Innocent Game Models of Untyped Lambda Calculus

Andrew D. Ker

University College, Oxford

Submitted for the degree of Doctor of Philosophy,
Michaelmas Term 2000

Oxford University Computing Laboratory
Programming Research Group

Innocent Game Models of Untyped Lambda Calculus

Andrew D. Ker
University College, Oxford
Doctor of Philosophy, Michaelmas Term 2000

Abstract

This thesis is a detailed examination of the application of game semantics to constructing denotational models of the pure untyped λ-calculus. Game semantics is a fairly recent technique, using a formal setting for interaction to model sequential programming languages in an accurate way.

We use a modification of the "innocent" games first used to give a fully-abstract syntax-independent model of PCF; the only difference is that in our setting the distinction between "question" and "answer" moves is removed. Many of the standard results for PCF games carry through into this setting. Cartesian closed categories of arenas and innocent strategies are constructed, leading to $\lambda\eta$-algebras $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$. By a method of approximation, these are shown to be sensible models (i.e. all unsolvable terms are equated) but they contain many undefinable elements and are not λ-models.

By introducing a new "economical" representation of innocent strategies we are able to prove a precise syntactic connexion between a term and its denotation. This leads to a new class of innocent strategies, the effectively almost-everywhere copycat (EAC) strategies, which also inhabit a cartesian closed category and hence give rise to a new $\lambda\eta$-algebra $\mathcal{D}_{\mathrm{EAC}}$. This is both sensible and universal: every element is the denotation of some term. To our knowledge, other than term models, the universality result is the first of its kind for the untyped λ-calculus. The equational theory induced by the above game models is shown to be the same as that of Scott's D_∞ models, the maximal consistent sensible theory $\mathcal{H}^*$.

In contrast to $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, $\mathcal{D}_{\mathrm{EAC}}$ is order-extensional. We give both a short syntactic proof, and a longer semantic proof; the latter is of interest because it can be extended to give a semantic proof and slight generalisation of Böhm's Theorem.

Finally, we construct a game model which does not validate η-conversion. By introducing the effectively and explicitly almost-everywhere copycat (EXAC) strategies, and (with some technical difficulties) constructing a cartesian closed category, we define a sensible model $\mathcal{D}_{\mathrm{XA}}$. Results analogous to those for the other game models show that the local structure of $\mathcal{D}_{\mathrm{XA}}$ is the λ-theory $\mathcal{B}$ (which equates two terms if and only if they have the same Böhm tree). $\mathcal{D}_{\mathrm{XA}}$ is universal, but not even weakly extensional.

Acknowledgements

I am greatly indebted to my supervisor, Luke Ong. He suggested the study of game models for untyped λ-calculus in my first term as a graduate student, and encouraged that project — initially a small one — to grow into this thesis. This academic input has shaped the direction of my work. Additionally, I am very grateful to him for support and advice away from academic matters; his help has gone well beyond what one might expect of a supervisor.

Hanno Nickau has been my other co-author in the published parts of this work. His input was of enormous help in clarifying the ideas presented here, and his hard work and attention to detail has been of inestimable value. Dominic Hughes and Guy McCusker gave me much of their time when I was beginning graduate study, patiently explaining away some of my misconceptions.

The UK Engineering and Physical Sciences Research Council supported me financially with a Research Studentship, and I am also grateful to Merton College, Oxford, who augmented this support with a Senior Scholarship. The body of the work presented here was completed while at Merton, although it was started and finished at University College.

However this thesis also owes its existence to a large number of people who are not computer scientists — my friends at Oxford. My fellow orchestral musicians, musical administrators, choir members, and undergraduate contemporaries (who put me to shame by being extremely faithful about keeping in touch) are far too numerous to name. But their support was and is greatly valued, especially during my difficult third year when full recovery from glandular fever seemed as if it would never come. I owe you all a great deal.

Vitaï Lampada

There's a breathless hush in the close to-night —
Ten to make and the match to win —
A bumping pitch and a blinding light,
An hour to play and the last man in.
And it's not for the sake of a ribboned coat.
Or the selfish hope of a season's fame,
But his Captain's hand on his shoulder smote
"Play up! play up! and play the game!"

The sand of the desert is sodden red —
Red with the wreck of the square that broke; —
The Gatling's jammed and the colonel dead,
And the regiment blind with dust and smoke.
The river of death has brimmed his banks,
And England's far, and Honor a name,
But the voice of a schoolboy rallies the ranks,
"Play up! play up! and play the game!"

This is the word that year by year
While in her place the School is set
Every one of her sons must hear,
And none that hears it dare forget.
This they all with joyful mind
Bear through life like a torch in flame,
And falling fling to the host behind —
"Play up! play up! and play the game!"

Sir Henry Newbolt

To the memory of my grandfather, Henry Dixon, who taught me the above poem and a great deal more.

Contents

Chapter 1

Introduction

This thesis is a detailed study of the use of game semantics for denotational models of the pure untyped λ-calculus.

1.1 Background and Motivation

We begin with a brief look at the history of the untyped λ-calculus and game semantics, concentrating on game models of languages closely related to the untyped λ-calculus. For details, and a survey, of other models of untyped λ-calculus, see [Bar84].

The pure untyped λ-calculus arose in the work of Church [Chu32] and in an alternative form as combinatory logic due to Schönfinkel [Sch24] and Curry [Cur30]. Originally from the borders of foundations of mathematics and philosophy, the λ-calculus has become an important tool for theoretical computer science, embodying some standard computational principles, especially that of substitution and reduction as computation. Several programming languages have been inspired by the λ-calculus.

The need for *denotational semantics* of programming languages appeared in the 1960s, and was expressed in idealized form in the λ-calculus. This spurred Scott to find a nontrivial solution to the domain equation $D \cong [D \to D]$, one of the most important steps forward in theoretical computer science. The discovery of denotational models of the pure untyped λ-calculus enabled denotational models of "real" programming languages to be constructed. This is because Scott's method also gave a solution to the problem of denotational models for recursion and datatypes.

Scott's first models of the pure untyped λ-calculus, D_∞, were announced in [Sco69], rapidly followed by the simpler models $P\omega$ ([Plo72] and [Sco74]), and models which induce alternative inequational theories such as $\mathbb{T}^\omega$ of [Plo78]. These early models

are constructed from familiar mathematical objects such as complete partial orders, and the clever part had been Scott's use of traditional structures to form the recursive objects needed for denotational semantics. As the field of denotational semantics has progressed, however, it has become apparent that the traditional mathematical structures are not necessarily sufficient for useful denotational models.

A classic example was the search for a fully-abstract denotational model of Scott's language PCF [Sco93]. Apart from term models, fully-abstract models remained stubbornly elusive. Domain models based on Scott-continuous functions are not fully-abstract because they contain parallel elements which are not reflected in the sequential language [Plo77], and for many years the problem remained unsolved (although it sparked off many other interesting theoretical concepts).

