $(\mathbb{N} \times \mathbb{N}_0)$-labelled tree p the tree $\{p\}^*$ is the same tree labelled identically, except that nodes at depth d labelled $(i, d+1)$ are relabelled $(i, d+2)$.

Similarly the tree $\{p\}^n$, for $n \in \mathbb{N}_0$, is labelled identically except for nodes of depth d as follows:

(1) those labelled (i, d) are relabelled $(i+n, d)$;
(2) those labelled $(i, d+1)$ for $i \leq n$ are relabelled $(n-i+1, d)$;
(3) those labelled $(i, d+1)$ for $i > n$ are relabelled $(i-n, d+1)$.

For a term s with free variables within Δ the *variable-free form* of the Nakajima tree of s, $\mathrm{VFF}_\Delta(s)$, is the following partially $(\mathbb{N}\times\mathbb{N}_0)$-labelled tree:

$$\mathrm{VFF}_\Delta(s) = \bot, \qquad \text{for unsolvable } s.$$

$$\mathrm{VFF}_\Delta(\lambda x_1 \ldots x_n \bullet s) = \{\mathrm{VFF}_{\Delta\cdot\langle x_1,\ldots,x_n\rangle}(s)\}^n, \qquad \text{if } s \text{ is of the form } v_j s_1 \ldots s_m.$$

$$\mathrm{VFF}_\Delta(v_j s_1 \ldots s_m) = \begin{array}{c} (j,1) \\ \{\mathrm{VFF}_\Delta(s_1)\}^* \ \cdots \ \{\mathrm{VFF}_\Delta(s_m)\}^* \quad CC(1) \quad CC(2) \ \cdots \end{array}$$

where $\Delta = \langle v_k, \ldots, v_1\rangle$ (note the reverse order).

Here $CC(i)$ is the infinite tree defined by

$$CC(i) = \begin{array}{c} (i,1) \\ CC(1) \quad CC(2) \quad CC(3) \ \cdots \end{array}$$

One should think of Δ as a context for the tree — when the pointer for a variable goes up one beyond the root of the tree, it references the context. The operation $\{-\}^n$ corresponds to looking up to see which variables lie in the last n of the context and adjoining them to the left of the tree (in reverse order, for technical reasons). The operation $\{-\}^*$ passes references to the context further up the tree.

We note here that for any tree p, $\{p\}^0 = p$ and $\{\{p\}^m\}^n = \{p\}^{m+n}$.

Lemma 53 *This definition coincides with the informal notion of variable-free form described earlier. Formally, suppose that s is a term with free variables within all occurring in the sequence $\Delta = \langle v_k, \ldots, v_1\rangle$. Construct the Nakajima tree of s, and rename all the bound variables so that if $\vec{a} \in \mathbb{N}^*$ is the sequence identifying a node of the Nakajima tree then the abstracted variables at that node are renamed to $x_1^{\vec{a}}, x_2^{\vec{a}}, \ldots$ (in that order). Let this renamed Nakajima tree have labelling function A, and consider $VFF_\Delta(s)$ also as a labelling function.*

Then for any sequence $\vec{a} = \langle a_1, \ldots, a_p\rangle$ there are three possibilities for $A(\vec{a})$:

(1) If $\vec{a} \notin dom(A)$ then $VFF_\Delta(s)$ is unlabelled or undefined at $\vec{a}$,
(2) If $A(\vec{a}) = \lambda x_1^{\vec{a}} x_2^{\vec{a}} \ldots \bullet v_j$ then $VFF_\Delta(s)(\vec{a}) = (j, p+1)$,
(3) If $A(\vec{a}) = \lambda x_1^{\vec{a}} x_2^{\vec{a}} \ldots \bullet x_j^{\langle a_1,\ldots,a_{p-r}\rangle}$ then $VFF_\Delta(s)(\vec{a}) = (j, r)$.

PROOF. By induction on p (the length of $\vec{a}$), for all terms s and sequences Δ (which contain all the free variables of s) simultaneously. Throughout this proof we use Γ as a syntactic abbreviation for $\Delta \cdot \langle x_1, \ldots, x_n \rangle$.

Base Case: There are three cases:

(1) If s is unsolvable then $\varepsilon \notin A$ and $\mathrm{VFF}_\Delta(s)$ is unlabelled at the root.
(2) If s has HNF $\lambda x_1 \ldots x_n.v_j s_1 \ldots s_m$
then $A(\varepsilon) = \lambda x_1^\varepsilon x_2^\varepsilon \ldots \bullet v_j$ and $\mathrm{VFF}_\Gamma(v_j s_1 \ldots s_m)(\varepsilon) = (j+n, 1)$
hence $\mathrm{VFF}_\Delta(s)(\varepsilon) = \{\mathrm{VFF}_\Gamma(v_j s_1 \ldots s_m)\}^n(\varepsilon) = (j, 1)$.
(3) If s has HNF $\lambda x_1 \ldots x_n.x_j s_1 \ldots s_m$
then $A(\varepsilon) = \lambda x_1^\varepsilon x_2^\varepsilon \ldots \bullet x_j^\varepsilon$ and $\mathrm{VFF}_\Gamma(x_j s_1 \ldots s_m)(\varepsilon) = (n-j+1, 1)$
hence $\mathrm{VFF}_\Delta(s)(\varepsilon) = \{\mathrm{VFF}_\Gamma(v_j s_1 \ldots s_m)\}^n(\varepsilon) = (j, 0)$.

In each case the result holds.

Inductive Step: We assume that the result holds for all terms s, sequences Δ which contain all the free variables of s, and sequences $\vec{a}$ of length up to and including l.

Take a particular term s and sequence $\Delta = \langle v_k, \ldots, v_1 \rangle$ containing all the free variables of s. If s is unsolvable then the result is trivial as in (i) above. Otherwise, suppose that s has HNF $\lambda x_1 \ldots x_n.y s_1 \ldots s_m$ for some variable y (which as above will be either x_j or v_j, it will make no difference which). Take any sequence $\vec{a} = \langle a_1, \ldots, a_p \rangle$ for $p \leq l$, and any $i \in \mathbb{N}^0$. We show that the result holds for the sequence $i \cdot \vec{a}$ (and we already know it holds for the sequence ε) which will establish the inductive step as required.

Suppose that we take the Nakajima tree of the term s_i, and renamed bound variables so that the i^{th} abstracted variable at the node encoded by $\vec{v}$ is renamed to $y_i^{\vec{v}}$. Let the labelling function of this tree be B. We know that the free variables of this tree are contained within Γ, and we will be applying the induction hypothesis to B. The definition of Nakajima tree describes how the labels of B are related to those of A.

There are six cases:

(1) $i \leq m$ and $i \cdot \vec{a} \notin \mathrm{dom}(A)$.
Then $\vec{a} \notin \mathrm{dom}(B)$, so by the induction hypothesis $\mathrm{VFF}_\Gamma(s_i)$ is unlabelled or undefined at $\vec{a}$, and hence $\mathrm{VFF}_\Delta(s)$ is unlabelled or undefined at $i \cdot \vec{a}$.
(2) $i \leq m$ and $A(i \cdot \vec{a}) = \lambda x_1^{i \cdot \vec{a}} \ldots \bullet v_j$.