The field of *game semantics* arose to model linear logics, and also to attack the PCF problem. Roughly simultaneously, syntax-free fully-abstract models of PCF were described using game semantics by Abramsky, Jagadeesan and Malacaria [AJM94], Hyland and Ong [HO00], and Nickau [Nic96]. A game is a formal setting for interaction, and such work presents a way to describe interaction by a mathematical structure suitable for denotational semantics. The idea of using "dialogue" to model logic is surveyed in [Fel86].

The variety of game used by Abramsky, Jagadeesan and Malacaria grew from the linear logic games of Blass [Bla92], and the Hyland/Ong and Nickau approach — which are essentially equivalent — descend from the earlier work of Lorenzen [LL78]. Subsequently variations of both of these games were used to give denotational models to languages including recursive types [McC98], Algol-like languages [AM95b], call-by-value reduction [HY97] and [AM99], System F polymorphism [Hug00], finite non-determinism [Har00], and languages with non-local control [Lai98].

In [AM95a], Abramsky and McCusker gave a model of a version of the lazy λ-calculus (introduced in [Abr90], for details of models see [Ong88] and [AO93]). This was the first game model of untyped λ-calculus; apart from the fact that the language is not the pure untyped λ-calculus, it also differs substantially from the work presented here in that it uses the "history-free" approach of Abramsky *et al.* A full-abstraction result is proved, but the model is not analysed in the same fine detail as in this work — in particular there is no exact correspondence between a term and its denotation. (McCusker subsequently informed the author that he had always known of the first of the pure untyped λ-calculus models presented in this thesis, but had not realised that they were at all interesting).

This thesis uses the style of game introduced by Hyland, Ong and Nickau, but with an important modification. Rather than considering each move as a "question" or "answer", this distinction is removed; we might call all the moves "declarations". Game models of languages with state, non-determinism, non-local control, and so

on, were constructed as modifications of the PCF game models with some relaxation on the conditions on strategies or legal positions which make up the model. Indeed, Abramsky has outlined an "intensional hierarchy" in which the connexion between the relaxation of some such conditions and language features is made precise, and shown to be independent (some of this work appears in[AM97]). Our removal of the distinction between questions and answers is equivalent to removing the so-called *well-bracketing* convention of [HO00], and this relaxation is also studied by Laird in [Lai98] for functional languages with non-local control. However we do not consider that the models of the pure untyped λ-calculus are fundamentally about non-local control; instead it is just that the well-bracketing convention becomes redundant for a pure untyped language, where there are no ground types and hence no answers to questions.

One feature of game semantics, and which seems to be especially visible in the Hyland/Ong/Nickau style, is that a strategy bears a close resemblance to the term it denotes. A recurring theme is that the strategy which denotes a term in some sense "is" the algorithm which performs the computation carried out by the term. In this work we find a cartesian closed category with a reflexive object — all that is needed for a model of the pure untyped λ-calculus — in a standard way, but the main interest is from this connexion, which we are able to make precise, and which allows a much finer analysis. Another recurring theme of innocent game models is that of "copycat" strategies. In this work we show that strategies which are (mostly) copycat are precisely those which denote terms.

We claim that the game models of untyped λ-calculus we present here are worthwhile for three reasons:

(i) They induce reasonable equational theories on the language.

(ii) The denotation of a term is a strategy. This strategy "is" the algorithm which is naïvely described by the term. This connexion is made precise.

(iii) Because of the above correspondence, we can refine the models to obtain a *universal* model — every element of the model is the denotation of some term.

The correspondence result and universality property appear to be the first of their kind for models of the λ-calculus.

Additionally, the universality property presents the possibility that some classical results of the λ-calculus may be proved or strengthened using the model. We present a semantic proof of Böhm's Theorem, which is quite different from the usual syntactic proof, and is slightly more general.

Recently, Di Gianantonio *et al.* described game models for the pure untyped λ-calculus [DFH99], with some subsequent developments [DF98] and [DF00]. Their work differs from that presented in this thesis in that it is founded on AJM games rather than innocence, and the only analysis of the models is based on the traditional techniques of approximation from [Hyl76]. The analysis does not contain

any analogue of our correspondence theorems and no extensionality or universality result. However they do characterise precisely the λ-theories which can be induced by game models of the untyped λ-calculus of their type. It would be interesting to know whether the same limitations apply to our setting. This work is discussed more in the conclusion of this thesis.

1.2 Prerequisites

The reader will need to be familiar with the untyped λ-calculus; a comprehensive reference is [Bar84]. Particularly important topics are solvability, Böhm trees, standard theories and models. Basic category theory (up to cartesian closed categories) is required, the classic reference is [Mac71], alternatively the first two chapters of [Cro93] more than suffice. We hope that the reader is aware of the connexion between models of the untyped λ-calculus and CCCs, but we outline the main result in the introductory material anyway. Some references are made to computability, see for example [Cut80].

Familiarity with the games literature, particularly the innocent approach of [HO00], [Nic96], and [McC98], will motivate many of the basic definitions, but is not required.

1.3 Overview

The remainder of Chapter 1 is an introduction the the untyped λ-calculus (for notational purposes only; we assume that the reader is already familiar with the definitions presented) and the way models of it arise in cartesian closed categories. We also introduce some notation; in particular our use of the term "tree" must be understood properly.

In Chapter 2 we make the basic definitions for the games, that of arena (which details the moves of the game), and legal position (which sets out some rules for the game). The game is played between two imaginary people who must alternate moves, and each move is justified by some preceding move. A strategy for the two participants in the game is defined initially as a set of possible traces of moves which the player will engage in, and we say what it means for a strategy to be innocent. Furthermore, strategies may be composed, and innocence is preserved by composition. These ideas are analogues of the standard definitions from the games literature, with the removal of the question and answer distinction, and the constructions of functions and product arenas are standard.

Another familiar idea is that of an innocent function (which is the presentation used in [Nic96]). 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. This will become the more important presentation later in the work.

Taking arenas as objects and strategies as morphisms gives rise to the categories $\mathbb{A}$ and $\mathbb{A}_{\mathrm{REC}}$. They are cartesian closed (Theorem 2.4.1), and both contain the same reflexive object U (in fact U is the same object as $U \Rightarrow U$, although an isomorphism suffices for the following properties). In a standard way this leads to $\lambda\eta$-algebras $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ (Theorem 2.4.2). The elements of $\mathcal{D}$ are the innocent strategies on U (and the elements of $\mathcal{D}_{\mathrm{REC}}$ are just the effective innocent strategies on U), and composition of the elements is defined in terms of composition of strategies.

By formulating 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 what Barendregt calls the "basic equations" for approximants hold (Lemma 2.5.4). This leads in a standard way to the fact that the models are sensible, i.e. all unsolvable terms have the same denotation (Theorem 2.5.5).