Then $B(\vec{a}) = \lambda y_1^{\vec{a}} \ldots \bullet v_j$ so by the induction hypothesis

$$\begin{aligned} && \mathrm{VFF}_\Gamma(s_i)(\vec{a}) &= (j+n, p+1) \\ \text{hence} && \{\mathrm{VFF}_\Gamma(s_i)\}^*(\vec{a}) &= (j+n, p+2) \\ \text{hence} && \mathrm{VFF}_\Gamma(ys_1 \ldots s_m)(i \cdot \vec{a}) &= (j+n, p+2) \\ \text{hence} && \mathrm{VFF}_\Delta(s)(i \cdot \vec{a}) &= (j, p+2). \end{aligned}$$

(3) $i \leq m$ and $A(i \cdot \vec{a}) = \lambda x_1^{i \cdot \vec{a}} \ldots \bullet x_j^{\varepsilon}$.
Then $B(\vec{a}) = \lambda y_1^{\vec{a}} \ldots \bullet x_j$ so by the induction hypothesis

$$\begin{aligned} && \mathrm{VFF}_\Gamma(s_i)(\vec{a}) &= (n-j+1, p+1) \\ \text{hence} && \{\mathrm{VFF}_\Gamma(s_i)\}^*(\vec{a}) &= (n-j+1, p+2) \\ \text{hence} && \mathrm{VFF}_\Gamma(ys_1 \ldots s_m)(i \cdot \vec{a}) &= (n-j+1, p+2) \\ \text{hence} && \mathrm{VFF}_\Delta(s)(i \cdot \vec{a}) &= (j, p+1). \end{aligned}$$

(4) $i \leq m$ and $A(i \cdot \vec{a}) = \lambda x_1^{i \cdot \vec{a}} \ldots \bullet x_j^{\langle i, a_1, \ldots, a_{n-r} \rangle}$.
Then $B(\vec{a}) = \lambda y_1^{\vec{a}} \ldots \bullet y_j^{\langle a_1, \ldots, a_{n-r} \rangle}$ so by the induction hypothesis

$$\begin{aligned} && \mathrm{VFF}_\Gamma(s_i)(\vec{a}) &= (j, r) \\ \text{hence} && \{\mathrm{VFF}_\Gamma(s_i)\}^*(\vec{a}) &= (j, r) \\ \text{hence} && \mathrm{VFF}_\Gamma(ys_1 \ldots s_m)(i \cdot \vec{a}) &= (j, r) \\ \text{hence} && \mathrm{VFF}_\Delta(s)(i \cdot \vec{a}) &= (j, r). \end{aligned}$$

(5) $i > m$ and $\vec{a} = \varepsilon$.
Then by the definition of Nakajima tree, $A(i) = \lambda x_1^i \ldots \bullet x_{i-m+n}^{\varepsilon}$. On the other hand,

$$\begin{aligned} && \mathrm{VFF}_\Gamma(ys_1 \ldots s_m)(i) &= (i-m, 1) \\ \text{so that} && \mathrm{VFF}_\Delta(s)(i) = \{\mathrm{VFF}_\Gamma(ys_1 \ldots s_m)\}^n(i) &= (i-m+n, 1). \end{aligned}$$

(6) $i > m$ and $\vec{a} = \vec{b} \cdot j$ for some $j \in \mathbb{N}_0$.
Then by the definition of Nakajima tree, $A(i \cdot \vec{a}) = \lambda x_1^{i \cdot \vec{a}} \ldots \bullet x_j^{i \cdot \vec{b}}$. On the other hand,

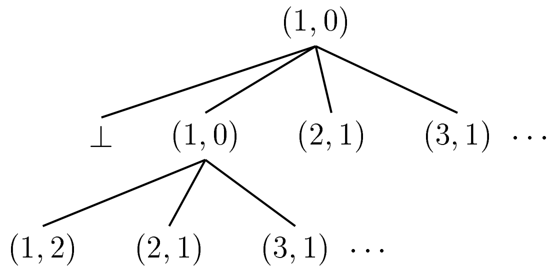
$$\begin{aligned} && \mathrm{VFF}_\Gamma(ys_1 \ldots s_m)(i \cdot \vec{a}) &= (j, 1) \\ \text{so that} && \mathrm{VFF}_\Delta(s)(i \cdot \vec{a}) &= (j, 1). \end{aligned}$$

We see that the result holds in each case. $\square$

Remark 54 The construction of VFF(s) from s, and that of NT(s) from s, is not a computable procedure in the same way the first definition of Böhm tree in [3, §10.1] is not. However Lemma 53 does give a recursive translation of Nakajima trees into variable-free form, and the latter can be presented effectively (as they are in [16]).

The connection between NT(s) and VFF(s), described formally in Lemma 53, is now illustrated. Take the term $s = \lambda x.x\Omega(\lambda y.yx)$, the Nakajima tree of which is given above in Example 51. Applying the rules for constructing the variable-free form gives a tree of which part is:

$\mathrm{VFF}_\varepsilon(s) =$



The reader is invited to compare this with the Nakajima tree, and check what Lemma 53 means at each pictured node.

4.3 Exact Correspondence and Local Structure

The representation of an innocent strategy described in Section 4.1, and that of Nakajima tree in Section 4.2 are now pulled together. In the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ the denotation of a term, in economical form, coincides with the labelling function of the variable-free form of its Nakajima tree.

Theorem 55 (Exact Correspondence Theorem) *If $s \in \Lambda$ with free variables in $\Delta = \langle v_k, \ldots, v_1 \rangle$ then $[\![s]\!]_\Delta = \{\, VFF_\Delta(s)\}^k$ when the former is considered in economical form and the latter as a labelling function.*

In particular for closed terms s, $[\![s]\!]_\varepsilon = VFF_\varepsilon(s)$.

PROOF. The proof is long-winded and the interested reader is referred to Appendix A. $\square$

Example 56 Referring back to the example of the last section, this means that the economical form of the strategy $[\![\lambda x.x\Omega(\lambda y.yx)]\!]$ is partially given by:

$$\begin{array}{ll} \varepsilon \mapsto (1,0) & \langle 4\rangle \mapsto (3,1) \\ \text{undefined on } \langle 1\rangle & \langle 2,1\rangle \mapsto (1,2) \\ \langle 2\rangle \mapsto (1,0) & \langle 2,2\rangle \mapsto (2,1) \\ \langle 3\rangle \mapsto (2,1) & \langle 2,3\rangle \mapsto (3,1). \end{array}$$

The Exact Correspondence Result gives us the local order structure of the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ immediately.

Corollary 57 *In the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, for closed terms s and t:*

$$[\![s]\!] \subseteq [\![t]\!] \overset{1}{\iff} NT(s) \subseteq NT(t) \overset{2}{\iff} D_\infty \models s \leq t \overset{3}{\iff} s \preceq t.$$

The order $\subseteq$ on the model is inclusion of strategies. That on Nakajima trees is inclusion of labelling function, modulo renaming of bound variables, with amounts to inclusion of variable-free form. The order $\leq$ on D_∞ is the standard order on the cpo, and the order $\preceq$ on Λ^0 is given by:

$$s \preceq t \iff \textit{for all contexts } C[\,],\ C[s] \textit{ solvable implies } C[t] \textit{ solvable.}$$

Thus the local structure of the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ is the $\lambda\eta$-theory $\mathcal{H}^*$, the maximal consistent sensible theory.

PROOF. Equivalence 1 is a trivial consequence of the Exact Correspondence Theorem, which shows that $[\![s]\!]$ and $\mathrm{NT}(s)$ are essentially the same thing modulo naming of bound variables.

Equivalence 2 was proved by Nakajima in [16, 3.5].

Equivalence 3 is Hyland's local structure theorem for D_∞ and a proof can be found in, for example, [3, §19.2]. The fact that the local structure is the $\lambda\eta$-theory $\mathcal{H}^*$ follows immediately from this. For a discussion of the properties of $\mathcal{H}^*$, see [3, §16.2]. □

5 EAC Strategies and a Universal Model of $\mathcal{H}^*$

The reason that $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ failed to be universal, or even weakly extensional, is the existence of strategies like $\perp'$ defined in the proof of Theorem 48. This strategy has the same applicative behaviour as $\perp$, and is not the denotation of any term. If we could throw out strategies which offend in this manner we would hope to improve the model.

Using a characterisation of Böhm-like trees which are the Böhm trees of some term, we can restrict our attention to the *effectively almost-everywhere copycat* strategies, and show that they live in a cartesian closed category which gives rise to a λ-algebra $\mathcal{D}_{\mathrm{EAC}}$. This does turn out to be universal. We then use the universality property to show that the model is also order-extensional (a stronger property than extensionality).

5.1 Effectively Almost-Everywhere Copycat Strategies

Now that we have another way to see what the denotations of terms are, we can translate standard results about definable Böhm-like trees into information about the definable strategies. These will be the *effectively almost-everywhere copycat* strategies, and in this section we define them and construct a cartesian closed category $\mathbb{A}_{\mathrm{EAC}}$ using them.

We first have to introduce some notation to refer to subtrees of an arena.