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 (Theorem 2.5.7).

In order to improve the model we seek in Chapter 3 a way to characterise the definable elements. Strategies are first written in an 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 with an infinite η-expansion. By encoding Nakajima trees in a variable-free form we prove the powerful Exact Correspondence Theorem 3.3.1, which says that the variable-free form of the Nakajima tree of a term coincides precisely with the economical form of the innocent strategy which denotes it. This immediately tells us the equational theory induced by the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ is the maximal consistent sensible theory $\mathcal{H}^*$ (Corollary 3.3.3). The same is true for Scott's D_∞ models.

We proceed to make use of the Exact Correspondence Theorem, 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, the 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 $\mathbb{A}_{\mathrm{EAC}}$ (compositionality of EAC strategies is a highly technical proof which is actually deferred until Chapter 4, where it properly belongs). This category still has the reflexive object U so we can identify a new λ-algebra $\mathcal{D}_{\mathrm{EAC}}$ (Theorem 3.5.1), which is both sensible and universal — every term is the denotation of some term (The-

orem 3.5.3). To our knowledge, other than term models, the universality result is the first of its kind.

In contrast to $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, $\mathcal{D}_{\mathrm{EAC}}$ is order-extensional. This can be shown in two ways. We first present the technical but powerful result called the Separation Lemma 3.6.1, which leads to order-extensionality via a semantic proof (Theorem 3.7.6). Weak extensionality follows easily, so $\mathcal{D}_{\mathrm{EAC}}$ is a $\lambda\eta$-model. Alternatively, it is possible to give a syntactic proof of this result, using the techniques developed by Barendregt to give the proof of Böhm's Theorem found in [Bar84, §10.3] and we include it at the end of Section 3.7.

The former method is of interest because it can be extended to give a semantic proof of Böhm's Theorem, either as a separation result for the λ-theory $\mathcal{H}^*$ (Lemma 3.8.3) or as a genuine generalisation of Böhm's classical result of [Böh68] for λ (Theorem 3.8.4).

In Chapter 4 we pursue a remark which crops up in discussion of EAC strategies, that copycat thresholds are not unique. Our aim is to modify the model $\mathcal{D}_{\mathrm{EAC}}$ to find a game model which does not validate η-conversion. By drawing a connexion between the parts of Nakajima trees which are generated by η-expansion, using the Exact Correspondence Theorem, and copycat thresholds, it becomes clear that additional information which specifies copycat thresholds at each P-view is the information we need to add to EAC strategies to invalidate η-conversion.

We are thus lead to the definition of an EXAC strategy. Algorithm 4.2.1 gives a method to compose EXAC strategies, and the highly technical Theorem 4.2.2 proves the correctness of this algorithm.

However it turns out to be difficult to describe a CCC of EXAC strategies. We present the "obvious" attempt, which does not work (Theorem 4.3.2), and a solution $\mathbb{XA}$ (Theorems 4.4.2 and 4.4.3). The CCC $\mathbb{XA}$ still has the reflexive object U, but this time the retractions are not iso, so we arrive at a model $\mathcal{D}_{\mathrm{XA}}$ which does not validate η-conversion. Because the category of EAC strategies $\mathbb{A}_{\mathrm{EAC}}$ embeds into $\mathbb{XA}$ in a way which preserves the cartesian closed structure (Theorem 4.5.2), $\mathcal{D}_{\mathrm{XA}}$ is also a sensible model.

We formulate an analogue of the Exact Correspondence Theorem, this time between Böhm trees in a variable-free form and denotation in $\mathcal{D}_{\mathrm{XA}}$ (Theorem 4.6.5). This means that the local structure of $\mathcal{D}_{\mathrm{XA}}$ is the λ-theory $\mathcal{B}$ (which equates two terms if and only if they have the same Böhm tree). Similarly to $\mathcal{D}_{\mathrm{EAC}}$, $\mathcal{D}_{\mathrm{XA}}$ is universal (Theorem 4.6.8). Unlike $\mathcal{D}_{\mathrm{EAC}}$, however, $\mathcal{D}_{\mathrm{XA}}$ is not extensional or even weakly extensional (Corollary 4.6.10).

We conclude the thesis with a closer look at the parallel work of Di Gianantonio *et al.*, and some directions for further research.

The results of Chapter 2 and Sections 3.1 to 3.5 are reported in [KNO01]. An extended abstract of Chapter 4 appears in [KNO99], and as a full-length paper [KNO00].

1.4 Notation

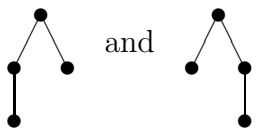
We first describe some conventions for sets and sequences:

- 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, especially when dealing with sequences of sequences (when we will usually use the vector notation for the highest-level sequence and not for the others).
- 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, written $\leq$. The subsequence pre-order is written $\preceq$.

In the graph theoretic sense, a *tree* is just a directed, connected, graph whose underlying undirected graph is acyclic. Usually, when we talk about a tree we mean a countably-branching *labelled* tree, which can be presented in a more concrete structure. Often, the labels will be sequences in $\mathbb{N}^*$; the root is labelled ε and the descendants of the node labelled $\vec{s}$ are labelled $\vec{s} \cdot 1, \ldots, \vec{s} \cdot n$, if there are n such descendants. Thus we can talk about the "m^{th} descendant of the node $\vec{s}$" — it is the node labelled $\vec{s} \cdot m$. We can describe a tree by the set of labels of its nodes.

When we draw trees they are 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* or *ancestor* of a node, or say that one node *inherits from* another, with the obvious meanings). Because of the labelling by sequences of natural numbers, the children come with an order, also like a family tree. When we draw a tree without labels, we intend that the numbering of children goes from left to right.

This means that we do not consider the trees drawn



to be the same. The first is described by the set $\{\varepsilon, \langle 1 \rangle, \langle 1, 1 \rangle, \langle 2 \rangle\}$ and the second by $\{\varepsilon, \langle 1 \rangle, \langle 2 \rangle, \langle 2, 1 \rangle\}$. In the graph-theoretic sense, we would expect these to be different pictures of the same tree, but for our purposes they are different trees.

More generally, a Σ-*labelled tree* is a (possibly infinitary) tree with nodes labelled with elements from Σ, and a *partially* Σ-*labelled tree* is a tree with nodes labelled with elements from $\Sigma \cup \{\bot\}$, such that any node labelled $\bot$ has no descendants. A node labelled $\bot$ is considered to be part of the tree, but is without a label.

Regardless of how a tree is labelled, we can encode each node by an element of $\mathbb{N}^*$ by enumerating the branches at each node (as long as the tree is only countably branching, and there is a well order on the labels of the descendants of each node). We then refer to the *depth* of an element of a tree, which is the length of the sequence which encodes it. Thus the root of a tree is at depth zero.

Using the same encoding of nodes, we can define the *labelling function* of a Σ-labelled tree, as a partial map from $\mathbb{N}^*$ to Σ, mapping the encoding of a node to its label, or undefined if the node in question does not exist in the tree. The labelling function of a partially Σ-labelled tree is undefined on sequences encoding nodes labelled $\bot$. The domain of definition of such labelling functions is always prefix-closed.

1.5 The Untyped λ-Calculus

We expect that the reader is familiar with the untyped λ-calculus, but for precision we give a brief survey of some important definitions. For motivation and examples, the reader is referred to the comprehensive survey [Bar84].

The language we study in this thesis consists of a set of *terms* and various equalities between these terms. The theory of the untyped λ-calculus includes other aspects, notably notions of *reduction* on terms, but we will not make use of these ideas in this work.

We assume that a countable set of *variables* $\mathcal{V} = \{v_0, v_1, \ldots\}$ has been identified.

Definition The *terms* of the untyped λ-calculus are finite strings of the symbols $\lambda, (,), ., v_0, v_1, \ldots$. The set of terms, Λ, is defined by the following BNF grammar, where s is a meta-variable for terms and x a meta-variable for variables:

$$s ::= s \mid (\lambda x.s) \mid (ss).$$

A term is named a *variable*, an *abstraction*, or an *application*, respectively, depending on which clause it matches.

We usually use letters such as x, y, z to range over variables, and s, t to range over terms. As usual, we omit parentheses wherever possible using the convention that a sequence of applications associates to the left, and that the scope of an abstraction is as large as possible. We also write $\lambda x_1 x_2 \ldots x_n.s$ as shorthand for $(\lambda x_1.(\lambda x_n.(\ldots \lambda x_n.s)))$.

The λ is a *binding operator*, and we define the *free variables* of a term in the usual way. The set of free variables of the term s is written $\mathrm{FV}(s)$. Note that the variable x in the term $x\lambda x.x$ occurs both free and bound. A term with no free variables is called *closed*, and the set of such terms is denoted Λ^0. We sometimes allow the conventional use of uppercase letters to denote closed terms.

Definition The operation of *substitution* is defined as usual: given a term s, the term $s[x{:=}t]$ is the result of substituting t for all free occurrences of x in s. Some renaming of bound variables may occur to avoid capturing the free variables of t in abstractions of s. Formally,

$$\begin{array}{rcl} x[x{:=}t] & = & t, \\ y[x{:=}t] & = & y \text{ if } x \neq y, \\ (su)[x{:=}t] & = & (s[x{:=}t])(u[x{:=}t]), \\ (\lambda x.s)[x{:=}t] & = & \lambda x.s, \\ (\lambda y.s)[x{:=}t] & = & \lambda y.(s[x{:=}t]) \text{ if } y \notin \mathrm{FV}(t), \\ (\lambda y.s)[x{:=}t] & = & (\lambda z.s[y{:=}z])[x{:=}t] \text{ if } y \in \mathrm{FV}(t), \\ & & \text{where } z \text{ is some variable not occurring in } s \text{ or } t. \end{array}$$

In order to reason about substitution on a purely syntactic level, without regard to variable capture, we also need the following concept.

Definition A *context* is a term with a "hole", into which any other term can be plugged. Formally, the set of contexts $\mathcal{C}[X]$ is the subset of the finite strings of the symbols $X, \lambda, (,), ., v_0, v_1, \ldots$ ranged over by meta-variables like C and given by the grammar

$$C = X \mid x \mid (\lambda x.C) \mid (CC).$$

If C is a context we write $C[t]$ for the term generated by replacing all occurrences of X in C by t. This replacement is regardless of whether this captures free variables of t in abstractions of C.

Contexts can also have more than one hole; the set of contexts with n holes is denoted $\mathcal{C}[X_1, \ldots, X_n]$, with the obvious meaning, and the n-fold syntactic substitution of terms for the holes of C by $C[s_1, \ldots, s_n]$.

We define a relation on terms which captures the idea that a variable bound by a λ is a "dummy" variable and could be renamed to any other.

Definition The relation between terms of *α-conversion*, or *α-equivalence*, is written $\equiv_\alpha$ and axiomatised by the following system:

$$(\alpha) \quad \lambda x.s \equiv_\alpha \lambda y.(s[x := y]), \text{ as long as } y \text{ does not occur in } s,$$

$$\text{(ER)} \quad \begin{cases} s \equiv_\alpha s, \\ s \equiv_\alpha t \implies t \equiv_\alpha s, \\ s \equiv_\alpha t \ \wedge \ t \equiv_\alpha u \implies s \equiv_\alpha u, \end{cases}$$

$$\text{(CC)} \quad \begin{cases} s \equiv_\alpha t \implies su \equiv_\alpha tu, \\ s \equiv_\alpha t \implies us \equiv_\alpha ut, \\ s \equiv_\alpha t \implies \lambda x.s \equiv_\alpha \lambda x.t. \end{cases}$$

The clauses labelled (ER) ensure that $\equiv_\alpha$ is an equivalence relation, and those labelled (CC) mean that $\equiv_\alpha$ is *compatible* with the constructions of application and abstraction i.e. for any context C, $s \equiv_\alpha t$ implies $C[s] \equiv_\alpha C[t]$.

We consider two terms which are α-equivalent to be syntactically the same, for example $\lambda x.x$ is the same term as $\lambda y.y$. Precisely, the set of terms is the set Λ given above quotiented by the relation $\equiv_\alpha$, although by an abuse of notation we still refer to it as Λ. We write $\equiv$, the usual symbol for *syntactic equality*, for $\equiv_\alpha$ to emphasise this.

We will write a term as some member of the equivalence class it belongs to, but allow ourselves to change representatives, i.e. rename the bound variables, at will. (The ability to rename bound variables explains why we never actually refer to the basic variables $\mathcal{V}$ — in any closed term they can be substituted for any other (unused) variables so they might as well be represented by meta-variables.)

In this work we adhere to the *variable convention*, which states that whenever we use a set of terms in one place we rename all the bound variables of the terms to be different from any free variables of the terms. In the presence of this convention, the complicated last clause in the definition of substitution becomes redundant.

We study equational theories between the terms of the λ-calculus, the prototypical theories being given by the following.