Definition 58 For tree-like $A \subseteq \mathbb{N}^*$, i.e. those subsets which are prefix-closed and satisfy $\vec{s} \cdot n \in A \implies \vec{s} \cdot m \in A$ for all $m < n$, we make the following definitions:

(1) If $\vec{s} \in A$ then $A \mathbin{@} \vec{s} = \{\vec{t} \mid \vec{s} \cdot \vec{t} \in A\}$.
(2) If $m \in \mathbb{N}_0$ then $A^{>m} = \{(i - m) \cdot \vec{s} \mid i \cdot \vec{s} \in A \wedge i > m\}$.

Thus $A \mathbin{@} \vec{s}$ is the subtree of A rooted at $\vec{s}$ (as defined it will still be a tree-like subset of $\mathbb{N}^*$), and $A^{>m}$ has had the first m branches of A deleted.

We also need a way to decode the economical form of innocent functions, at least to see where in the arena a decoded strategy is playing.

Definition 59 If f is the economical form of an innocent strategy on a single-tree arena A and $\vec{v} = \langle v_1, \ldots, v_n \rangle \in \mathrm{dom}(f)$ then we define a sequence of moves $\langle m_1, \ldots, m_{2n+2} \rangle$ as follows:

$$\begin{aligned} m_1 &= \varepsilon \\ m_{2k} &= m_{2(k-p)-1} \cdot i \qquad \text{if } f : \langle v_1, \ldots, v_{k-1} \rangle \mapsto (i, p); \\ m_{2k+1} &= m_{2k} \cdot v_k \qquad \text{for } k > 0 \end{aligned}$$

By $m_{2(k-p)-1} \cdot i$ we mean the move corresponding to that sequence in the sequence-subset representation of A, i.e. the i^{th} child of the move $m_{2(k-p)-1}$.

We say that the mapping $f : \vec{v} \mapsto (i,p)$ *codes* the P-move m_{2n+2}, because by following the P-strategy dictated by the function this is the move that will be the one specified by that mapping. We denote the move m_{2n+2} constructed from $\vec{v}$ and f in this manner by $\mathbf{m_p}^f(\vec{v})$. We omit the superscript wherever it is clear which strategy is intended.

Similarly the fact that $\vec{v} \in \mathrm{dom}(f)$ means that the O-move m_{2n+1} is in the strategy, and for this move we write $\mathbf{m_o}^f(\vec{v})$.

Note that for any innocent strategy the O-move $\mathbf{m_o}(\varepsilon)$ is the initial move ε and $\mathbf{m_p}(\varepsilon)$ is the first P-move made by the strategy in response.

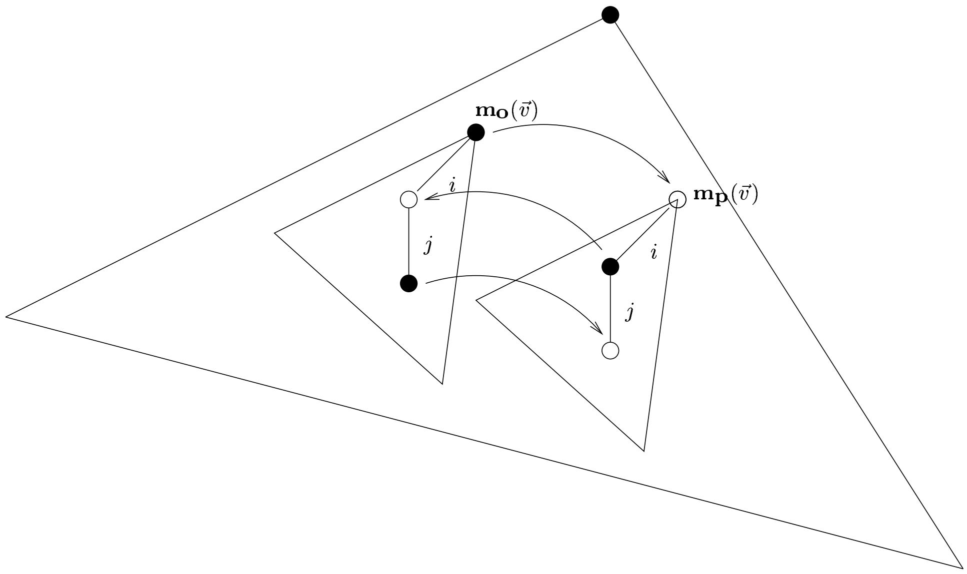
Now we can make the key definitions:

Definition 60 Consider an innocent strategy in economical form $f : \mathbb{N}^* \to \mathbb{N} \times \mathbb{N}_0$, over some single-tree arena A.

We say that f is *everywhere copycat* (EC) at $\vec{v} \in \mathbb{N}^*$ if f is undefined at $\vec{v}$ or the following hold:

(1) The arenas $A \,@\, \mathbf{m_o}(\vec{v})$ and $A \,@\, \mathbf{m_p}(\vec{v})$ are order-isomorphic (with respect to the prefix ordering, considered as subsets of $\mathbb{N}^*$);
(2) Whenever $\vec{w} \geq \vec{v}$ we have that for all $i \in \mathbb{N}$ such that the move $\mathbf{m_o}(\vec{w} \cdot i)$ exists, $f(\vec{w} \cdot i) = (i, 1)$;
(3) If $f(\vec{v}) = (i,p)$ then $p > 0$.

We take the opportunity to illustrate an everywhere copycat strategy. Let us assume that f is an innocent strategy in economical form on an arena A, and suppose f is defined at $\vec{v}$.



Intuitively we say that f is everywhere copycat at $\vec{v}$ if, from $\mathbf{m_p}(\vec{v})$ onwards, f's behaviour is simply to play copycat for as long as the arena will allow it. In the figure, the big triangle represents the arena A; the smaller triangle on the left represents the subarena $A \,@\, \mathbf{m_o}(\vec{v})$ and that on the right represents $A \,@\, \mathbf{m_p}(\vec{v})$ — note that by condition (i) the two are assumed to be isomorphic. Suppose O's response at $\mathbf{m_p}(\vec{v})$ is to play its i^{th} child, then P responds with the i^{th} child of $\mathbf{m_o}(\vec{v})$. If O at that point plays the j^{th} child of P's last move, then P moves over to the other subarena and responds with the j^{th} child of O's last move in that subarena, and so on. In the figure, the strategy f's action is indicated by the arrows i.e. P's response is always to flip to the other subarena and copy O's last move there. Condition (i) in the definition guarantees that P's copycat move will always be available.

We may view the definition in light of the correspondence between innocent strategies and Nakajima trees, thanks to the Exact Correspondence Theorem, and here we see that condition (ii) specifies that the subtree of the Nakajima tree corresponding to f, rooted at $\vec{v}$, has the following shape:

$$\lambda x_1 x_2 \ldots \bullet\, y$$

$$\mathrm{NT}(x_1) \qquad \mathrm{NT}(x_2) \quad \cdots$$

Condition (iii) of the definition is a technicality, which ensures that the variable y is not one of the x_i.

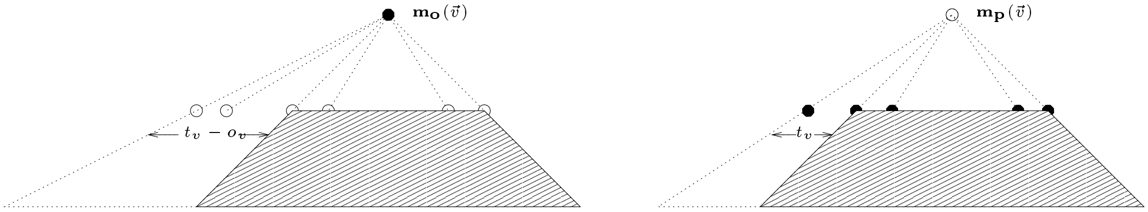
Definition 61 We say that f is *almost-everywhere copycat* (AC) at $\vec{v}$ if f is undefined at $\vec{v}$ or there exist numbers $t_{\vec{v}} \in \mathbb{N}_0$ and $o_{\vec{v}} \in \mathbb{Z}$ with $o_{\vec{v}} \leq t_{\vec{v}}$ called the *copycat threshold* and *offset* respectively, such that

(1) The arenas $(A \mathbin{@} \mathbf{m_o}(\vec{v}))^{>(t_{\vec{v}}-o_{\vec{v}})}$ and $(A \mathbin{@} \mathbf{m_p}(\vec{v}))^{>t_{\vec{v}}}$ are order-isomorphic;
(2) For all $i > t_{\vec{v}}$ such that the move $\mathbf{m_o}(\vec{v} \cdot i)$ exists, $f(\vec{v} \cdot i) = (i - o_{\vec{v}}, 1)$ and f is everywhere copycat at $\vec{v} \cdot i$;
(3) For all $\vec{w} \geq (\vec{v} \cdot k)$ with $k \leq t_{\vec{v}}$, if $f(\vec{w}) = (i, |\vec{w}| - |\vec{v}|)$ then $i \leq t_{\vec{v}} - o_{\vec{v}}$;
(4) If $f(\vec{v}) = (i, 0)$ then $i \leq t_{\vec{v}} - o_{\vec{v}}$.