Definition The theory λ is an equational theory between terms, i.e. a set of

sentences of the form $s = t$ for $s, t \in \Lambda$. It is axiomatised by the following system:

$$(\beta) \quad (\lambda x.s)t = s[x := t],$$

$$(\mathrm{ER}) \quad \begin{cases} s \equiv_\alpha s, \\ s \equiv_\alpha t \implies t \equiv_\alpha s, \\ s \equiv_\alpha t \ \wedge \ t \equiv_\alpha u \implies s \equiv_\alpha u, \end{cases}$$

$$(\mathrm{CC}) \quad \begin{cases} s \equiv_\alpha t \implies su \equiv_\alpha tu, \\ s \equiv_\alpha t \implies us \equiv_\alpha ut, \\ s \equiv_\alpha t \implies \lambda x.s \equiv_\alpha \lambda x.t. \end{cases}$$

The theory $\lambda\eta$ is axiomatised by the above plus the following additional axiom:

$$(\eta) \quad \lambda x.sx = s, \text{ provided } x \notin \mathrm{FV}(s).$$

If $s = t$ is provable in the theory λ we write $\lambda \vdash s = t$, similarly for $\lambda\eta$.

The theory λ captures the intuition that the λ operator in $(\lambda x.s)t$ binds the variable x to the term t to which the abstraction is applied, and ensures that this notion of equality is an equivalence relation and is closed under the constructions of application and abstraction. Use of the rule β is called *β-conversion*. Use of the rule η is generally called *η-conversion*, also replacing a term s with $\lambda x.sx$ is called *η-expansion*, and vice versa is called *η-reduction*.

It is of interest to study extensions of the theory λ: equational theories which include all the equations of λ plus possibly some others. We will only be interested in theories which are closed under the axioms of λ (and which are *consistent*, i.e. do not equate all terms).

It is traditional to concentrate on *closed equations*, that is sentences of the form $s = t$ for closed terms s and t. We identify an equational theory with the set of closed equations provable in it, so that for example $\lambda = \{s = t \mid \lambda \vdash s = t, \ s, t \in \Lambda^0\}$. Given a set of sentences $\mathcal{T}$ we write $\lambda + \mathcal{T}$ to mean the theory axiomatised by the rules of λ, with each sentence of $\mathcal{T}$ included as an additional axiom.

Definition A *λ-theory* $\mathcal{T}$ is a consistent equational theory between closed terms satisfying $\lambda + \mathcal{T} \vdash s = t \iff \mathcal{T} \vdash s = t$ (for closed terms s and t).

Remark 1.5.1 It is simple to check that for any λ-theory $\mathcal{T}$, $\lambda + \mathcal{T} \vdash s = t \iff \lambda + \mathcal{T} \vdash \lambda x.s = \lambda x.t$. Hence it is not necessary to restrict our attention to closed equations. In view of this we write $\mathcal{T} \vdash s = t$ for $\lambda + \mathcal{T} \vdash s = t$, for arbitrary terms s and t. (This is a subtle point: for terms s and t which are not closed, it cannot be the case that $\mathcal{T} \vdash s = t$ because $\mathcal{T}$ consists only of closed equations.)

If a λ-theory $\mathcal{T}$ is closed under the rule η of the system $\lambda\eta$ (equivalently, if $\lambda\eta + \mathcal{T} \vdash s = t \iff \mathcal{T} \vdash s = t$) then we say that $\mathcal{T}$ is a *$\lambda\eta$-theory.*

The following result characterises the $\lambda\eta$-theories; a proof can be found in [Bar84, §4.1].

Lemma 1.5.2 Define two standard terms $I = \lambda x.x$ and $1 = \lambda xy.xy$. Then a λ-theory $\mathcal{T}$ is a $\lambda\eta$-theory if and only if $\mathcal{T} \vdash I = 1$.

We now turn our attention away from general theories and concentrate on the standard theory λ. What follows is an introduction to the notions of Böhm tree and Nakajima tree, two ways of presenting a term which will be of importance to the game models in the body of this work.

Definition A term s is in *head normal form* if $s \equiv \lambda x_1 \dots x_n.ys_1 \dots s_m$ for some variable y, terms $s_1, \dots, s_m$, and $n, m \in \mathbb{N}_0$ (so that there are possibly no abstractions or terms s_i). We say that y is the *head variable* of s.

A term s *has a head normal form* is $\lambda \vdash s = t$ for some term t in head normal form, in which case the head variable of s is the head variable of t.

We often abbreviate the phrase "head normal form" to HNF.

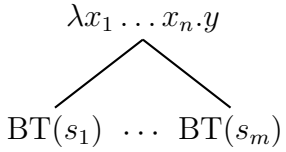
The property of having a HNF is equivalent to another notion. A closed term s is *solvable* if there are terms $t_1, \dots t_n$ such that $st_1 \dots t_m = I$ (I is defined above). A term s with free variables in the set $\{x_1, \dots, x_n\}$ is solvable if $\lambda x_1 \dots x_n.s$ is solvable, and this turns out to be independent of the set and the order of the abstractions. It was proved in [Wad71] that a term is solvable if and only if it has a HNF, and so we shall use the terms "solvable" and "has a HNF" (and "unsolvable" and "has no HNF") interchangeably. An example of an unsolvable term is $\Omega = (\lambda x.xx)(\lambda x.xx)$.

If a term s has a HNF $\lambda x_1 \dots x_n.ys_1 \dots s_m$ we can examine the HNF of the terms $s_1, \dots, s_m$, if they exist. Those that do will themselves have sequences of terms which we can examine, and we can continue this process repeatedly. In this way we construct the *Böhm tree* of the term s, first introduced by Barendregt in [Bar77]. What follows is not a formal definition, and for a more thorough treatment the reader is referred to [Bar84, Chapter 10].

Informal Definition Let Σ be $\{\lambda x_1 \dots x_n.y \mid n \in \mathbb{N}_0, x_1, \dots, x_n, y \text{ variables}\}$.

For a term s the *Böhm tree* of s, written $\mathrm{BT}(s)$, is the partially Σ-labelled tree defined inductively as follows:

$\mathrm{BT}(s) = \bot$ if s is unsolvable

$\mathrm{BT}(s) =$  if s has HNF $\lambda x_1 \ldots x_n.y s_1 \ldots s_m$

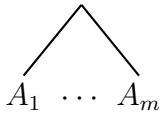
In the first clause it is important to note that the tree consists of a single unlabelled node, as opposed to being an "undefined" tree. The labelling function of the tree, however, is everywhere undefined.

Remark 1.5.3 This definition does not actually define a computable procedure for constructing the Böhm tree of a term, since the predicate of unsolvability is undecidable. An alternative definition (called the *effective Böhm tree*) is presented in [Bar84, 10.1.9], but since the effect of the definition is equivalent we prefer to stick to the more comprehensible version given above.

We say that a a *Böhm-like tree* is a finitely-branching Σ-labelled tree. Not all Böhm-like trees are the Böhm trees of a term, however. It is of importance to this work to characterise those that are, and to do so we need the following.

Definition The set of *free variables* of a Böhm-like tree A, written $\mathrm{FV}(A)$, is defined inductively as follows.

$\mathrm{FV}(\bot) = \emptyset$

$\mathrm{FV}(A) = (\{y\} \cup \bigcup_{i=1}^{m} \mathrm{FV}(A_i)) \setminus \{x_1, \ldots, x_n\}$ if A is of the form 

Somewhat counterintuitively, although one has $\mathrm{FV}(\mathrm{BT}(s)) \subseteq \mathrm{FV}(s)$ the inclusion may be proper (for an example and some explanation see [Bar84, 10.1.22]).

Definition A Böhm-like tree A is *recursively enumerable*, or just r.e., if after some coding of the set Σ the labelling function is partial recursive.

We may now characterise the Böhm-like trees which are Böhm trees of some term. A proof may be found in [Bar84, §10.1]

Theorem 1.5.4 Let A be a Böhm-like tree. Then

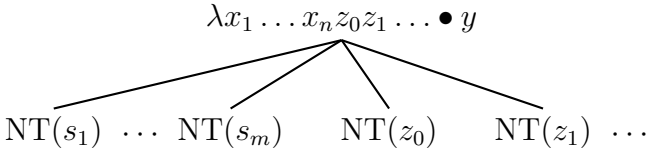
$$\exists s \in \Lambda.\ \mathrm{BT}(s) = A \iff A \text{ is r.e. and } \mathrm{FV}(A) \text{ is finite.}$$

We introduce another presentation of terms as trees, first proposed by Nakajima in [Nak75]. 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, here given only informally.

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

$\mathrm{NT}(s) = \bot$ if s is unsolvable

$\mathrm{NT}(s) =$ if s has HNF $\lambda x_1 \ldots x_n . y s_1 \ldots s_m$

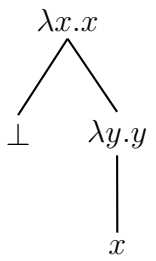


where $z_0, z_1, \ldots$ are countably many fresh variables.

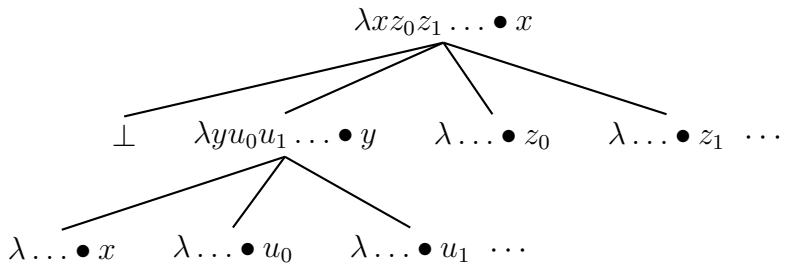
A formalization of the process of finding fresh variables at each stage is given in [Nak75].

Example 1.5.5 We illustrate the definitions of Böhm and Nakajima tree. Take the term $s = \lambda x.x\Omega(\lambda y.yx)$ (where Ω is the unsolvable term described above). The Böhm tree and part of the Nakajima tree of s are given by

$\mathrm{BT}(s) =$

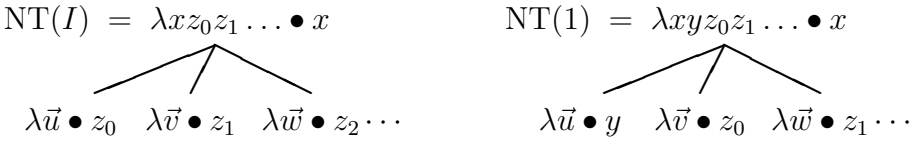


$\mathrm{NT}(s) =$



Example 1.5.6 This example illustrates that the process of infinite η-expansion which motivates the definition of Nakajima tree causes terms which differ only by η-conversion to be identified.

Recall the terms I and 1. The reader may wish to verify that the following represents the first two levels of the Nakajima trees of those terms:



After renaming of bound variables, these are the same.

Note that different terms may have the same Böhm tree or Nakajima tree, even when the terms are not equated in the theory λ or $\lambda\eta$. (In particular all unsolvable terms have the same Böhm tree and Nakajima tree $\bot$, but not all unsolvable terms are equated by λ or $\lambda\eta$.) Thus we make the following definitions:

Definition

(i) $\mathcal{B} = \{s = t \mid s, t \in \Lambda^0, \mathrm{BT}(s) = \mathrm{BT}(t)\}$,

(ii) $\mathcal{H}^* = \{s = t \mid$ for all contexts C. $C[s]$ solvable $\iff C[t]$ solvable$\}$.

Proofs of the following results, which are nontrivial, can be found in [Bar84]. and [Nak75].

Lemma 1.5.7

(i) $\mathcal{B}$ and $\mathcal{H}^*$ are λ-theories,

(ii) $\mathcal{H}^* \vdash s = t \iff \mathrm{NT}(s) = \mathrm{NT}(t)$.

Any unsolvable term has the same Böhm and Nakajima tree, namely $\bot$. This means that $\mathcal{B}$ and $\mathcal{H}^*$ are examples of *sensible* λ-theories.

Definition A λ-theory $\mathcal{T}$ is *sensible* if $\mathcal{T} \vdash s = t$ for all unsolvable terms s and t.

For a motivation of why such a property should be called *sensible*, the reader is referred to the latter part of [Bar84, §2.2].

The λ-theory $\mathcal{H} = \{s = t \mid s, t \in \Lambda^0, s, t$ unsolvable$\}$ is the smallest sensible theory, but it is not equal to either $\mathcal{B}$ or $\mathcal{H}^*$. The following gives the relationships between the theories (where the ordering is inclusion, i.e. $\mathcal{T} \subseteq \mathcal{T}'$ if $\mathcal{T} \vdash s = t$ implies $\mathcal{T}' \vdash s = t$):

Lemma 1.5.8 The following are proper inclusions between consistent λ-theories:

$$\lambda \subset \mathcal{H} \subset \mathcal{B} \subset \mathcal{H}^*.$$

Furthermore, $\mathcal{H}^*$ is *maximal* in the following sense: given any λ-theory $\mathcal{T} \supseteq \mathcal{H}^*$, either $\mathcal{T} = \mathcal{H}^*$ or $\mathcal{T}$ is inconsistent.

Since $\mathcal{H}^* \vdash I = 1$ (as we showed in Example 1.5.6), $\mathcal{H}^*$ is in fact a $\lambda\eta$-theory. The same does not hold for $\mathcal{B}$.

In fact, this is not the only difference between $\mathcal{H}^*$ and $\mathcal{B}$. We can augment the λ-theory $\mathcal{B}$ with the rule η by forming the $\lambda\eta$-theory $\mathcal{B}\eta = \{s = t \mid s, t \in \Lambda^0,\ \lambda\eta + \mathcal{B} \vdash s = t\}$, but it is shown in [Bar84, 16.4.4] that $\mathcal{B}\eta$ is a proper subset of $\mathcal{H}^*$. This is because $\mathcal{H}^*$, being equality of Nakajima trees, supports infinite η-conversion, whereas $\mathcal{B}\eta$ only allows finitely many η-conversions in a proof of $\mathcal{B}\eta \vdash s = t$.