(Note that f is EC at $\vec{v}$ if and only if f is AC at $\vec{v}$ with $t_{\vec{v}} = o_{\vec{v}} = 0$.)

Finally, we say that f is *effectively almost-everywhere copycat* (EAC) if f is recursive, almost-everywhere copycat at every sequence on which it is defined, and the functions $\vec{v} \mapsto t_{\vec{v}}$ and $\vec{v} \mapsto o_{\vec{v}}$ are recursive. A strategy σ over a single-tree arena A is EAC if its innocent function is EAC, and a strategy over a multiple-tree arena is EAC if all of its components are EAC.

Suppose P plays a strategy which is almost-everywhere copycat at $\mathbf{m_o}(\vec{v})$. The two arenas $A \mathbin{@} \mathbf{m_o}(\vec{v})$ and $A \mathbin{@} \mathbf{m_p}(\vec{v})$ are shown below.



The idea is that except for finitely many subtrees of the moves in question, P's behaviour is "everywhere copycat" at $\mathbf{m_o}(\vec{v})$ i.e. P simply copies O's move between two isomorphic subarenas (which are shaded in the figure).

Since the notion of EAC is only defined for innocent strategies, we will sometimes just say "EAC strategy" instead of "EAC innocent strategy".

For a specific P-view $\vec{v}$ of such a function f, we will say that $t_{\vec{v}}$ and $o_{\vec{v}}$ are *valid* copycat threshold and offset if f satisfies the conditions (i)-(iv) of AC at that P-view with those particular values. Valid copycat thresholds are not unique, as the following lemma shows.

Lemma 62 *If $f : \mathbb{N}^* \to \mathbb{N} \times \mathbb{N}_0$ is an innocent strategy in economical form, and f is defined and AC at some P-view $\vec{v}$ with copycat threshold and offset $t_{\vec{v}}$ and $o_{\vec{v}}$ respectively, then for any $t' \geq t_{\vec{v}}$, f is also AC at the P-view $\vec{v}$ with threshold and offset t' and $o_{\vec{v}}$ respectively. That is, any value larger than a valid copycat threshold is still a valid threshold for a specific P-view (with the same offset).*

Thus at each P-view of an EAC strategy there will be a *least copycat threshold*, the least value for $t_{\vec{v}}$ which is still a valid threshold. However, the existence of a computable function giving valid thresholds does not imply the computability of the function giving least thresholds.

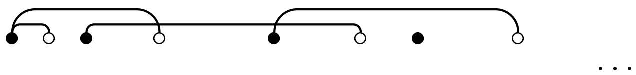
The following proof is important for what follows, and also serves as an illustration of the definition of EAC strategies.

Lemma 63 *For any arena $A = \langle A_1, \ldots, A_n \rangle$ the strategies $\pi^A_{A_i}$ are EAC. Hence for any arena A the strategy id_A is EAC.*

PROOF. The innocent function of $\pi^A_{A_i}$ is shown in section 2.3. The economical form is therefore some subset (depending on the arenas) of the function f defined by:

$$\begin{aligned} f(\varepsilon) &= (i, 0) \\ f(a) &= (a + n, 1) \\ f(\vec{s} \cdot a) &= (a, 1) \text{ for nonempty } \vec{s} \end{aligned}$$

A typical P-view of a legal position of this strategy might be



$\varepsilon \quad i \quad ia \; (a+n) \; (a+n)b \; iab \; iabc \; (a+n)bc$

Hence $\mathbf{m_o}(\varepsilon) = \varepsilon, \mathbf{m_p}(\varepsilon) = i$; for even-length $\vec{s}$, $\mathbf{m_o}(a \cdot \vec{s}) = (a + n) \cdot \vec{s}$ and $\mathbf{m_p}(a \cdot \vec{s}) = i \cdot a \cdot \vec{s}$; for odd-length $\vec{s}$, $\mathbf{m_o}(a \cdot \vec{s}) = i \cdot a \cdot \vec{s}$ and $\mathbf{m_p}(a \cdot \vec{s}) = (a + n) \cdot \vec{s}$.

Now if $B = A \Rightarrow A_i$ then $B \mathbin{@} \langle i \rangle = A_i = B^{>n}$. Thus this strategy is AC at ε with $t_\varepsilon = 0$ and $o_\varepsilon = -n$, and in fact is EC everywhere else. Everything in sight is computable and so the strategy is EAC.

The result that identity strategies are EAC follows from their definition as tupled projection strategies. $\square$

It remains to show the following:

Lemma 64 *If σ is an EAC strategy on $A \Rightarrow B$ and τ is an EAC strategy on $B \Rightarrow C$ then the composite strategy $\sigma\,;\tau$ is an EAC strategy on $A \Rightarrow C$.*

PROOF. We compose EAC strategies in the standard way (i.e. by "parallel composition plus hiding"). The composite strategy is innocent and recursive by Lemmas 26 and 33. It remains to give an algorithm that returns a threshold and an offset, which satisfy the four conditions of almost-everywhere copycat strategies, at every sequence where the composite strategy is defined. The algorithm and the proof, which take some six pages when written out in full, are very technical, and some care is needed in the case analysis. As this paper has a sequel [13] (an extended abstract appears as [12]) in which we develop the idea of almost-everywhere copycat strategies in more depth, to obtain a universal model for the Böhm tree λ-theory, it seems to us more sensible to present the material there as part of a systematic account. A full version of the proof can also be found in the first author's technical report [11]. $\square$

Definition 65 The *category of arenas and EAC strategies*, $\mathbb{A}_{\mathrm{EAC}}$, has r.e. arenas as objects and EAC strategies on $A \Rightarrow B$ as morphisms from A to B.

Theorem 66 *$\mathbb{A}_{\mathrm{EAC}}$ is a cartesian closed category, in fact a lluf subcategory*[2] *of $\mathbb{A}_{\mathrm{REC}}$ with the same cartesian closed structure.*

PROOF. Certainly $\mathbb{A}_{\mathrm{EAC}}$ has the same objects as $\mathbb{A}_{\mathrm{REC}}$, and Lemma 63 shows that the same identity strategies are morphisms of $\mathbb{A}_{\mathrm{EAC}}$. Lemma 64 completes the proof that it is a category.

Lemma 63 also gives that the projections for $\mathbb{A}_{\mathrm{REC}}$ are also in $\mathbb{A}_{\mathrm{EAC}}$, and recall that the evaluation morphism is just given by the identity strategy, so cartesian closure of $\mathbb{A}_{\mathrm{EAC}}$ follows from that of $\mathbb{A}_{\mathrm{REC}}$. $\square$

5.2 The Model $\mathcal{D}_{\mathrm{EAC}}$

The arena U is an object of $\mathbb{A}_{\mathrm{EAC}}$, as are the morphisms *Fun* and *Gr* (since they are specified by the identity strategy on U). Thus $\mathbb{A}_{\mathrm{EAC}}$ has the same reflexive object as $\mathbb{A}$. This allows us to define a new λ-algebra as follows:

Definition 67 We write $\mathcal{D}_{\mathrm{EAC}}$ for the λ-algebra $\mathcal{M}(\mathbb{A}_{\mathrm{EAC}}, U, \mathit{Fun}, \mathit{Gr})$.