1.6 Model Theory of the Untyped λ-Calculus

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. In this section we give a very brief presentation of what we consider to be an abstract model of the λ-calculus, and show how these arise in a categorical setting. It is in a categorical framework that the game models will be presented in the body of this work.

The description we give here is very cursory, and purposefully avoids mention of combinatory algebras except in passing. For a thorough treatment the reader is referred to [Bar84, Chapter 5], or [Koy82].

Definition An *applicative structure* is a pair $\langle \mathcal{A}, \bullet \rangle$ consisting of a set $\mathcal{A}$ and a binary operation $\bullet$ on $\mathcal{A}$ called *application*.

In an applicative structure the operation of application will associate to the left, as in the λ-calculus.

Since the untyped λ-calculus is based on application, a model will be an applicative structure $\langle \mathcal{A}, \bullet \rangle$, together with a map $[\![-]\!]$ describing how each term may be interpreted in $\mathcal{A}$. We will provide an interpretation of an extension of the λ-calculus which includes the *constants* $\mathcal{C}_\mathcal{A} = \{c_a \mid a \in \mathcal{A}\}$ (where each constant represents a fixed member of $\mathcal{A}$). The set of such terms is written $\Lambda(\mathcal{A})$, ranged over by the meta-variable s, and is given by the grammar

$$s = x \mid c_a \mid (\lambda x.s) \mid (ss).$$

Closed terms of the λ-calculus will be interpreted as elements of $\mathcal{A}$, and each constant as the element of $\mathcal{A}$ it represents, but we can only interpret terms with free variables relative to a *valuation*, namely a map ρ from the set of all variables to $\mathcal{A}$. Relative to a valuation ρ, a term s will be interpreted as an element of $\mathcal{A}$ by a function $[\![-]\!]_\rho : \Lambda(\mathcal{A}) \to \mathcal{A}$, with the intention that this is the interpretation of s under the assumption that the free variables of s are instantiated to the elements of $\mathcal{A}$ as specified by ρ. Given a valuation ρ we write $\rho(x := a)$ for the valuation which is the same except it replaces whatever the value of x was by a. That is, $\rho(x{:=}a)(x) = a$, and $\rho(x{:=}a)(y) = \rho(y)$ for $y \neq x$.

In order to be a faithful model, we place certain conditions on the applicative structure and interpretation function:

Definition A λ-*algebra* is a tuple $\langle \mathcal{A}, \bullet, [\![-]\!]_- \rangle$, where $\langle \mathcal{A}, \bullet \rangle$ is an applicative structure, and for each valuation ρ, $[\![-]\!]_\rho : \Lambda(\mathcal{A}) \to \mathcal{A}$, such that the following hold:

(i) $[\![x]\!]_\rho = \rho(x)$,
(ii) $(\forall x \in \mathrm{FV}(s).\ \rho(x) = \rho'(x)) \implies [\![s]\!]_\rho = [\![s]\!]_{\rho'}$,
(iii) $[\![c_a]\!] = a$,
(iv) $[\![st]\!]_\rho = [\![s]\!]_\rho \bullet [\![t]\!]_\rho$,
(v) $[\![\lambda x.s]\!]_\rho \bullet a = [\![s]\!]_{\rho(x:=a)}$,
(vi) $\lambda \vdash s = t \implies [\![s]\!]_\rho = [\![t]\!]_\rho$.

Conditions (i) and (ii) specify that the valuations instantiate only the free variables of the term, and do so directly. In view of (ii), we can just write $[\![s]\!]$ for $[\![s]\!]_\rho$ when s has no free variables. Condition (iii) forces constants to be interpreted properly. Condition (iv) means that application in the λ-calculus corresponds to application in the applicative structure $\langle \mathcal{A}, \bullet \rangle$. Condition (v) means that, after interpretation, an abstraction has the intuitive meaning of binding the term it is applied to. Condition (vi) ensures the interpretation reflects all the equalities of the λ-calculus (many of which are automatic from the other conditions).

Remark 1.6.1 Any λ-algebra will be a *combinatory algebra*, which is an applicative structure equipped with distinguished elements k and s satisfying $k \bullet x \bullet y = x$ and $s \bullet x \bullet y \bullet z = x \bullet z \bullet (y \bullet z)$ for all elements x, y, z of the structure. In the case of a λ-algebra $\langle \mathcal{A}, \bullet, [\![-]\!]_- \rangle$, the elements k and s can be specified by $k = [\![\lambda xy.x]\!]$ and $s = [\![\lambda xyz.xz(yz)]\!]$.

Traditionally, one introduces the λ-algebras as combinatory algebras satisfying additional properties, but we have purposefully avoided this in order to shorten the exposition. In fact, the λ-algebras may be characterised as those combinatory algebras which satisfy a certain finite set of equations.

We consider any λ-algebra to be in principle a model of the untyped λ-calculus. However, there are other desirable properties of models which we might aim for:

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

(i) $\mathcal{M}$ is a λ-*model* if it is *weakly extensional*, that is for all $s, t \in \Lambda(\mathcal{A})$,

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

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

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

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

(iii) $\mathcal{M}$ is *universal* if every element of $\mathcal{A}$ is the denotation of some term (not involving constants), 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 we will see later that λ-models have a pleasing categorical structure.

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 and b, but $I \neq 1$. However, if η-conversion is included the calculus will have this property (for example $I = 1$ in the theory $\lambda\eta$).

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 within the model. A universal model could even be considered an alternative presentation of the λ-calculus, with respect to some λ-theory.

The proofs of the following results can be found in [Bar84, §5.2], along with a full exploration of the properties of extensionality and weak extensionality.

Lemma 1.6.2 Let $\mathcal{M} = \langle \mathcal{A}, \bullet, [\![-]\!]_- \rangle$ be a λ-algebra.

(i) If $\mathcal{M}$ is extensional then it is weakly extensional, i.e. a λ-model.

(ii) $\mathcal{M}$ is extensional if and only if it is weakly extensional and $[\![I]\!] = [\![1]\!]$.

In any λ-algebra we know that for closed terms s and t, $\lambda \vdash s = t \implies [\![s]\!] = [\![t]\!]$. Thus the model will equate at least all those terms which are provably equal in λ — it is of importance in studying the model to know about any other terms it might equate.

Definition Given a λ-algebra $\langle \mathcal{A}, \bullet, [\![-]\!]_- \rangle$ the *local structure*, or *equational theory*, is the set of sentences $\{s = t \mid s, t \in \Lambda^0,\ [\![s]\!] = [\![t]\!]\}$.