Since the structure of $\mathbb{A}_{\mathrm{EAC}}$ is the same as that of $\mathbb{A}_{\mathrm{REC}}$, we know that the elements of the model are a subset of those of $\mathcal{D}_{\mathrm{REC}}$, and the function $[\![-]\!]_{-}$ of $\mathcal{D}_{\mathrm{EAC}}$ is the same as that of $\mathcal{D}_{\mathrm{REC}}$. Hence,

Theorem 68 *$\mathcal{D}_{\mathrm{EAC}}$ is a $\lambda\eta$-algebra with local structure equal to the λ-theory $\mathcal{H}^*$ (and local order structure as described for $\mathcal{D}$ in Corollary 57).*

[2] by a *lluf subcategory* of a category $\mathbb{C}$ we mean a subcategory of $\mathbb{C}$ which has the same class of objects of $\mathbb{C}$. See e.g. [5, §2.3].

The aim was that the EAC strategies should be those that correspond to terms of the λ-calculus. We know that every element of the model $\mathcal{D}$ which is the denotation of some term must be EAC, by the above comments. We now show that the converse holds.

Lemma 69 *Given an EAC innocent strategy on the arena U with economical form f there is some closed term s of the λ-calculus such that (as a labelling function $\mathbb{N}^* \rightharpoonup \mathbb{N} \times \mathbb{N}_0$) $VFF_\varepsilon(s) = f$.*

PROOF. Suppose that the copycat threshold and offset of f at the P-view coded by $\vec{v}$ are $t_{\vec{v}}$ and $o_{\vec{v}}$ respectively.

Let the set $X \subseteq \mathbb{N}^*$ be defined inductively by:

$$\varepsilon \in X$$
$$\vec{v} \in X \implies \forall i.\ 1 \leq i \leq t_{\vec{v}} \implies \vec{v} \cdot i \in X.$$

Then X has the following properties:

(1) If $\vec{v} = \vec{u} \cdot i$ with $\vec{u} \in X$ and $\vec{v} \notin X$ then $i > t_{\vec{u}}$, so $f(\vec{v}) = (i - o_{\vec{u}}, 1)$ (by clause (ii) of the definition of AC).
(2) If $\vec{v} = \vec{u} \cdot i$ with $\vec{u} \notin X$ then $f(\vec{v}) = (i, 1)$ (since f must be EC at $\vec{u}$, so by clause (ii) of the definition of EC).

We define the labelling function of Böhm-like tree A (by what we call *copycat collapse*) as follows. The domain of A (i.e. the shape of the partially labelled tree A) is the set X.

For any sequence $\vec{v} = \langle v_1, \ldots, v_p \rangle \in X$ we define $A(\vec{v})$ by:

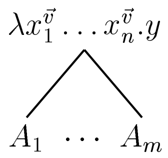
(1) If $\vec{v} \notin \mathrm{dom}(f)$ then $A(\vec{v})$ is undefined (the partially-labelled tree has label $\perp$ at this node).
(2) If $f(\vec{v}) = (i, r)$ then $A(\vec{v}) = \lambda x_1^{\vec{v}} \ldots x_n^{\vec{v}}.x_i^{\langle v_1, \ldots, v_{p-r} \rangle}$, where $n = t_{\vec{v}} - o_{\vec{v}}$.

We make use of the characterisation of those Böhm-like trees which are the Böhm trees of terms, which can for instance be found in [3, 10.1.23]. This states that a Böhm-like tree A is the Böhm tree of some term if and only if A has only finitely many free variables and the labels of A are given recursively.

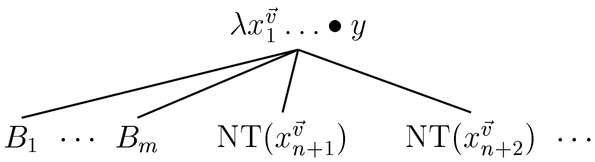
Now there are no free variables in the Böhm-like tree A (since in the second clause above, if $f(\langle v_1, \ldots, v_p \rangle) = (i, r)$ then $r \leq p$). Since the functions $\vec{v} \mapsto t_{\vec{v}}$ and $\vec{v} \mapsto o_{\vec{v}}$ are recursive, and f is recursive, and our procedure for computing $A(\vec{v})$ from these is clearly recursive, A itself must be a recursive labelling function. Hence there is some term s with $\mathrm{BT}(s) = A$.

Now we prove that $\mathrm{VFF}_\varepsilon(s) = f$, and to examine the former we consider $\mathrm{NT}(s)$ and use Lemma 53. Now the relationship between the Böhm and Nakajima trees of a term can be deduced fairly easily from the definition. Suppose that $\mathrm{NT}(s)$ has had bound variables renamed so that the i^{th} abstracted variable at the node coded by $\vec{v}$ is $x_i^{\vec{v}}$ and that this renamed tree has labelling function B (so that the abstracted variables at each label of A match the first few at the same label of B).

Then at any node where A is unlabelled $\perp$, so is $\mathrm{NT}(s)$. At a node of the tree labelled by A of the form



for some trees $A_1, \dots, A_m$, we can deduce that the tree labelled by B has the corresponding subtree of the form



for some trees $B_1, \dots, B_m$.

Now we show by case analysis on $\vec{v}$ that $f(\vec{v}) = \mathrm{VFF}_\varepsilon(s)(\vec{v})$. There are four cases:

(1) $\vec{v} \in X$ but $\vec{v} \notin \mathrm{dom}(f)$. In this case A is unlabelled at the node coded by $\vec{v}$, so $\vec{v} \notin \mathrm{dom}(B)$, and so by Lemma 53 $\vec{v} \notin \mathrm{dom}(\mathrm{VFF}_\varepsilon(s))$.
(2) $\vec{v} \in X$ and $f(\vec{v}) = (i, r)$. Then we know that $A(\vec{v}) = \lambda x_1^{\vec{v}} \dots x_n^{\vec{v}}.x_i^{\langle v_1, \dots, v_{p-r} \rangle}$, so $B(\vec{v}) = \lambda x_1^{\vec{v}} \dots \bullet\, x_i^{\langle v_1, \dots, v_{p-r} \rangle}$. Hence, by Lemma 53, $\mathrm{VFF}_\varepsilon(s)(\vec{v}) = (i, r)$.
(3) $\vec{v} \notin X$ but $\vec{v} = \vec{u} \cdot i$ with $\vec{u} \in X$. Then we know that $f(\vec{v}) = (i - o_{\vec{u}}, 1)$. Also we know that $A(\vec{u}) = \lambda x_1^{\vec{u}} \dots x_n^{\vec{u}}.y$, and has m descendants, for some variable y and where $m = t_{\vec{u}}$ and $n = t_{\vec{u}} - o_{\vec{u}}$. Therefore by examining the diagrams comparing nodes of Böhm and Nakajima trees above, we see that $B(\vec{v}) = \lambda x_1^{\vec{v}} \dots \bullet\, x_{i-m+n}^{\vec{u}}$. Hence, by Lemma 53, $\mathrm{VFF}_\varepsilon(s)(\vec{v}) = (i - m + n, 1) = (i - o_{\vec{u}}, 1)$.
(4) $\vec{v} \notin X$ and $\vec{v} = \vec{u} \cdot i$ with $\vec{u} \notin X$. Then we know that $f(\vec{v}) = (i, 1)$. Also we know that the node coding $\vec{v}$ is in one of the trees $\mathrm{NT}(y)$ for some variable y, so $B(\vec{v}) = \lambda x_1^{\vec{v}} \dots \bullet\, x_i^{\vec{u}}$. Hence, by Lemma 53, $\mathrm{VFF}_\varepsilon(s)(\vec{v}) = (i, 1)$.

Hence in every case $\mathrm{VFF}_\varepsilon(s)(\vec{v}) = f(\vec{v})$. $\square$

Combining Lemma 69 with the Exact Correspondence Theorem gives that every member of the homset $\mathrm{Hom}_{\mathbb{A}_{\mathrm{EAC}}}(\mathbf{1}, U)$ is the denotation of some term. Hence,

Theorem 70 *$\mathcal{D}_{\mathrm{EAC}}$ forms a universal $\lambda\eta$-algebra.*

5.3 Extensionality of $\mathcal{D}_{\mathrm{EAC}}$

Finally, we show that the model $\mathcal{D}_{\mathrm{EAC}}$ is extensional. In fact we show the stronger property of order-extensionality. A direct proof can be given, by means of a "Separation Lemma" for EAC strategies, but in this work we give a much shorter proof which relies on known results of the λ-calculus and the universality of the model.

Theorem 71 *$\mathcal{D}_{\mathrm{EAC}}$ is* order-extensional. *That is, for all σ and τ in $\mathcal{D}_{\mathrm{EAC}}$,*

$$(\forall \rho \in \mathcal{D}_{\mathrm{EAC}}.\ \sigma \bullet \rho \subseteq \tau \bullet \rho) \iff \sigma \subseteq \tau.$$

Hence $\mathcal{D}_{\mathrm{EAC}}$ is an extensional $\lambda\eta$-model.

PROOF. The implication ($\Longleftarrow$) follows from the obvious monotonicity of $\bullet$.

To show ($\Longrightarrow$), let σ, τ be elements of $\mathcal{D}_{\mathrm{EAC}}$. Then, by Universality, $\sigma = [\![s]\!]$ and $\tau = [\![t]\!]$, for some closed terms s and t. Assume that $\sigma \not\subseteq \tau$. By the Exact Correspondence Result, $\mathrm{NT}(s) \not\subseteq \mathrm{NT}(t)$, where the ordering on Nakajima trees is inclusion of the labelling function (modulo renaming of bound variables).

Now we use standard results, commonly found as part of the proof of Böhm's Theorem for the λ-calculus. We will quote the versions appearing in [3, 10.2-10.4].

The fact that $\mathrm{NT}(s) \not\subseteq \mathrm{NT}(t)$ means that $\mathrm{BT}(s) \not\sim_\alpha \mathrm{BT}(t)$ for some α of minimal length (see [3, 10.2.21] for the definition of $\sim_\alpha$ and [3, 10.2.31] for the proof). Then by [3, 10.3.13, 10.4.1(ii) and 10.3.4], there is a sequence of terms $u_1, \ldots, u_n$ such that $\lambda \vdash s u_1 \ldots u_n = I$ and $t u_1 \ldots u_n$ is unsolvable. Hence

$$[\![s]\!] \bullet [\![u_1]\!] \bullet \cdots \bullet [\![u_n]\!] = [\![I]\!] \not\subseteq \bot = [\![t]\!] \bullet [\![u_1]\!] \bullet \cdots \bullet [\![u_n]\!].$$

But $\bullet$ is monotone so $\sigma \bullet [\![u_1]\!] \not\subseteq \tau \bullet [\![u_1]\!]$. $\square$

Remark 72 This proof uses ideas from the standard proof of Böhm's Theorem. If we avoid this, taking the longer route to the proof, using properties of

EAC strategies and a “Separation Lemma”, we can in fact produce a semantic proof of Böhm’s Theorem instead. Details can be found in [11].

6 Further Work

A Böhm Tree Model

We have defined an EAC strategy to be one such that there exists a computable function specifying copycat threshold and offset, and the strategy obeys some constraints determined by this function. We are not saying that every EAC strategy comes with such a function specified.

This does make a difference because if we are just given a strategy, even if told that it is EAC, we cannot effectively determine any such function (and additionally it is not even semi-decidable whether a given innocent strategy is EAC). Now in order to turn an EAC strategy into a term we need to know the copycat thresholds and offsets, and different thresholds give rise to different terms (the difference is η-conversion).

So we might wish to define an *explicit EAC strategy*, which is one which comes with a threshold/offset function. (We note that the offset can be effectively and simply recovered if we know the threshold, so we could miss that out if we wished). Although EAC strategies compose, and the proof that they do so is constructive in the sense that given threshold functions of the composed strategies we have a procedure for calculating the threshold function of the result, composition of explicit EAC strategies presents some technical problems. To make a CCC of such strategies we need to make some fundamental alterations to the structure, in particular using different objects. (One consequence is that there is not an object U with $U = U \Rightarrow U$, which one would hope for in the light of the comments below). The details of this construction are to presented in a sequel, [13], of which an extended abstract appears as [12].

This makes a finer distinction between strategies, because given a threshold at a P-view of an EAC strategy it is still valid to claim any higher threshold. In terms of terms this is because for any term s $[\![s]\!] = [\![\lambda x.sx]\!]$ for fresh x. So two strategies which have the same moves may be different because they have different threshold functions, corresponding to different η-expansions. The CCC of explicit EAC strategies will be a λ-algebra which does not validate η-conversion. It turns out that it has the same local structure as the Böhm tree model $\mathcal{B}$, and we can prove a result akin to the Exact Correspondence Theorem, a correspondence between the standard variable-free form of Böhm trees and their explicit EAC denotation.

Questions which arise from this are: can we add the copycat threshold information into the moves of the game, recovering a "non-economical" form which would be not quite so close to a term model. A fairly contrived way exists, but maybe it could be made more natural. Related to this, we note that it is more economical to remove all the copycat parts of explicit EAC strategies, since they are determined anyway by the threshold and offset. Is there a smart representation, and does it help with an efficient composition algorithm?

Finally, is there something else we can do in order to model exactly the smaller theory $\mathcal{H}$ (the minimal sensible theory)? More understanding of the difference between $\mathcal{B}$ and $\mathcal{H}$ is needed.

The Extensional Collapse of $\mathcal{D}_{\mathrm{REC}}$ and Infinitary λ-calculus

In order to obtain an extensional and universal model, we restricted the morphisms of $\mathbb{A}_{\mathrm{REC}}$ directly to the EAC strategies, which corresponds to a reformulation of the conditions for a Böhm-like tree to come from a term, after infinite η-expansion and in the language of strategies. An interesting line of investigation is whether $\mathbb{A}_{\mathrm{EAC}}$, or at least $\mathcal{D}_{\mathrm{EAC}}$, can be reached by some algebraic construction from $\mathbb{A}_{\mathrm{REC}}$.

A standard approach to getting an extensional game model is to take an *extensional collapse*, i.e. quotient by observational equivalence. If we can decide on a good definition of observational equivalence for the model $\mathcal{D}$ (more difficult than usual because we don't have a clear input/output behaviour) it should be the case that all EAC strategies live in different equivalence classes. However there will probably be equivalence classes not inhabited by any EAC strategy — the innocent strategy corresponding to the term intuitively given by $\lambda x_1 x_2 x_3 \ldots \bullet\ x_2 x_1 x_4 x_3 x_6 x_5 x_8 x_7 \ldots$ will be one. This term is not an unreasonable one for some type of infinitary lambda-calculus, which one might think of as reasoning about streams of processes (which themselves act on streams of processes) or perhaps as a non-terminating process in the language of λ-calculus.

It looks like the EAC condition is imposing extensionality and an extra "finiteness" condition which is not inherent in innocent strategies. It would be interesting to find out exactly what sort of infinitary language is being modelled by the extensional collapse of $\mathcal{D}_{\mathrm{REC}}$, and if possible to split the EAC condition into two distinct parts (extensionality and finiteness) in order to understand it better.

Nakajima introduced a brand of infinite terms when presenting the η-expanded trees in [16]. This bears further investigation, although his method (each term is a sequence of terms which approximate it) is based on the D_∞ model and to some extent is looking from the wrong angle.

Infinitary lambda-calculi are a topic of current research interest. In [10] Kennaway *et al.* describe a uniform method for constructing infinite terms, identifying three independent ways in which terms can be infinite (infinite abstraction, infinite depth and infinite application). By describing 8 metrics on parse trees for terms and constructing the metric completions they give calculi with all possible combinations of these infinite phenomena. However it looks like the intuitive infinite calculus we are after is none of these.

Another approach is described by Berarducci in [4], but the focus is on infinite term rewriting and again the set of infinite terms seems not to be what we want.

John Longley has commented that the space of increasing sequences of Böhm trees might perhaps correspond to the recursive innocent strategies. This could be the way to approach it. The direction we would like to pursue is to identify exactly what the language we are thinking about is, possibly to find abstract categorical properties of models of it, and to check that the extensional collapse of $\mathcal{D}_{\mathrm{REC}}$ is a universal, fully abstract model. This language might have some interesting features (solvability is not equivalent to the existence of head normal forms, for example).

A Proof of the Exact Correspondence Theorem

We aim to show: if $s \in \Lambda$ with free variables in $\Delta = \langle v_k, \ldots, v_1 \rangle$ then $[\![s]\!]_\Delta = \{\mathrm{VFF}_\Delta(s)\}^k$ when the former is considered in economical form and the latter as a labelling function.

The two sides are partial functions from $\mathbb{N}^*$ to $\mathbb{N}_0 \times \mathbb{N}$ so we need to show that for all $\vec{\alpha} \in \mathbb{N}^*$,

$$[\![s]\!]_\Delta(\vec{\alpha}) = \{\mathrm{VFF}_\Delta(s)\}^k(\vec{\alpha}).$$

We prove this by induction on the length of $\vec{\alpha}$ for all terms s and contexts Δ simultaneously. Notice that the variables of Δ are labelled in reverse order again, this is for convenience in the proof and irrelevant to the statement of the theorem.

Base Case: If s is unsolvable then both sides are the empty function.

If s is solvable, then either the head variable is free or not. Let us first suppose that $s = \lambda x_1 \ldots x_n.x_j s_1 \ldots s_m$, and $\Delta = \langle v_k, \ldots, v_1 \rangle$. Then

$$[\![s]\!]_\Delta = \underbrace{\Lambda(\cdots\Lambda(\Lambda}_{n\ \Lambda\text{'s}}([\![x_j s_1 \ldots s_m]\!]_{\Delta\cdot\langle x_1,\ldots,x_n\rangle}); Gr); Gr\cdots); Gr$$

but since as strategies $Gr = \mathrm{id}_U$ and $\Lambda(f) = f$ we can ignore these for the purposes of calculating the denotation (as long as we keep track of the domain and codomain of each strategy so we know which bits to hide when composing). Now,

$$[\![x_j s_1 \ldots s_m]\!]_{\Delta\cdot\langle x_1,\ldots,x_n\rangle} = (\Pi_{x_j}^{\Delta\cdot\langle x_1,\ldots,x_n\rangle} \bullet [\![s_1]\!]_{\Delta\cdot\langle x_1,\ldots,x_n\rangle}) \cdots \bullet [\![s_m]\!]_{\Delta\cdot\langle x_1,\ldots,x_n\rangle}$$

where each term is a map from U^{n+k} to U.

We will examine the nature of these sorts of compositions in a moment, but at this stage all we need to note is that the first P-move in the calculation of such a composition is the first P-move of the first term, which is $\Pi_{x_j}^{\Delta\cdot\langle x_1,\ldots,x_n\rangle}$. This has first P-move $k+j$, justified by the initial O-move, and so $[\![s]\!]_\Delta(\varepsilon) = (k+j, 0)$. This move is visible in the composition.

On the other hand, $\mathrm{VFF}_\Delta(s) = \{\mathrm{VFF}_{\Delta\cdot\langle x_1,\ldots,x_n\rangle}(x_j s_1 \ldots s_m)\}^n$.
Now $\mathrm{VFF}_{\Delta\cdot\langle x_1,\ldots,x_n\rangle}(x_j s_1 \ldots s_m)$ has root node $(n-j+1, 1)$, from the definition of VFF, and so the same tree operated on by $\{-\}^n$ will have root node $(j, 0)$. Thus $\{\mathrm{VFF}_\Delta(s)\}^k(\varepsilon) = (k+j, 0)$.

The remaining case is when $s = \lambda x_1 \ldots x_n.v_j s_1 \ldots s_m$ and $\Delta = \langle v_k, \ldots, v_1 \rangle$ and this is entirely similar with both sides mapping ε to $(k-j+1, 0)$.

Inductive Case: Suppose that the result holds for all terms s, all contexts Δ containing the free variables of s, for all sequences $\vec{\alpha}$ up to length l.

Again either s is unsolvable, in which case the result is trivial, or s has HNF $\lambda x_1 \ldots x_n.x_j s_1 \ldots s_m$ or $\lambda x_1 \ldots x_n.v_j s_1 \ldots s_m$ for $v_j \in \Delta$. Again, the last two cases are similar and we will be able to prove them together.

To do so we will prove the result that $[\![v_j s_1 \ldots s_m]\!]_\Delta(\vec{\alpha}) = \{\mathrm{VFF}_\Delta(v_j s_1 \ldots s_m)(\vec{\alpha})\}^k$ for sequences $\vec{\alpha}$ up to length $l+1$ and then note that if $s = \lambda x_1 \ldots x_n.t$ (where t has variable at the head) then for $\vec{\alpha}$ up to length $l+1$,

$$\begin{aligned}
[\![s]\!]_\Delta(\vec{\alpha}) &= [\![t]\!]_{\Delta\cdot\vec{x}}(\vec{\alpha}) \\
&= \{\mathrm{VFF}_{\Delta\cdot\vec{x}}(t)\}^{k+n}(\vec{\alpha}) \\
&= \{\{\mathrm{VFF}_{\Delta\cdot\vec{x}}(t)\}^n\}^k(\vec{\alpha}) \qquad \text{since } \{\{p\}^m\}^n = \{p\}^{m+n} \\
&= \{\mathrm{VFF}_\Delta(\lambda x_1 \ldots x_n.t)\}^k(\vec{\alpha})
\end{aligned}$$

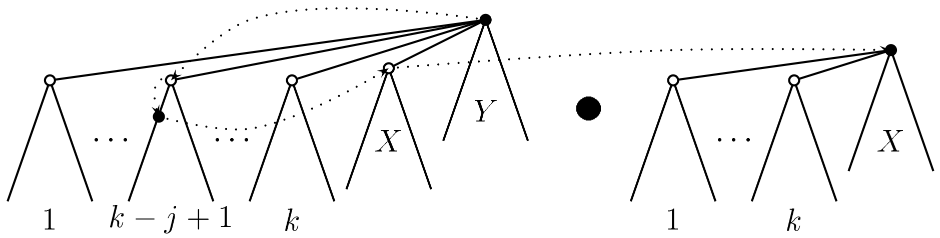
which implies the required result.

So suppose $t = v_j s_1 \ldots s_m$. Then the induction hypothesis gives that $[\![s_i]\!]_\Delta(\vec{\alpha}) = \{\mathrm{VFF}_\Delta(\vec{\alpha})\}^k$ for sequences $\vec{\alpha}$ up to length l. Now,

$$[\![t]\!]_\Delta = (\Pi^\Delta_{v_j} \bullet [\![s_1]\!]_\Delta) \cdots \bullet [\![s_m]\!]_\Delta$$

Now we have to take a detailed look at the nature of the above application. Recall that for $s, t : A \to U$, $\sigma \bullet \tau = \langle \sigma; \mathit{Fun}, \tau \rangle; \mathrm{eval}_{U,U}$ and that as a strategy *Fun* does nothing.

Let us restrict our attention to the case $m = 1$ so that $t = v_j s'$. The trees we compose look like this:



The application consists of composing the above pair with the eval strategy, which has the effect of copying moves made in one X component to the other and hiding both, and allows moves played in the context subtrees $1, \ldots, k$ to be identified and made visible.

Further, we know that on the left-hand tree we are playing the strategy $\Pi^\Delta_{v_j}$ which has initial move $k - j + 1$ and thereafter copies moves from the $X \Rightarrow Y$ component into the $k - j + 1$ context component.

Now a strategy $\sigma = [\![s']\!]_\Delta$ is played on the right-hand tree. The diagram above shows the first few moves of the composition. The composite strategy makes an initial move of $k - j + 1$ and then in response to the move $(k - j + 1)1$ copies across to the X component on the left, thence to the X component on the right. Thereafter any moves played in the X component of the right-hand side are copied over to the left-hand side, and then to the $k - j + 1$ component. These moves are visible on the left-hand side.

In summary, moves played by σ in X appear in the $k - j + 1$ component. Other moves made by σ, i.e. those in the first k subtrees, appear visible in the same subtree on the left-hand side.

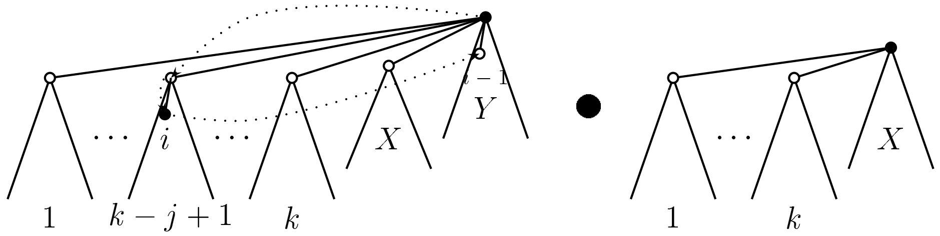
Now the induction hypothesis is that down to depth l of the economical form, the strategy σ is $\{\mathrm{VFF}_\Delta(s')\}^k$. By the definition of $\Pi^\Delta_{v_j}$ the initial move of the composition is $k - j + 1$ so the root of the economical form tree is $(k - j + 1, 0)$.

Let us consider what the first subtree of the economical form of this composition will be, down to depth l of that subtree (depth $l+1$ of the whole tree):

There are two cases to consider. In the tree $\mathrm{VFF}_\Delta(s')$ any node at depth d labelled (i,d) will be relabelled by the first clause of the definition of $\{-\}^k$ to $(i+k,d)$. These will appear as moves justified by the root of the tree and appearing in the X part of the tree on the right. Nodes with second component strictly less than their depth are unaffected. Then the copycat strategy $\Pi^\Delta_{v_j}$ reproduces these in the $k-j+1$ subtree. Hence the nodes of the economical form will be precisely those of $\mathrm{VFF}_\Delta(s')$, at least to depth l.

However there might be also nodes at depth d labelled $(i,d+1)$ in $\mathrm{VFF}_\Delta(s')$, but since all the free variables of s' are in Δ we can be sure that $i \le k$. Hence these nodes are mapped by $\{-\}^k$ to $(k-i+1,d)$. As a strategy these are moves in the context subtrees, and in the composition they will appear as children of the very first move, hence justified by a move two before the root of X. Hence the economical form will have corresponding label $(k-i+1,d+1)$.

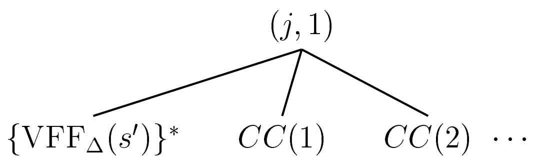
That completes the description of the first subtree of the economical form of the composition.



If Opponent's response to Proponent's first move is $(k-j+1)i$ for $i>1$ then this move is copied into the Y component on the left-hand side and moves in response are copied back-and-forth between the subtree i of Y and the $k-j+1$ context subtree. σ is not activated, and because the left-hand side has the X component hidden, these copied moves appear in the composition to be from the $k-j+1$ subtree to the $k+i-1$ subtree.

Thus the economical form of the composition has the following i^{th} subtree, for $i>1$: The root node is labelled $(k+i-1,1)$ corresponding to the move shown in the diagram above, and since we play copycat thereafter all other moves are justified by the one three beforehand in the P-view, hence the j^{th} child of any node is labelled $(j,1)$. That is, this subtree is the same as $CC(k+i-1)$.

Recall that $\mathrm{VFF}_\Delta(t) =$



Thus $\{\mathrm{VFF}_\Delta(t)\}^k =$

$$\begin{tikzcd}[column sep=small] & (k-j+1,0) \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\ \{\mathrm{VFF}_\Delta(s')\}^\times & CC(k+1) & CC(k+2)\ \cdots \end{tikzcd}$$

where $\{\mathrm{VFF}_\Delta(s')\}^\times$ is the same as $\mathrm{VFF}_\Delta(s')$ except that

(1) nodes at depth d labelled $(i, d+1)$ for $i \leq k$ are relabelled $(k-i+1, d+1)$;
(2) nodes at depth d labelled $(i, d+1)$ for $i > k$ are relabelled $(i-k, d+1)$.

However since all the free variables of s' are in Δ, we can be sure that the second case never happens. And that leaves what we have described for the economical form of the composition, down to depth l for each subtree, i.e. depth $l+1$ for the whole tree.

Finally, we claim that the generalisation for any m clearly works in the same way.

Acknowledgements

The work presented in this paper was undertaken as part of the first-named author's EPSRC-funded doctoral research, and also under grant GR/L27787.

We are most grateful to John Longley, whose care in looking at a very early draft of this material went beyond the call of duty. From him we have received invaluable feedback and ideas.

The notations $A\,@\,m$ and $A^{>n}$ are due to Dominic Hughes, whose ideas on the optimal way to present arenas and strategies have been most useful. We are also grateful to Guy McCusker for many challenging discussions.

References

[1] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF (extended abstract). In *Theoretical Aspects of Computer Software: TACS'94, Sendai, Japan*, pages 1–15. Springer-Verlag, 1994. LNCS Vol. 789.

[2] S. Abramsky and G. A. McCusker. Games and full abstraction for the lazy λ-calculus. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 234–243. IEEE Computer Society Press, 1995.

[3] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 2nd edition, 1984.

[4] A. Berarducci. Infinite lambda-calculus and non-sensible models. In *Logic and Algebra*, volume 180 of *Lecture Notes in Pure and Applied Mathematics*, pages 339–378. Marcel Dekker Inc., 1996.

[5] R. L. Crole. *Categories For Types.* Cambridge mathematical textbooks. Cambridge University Press, 1993.

[6] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation. *Indag. Math.*, 41:381–392, 1972.

[7] P. Di Gianantonio, G. Franco, and F. Honsell. Games semantics for untyped λ-calculus. In J. Y. Girard, editor, *4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings*, number 1581 in Lecture Notes in Computer Science. Springer-Verlag, 1999.

[8] J. M. E. Hyland. A syntactic characterization of the equality in some models of the lambda calculus. *Journal of the London Mathematical Society*, 2(12):361–370, 1976.

[9] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. To appear in Information and Computation, `ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Luke.Ong/`, 1994.

[10] J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. Infinitary lambda calculus. *Theoretical Computer Science*, 175:93–125, 1997.

[11] A. Ker. Universal game models of the Nakajima tree and Böhm tree λ-theories. Technical Report, available from `http://web.comlab.ox.ac.uk/oucl/work/andrew.ker/`, 1999.

[12] A. Ker, H. Nickau, and C.-H. L. Ong. A universal innocent game model of the Böhm tree lambda theory. In *Computer Science Logic: Proceedings of the 8th Annual Conference of the EACSL, Madrid, Spain, September 1999*, number 1683 in Lecture Notes in Computer Science, pages 405–419. Springer-Verlag, 1999.

[13] A. Ker, H. Nickau, and C.-H. L. Ong. A universal innocent model of the Böhm tree lambda theory. Submitted for publication, currently available from `http://web.comlab.ox.ac.uk/oucl/work/andrew.ker/`, 1999.

[14] S. Mac Lane. *Categories for the Working Mathematician.* Springer-Verlag, 1971.

[15] G. A. McCusker. *Games and Full Abstraction for a Functional Metalanguage with Recursive Types.* Cambridge University Press, 1998.

[16] R. Nakajima. Infinite normal forms for the λ-calculus. In *Proc. Symp. λ-calculus and Computer Science Theory*, pages 62–82. Springer-Verlag, 1975.

[17] H. Nickau. *Hereditarily Sequential Functionals: A Game-Theoretic Approach to Sequentiality.* Shaker-Verlag, 1996. Dissertation, Universität Gesamthochschule Siegen. Shaker-Verlag, 1996.

[18] C. P. Wadsworth. *Semantics and Pragmatics of the λ-calculus.* PhD thesis, University of Oxford, 1971.

[19] C. P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the λ-calculus. *SIAM Journal of Computing*, 5:488–521, 1976.