By the conditions imposed on a λ-algebra, this will be a λ-theory as long as the λ-algebra is *nontrivial* (i.e. does not interpret all terms as the same element). If the equational theory of a λ-algebra $\mathcal{M}$ satisfies the rule of η-conversion (is a $\lambda\eta$-theory), then we say that $\mathcal{M}$ is a *$\lambda\eta$-algebra* (and a weakly extensional $\lambda\eta$-algebra is called a *$\lambda\eta$-model*). Lemmas 1.5.2 and 1.6.2 mean that a λ-algebra is extensional if and only if it is $\lambda\eta$-model. We say that a λ-algebra is sensible if its local structure is a sensible λ-theory.

If the elements of the λ-algebra come with a partial ordering $\preceq$, it may also be of interest to study the partial order this induces on terms. We call this the *local order structure*, and it is the set of sentences $\{s \subseteq t \mid s, t \in \Lambda^0,\ [\![s]\!] \preceq [\![t]\!]\}$.

We now show how λ-algebras arise in cartesian closed categories. We first outline the standard notation we use for categories.

In a category we will write $f\,;g$ for composition of the morphisms $f : A \to B$ and $g : B \to C$ and id_A for the identity morphism on the object A. Where it exists write $\mathbf{1}$ for the terminal object and $!_A$ for the unique morphism $A \to \mathbf{1}$. In a category with binary products we denote the first projection $A \times B \to A$ as $\pi_A^{A\times B}$, the second projection as $\pi_B^{A\times B}$, and the pairing of $f : A \to B$ and $g : A \to C$ as $\langle f, g\rangle : A \to (B \times C)$. 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 $\Pi_{x_i}^{\Delta}$ for the obvious projection onto the i^{th} component. In a CCC we write $\mathrm{eval}_{A,B}$ for the evaluation map $(A \Rightarrow B) \times A \to B$, and for a morphism $f : C \times A \to B$ the curried morphism is denoted $\Lambda(f) : C \to (A \Rightarrow B)$.

Definition Let $\mathbb{C}$ be a cartesian closed category. An object R of $\mathbb{C}$ 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 $\langle \mathcal{A}, \bullet, [\![-]\!]_- \rangle$ as follows:

(i) $\mathcal{A}$ is the homset $\mathrm{Hom}_{\mathbb{C}}(\mathbf{1}, R)$.

(ii) 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 $\mathcal{A}$.

(iii) 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:

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

In the last clause we may assume that x does not appear in Δ (by renaming if necessary).

(iv) 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 in [Bar84, §5.5].

Proposition 1.6.3 With the above construction $\langle \mathcal{A}, \bullet, [\![-]\!]_- \rangle$ is a λ-algebra. We denote this model $\mathcal{M}(\mathbb{C}, R, \mathit{Fun}, \mathit{Gr})$.

If a λ-algebra arises in such a way, we can express extensionality properties categorically.

Proposition 1.6.4

(i) $\mathcal{M}(\mathbb{C}, R, \mathit{Fun}, \mathit{Gr})$ is a λ-model if and only if R *has enough points*. That is, for all $f, g \in \mathrm{Hom}(R, R)$, $(\forall x : \mathbf{1} \to R.\ x \,;\, f = x \,;\, g)$ implies $f = g$.

(ii) $\mathit{Gr} \,;\, \mathit{Fun} = \mathrm{id}_R$ (in other words, the retraction morphisms form an isomorphism between R and $R \Rightarrow R$) if and only if $[\![1]\!] = [\![I]\!]$ in the model $\mathcal{M}(\mathbb{C}, R, \mathit{Fun}, \mathit{Gr})$.

As a consequence of these, $\mathcal{M}(\mathbb{C}, R, \mathit{Fun}, \mathit{Gr})$ is extensional if and only if $R \cong (R \Rightarrow R)$ via *Fun* and *Gr* and R has enough points.

In fact, every λ-algebra can be obtained from some CCC with a reflexive object, and every λ-model from some CCC with a reflexive object having enough points. This was first observed by Scott and is proved in [Koy82], and the construction is duplicated in [Bar84]. However, this is not to say that every λ-algebra can be obtained from a CCC which has a natural, syntax-free, presentation!

Chapter 2

Games, Innocence, and the Model $\mathcal{D}$

We build on the dialogue games of Hyland and Ong [HO00] and Hanno Nickau [Nic96] (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 also use very concrete representations of arenas and strategies, more so than as is traditional, to aid the fine analysis later in the work. 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 [HO00], [Nic96] and [McC98], 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 An *arena* is a finite tuple of nonempty trees of *moves*. The root of each tree is called an *initial move*.

We will only be interested in arenas made up of countably branching trees. 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 the uniform way described in the introduction — the root is labelled ε and the n^{th} child of the node labelled $\vec{s}$ is labelled $\vec{s} \cdot n$. We will only ever refer to moves by their label. 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.**

Remark 2.1.1 The standard definition of [HO00] 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 of 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).

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 2.1.2 There are three important arenas which will be referred to often in this work:

(i) E is the arena consisting of no trees at all; in sequence-subset form it is represented by $\langle\rangle$.

(ii) M is the "minimal" single-tree arena, consisting of one tree of one node; in sequence-subset form it is represented by $\langle\{\varepsilon\}\rangle$.

(iii) 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

(i) 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 $B_i \subseteq A_i$.

(ii) An arena is *finitely branching* if every tree in it is finitely branching.

(iii) An arena is *recursive* if, as a subset of $\mathbb{N}^*$, each tree in it is a recursive set (i.e. membership is decidable).

(iv) 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.

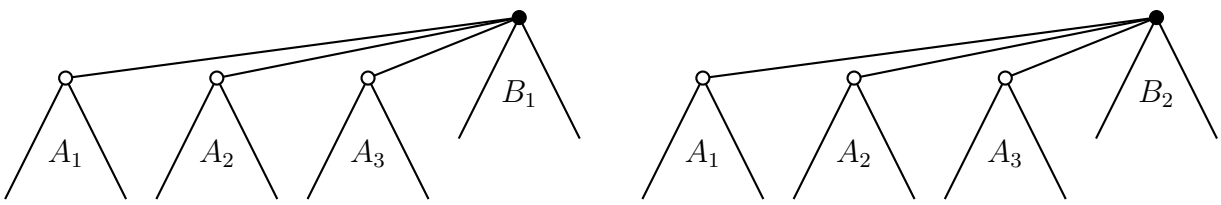
Definition Suppose that $A = \langle A_1, \ldots, A_m\rangle$ and $B = \langle B_1, \ldots, B_n\rangle$ are arenas.

(i) The *product arena* $A \times B$ is the "disjoint union" of the trees of A and B, the concatenation of their tuples. Formally $A \times B = \langle A_1, \ldots, A_m, B_1, \ldots, B_n\rangle$.

(ii) 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

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

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 moves in A 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 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.

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

Definition 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 permissable sequences of moves in the games played on the arena. One could think of these conditions as imposing some basic ground rules.

Remark 2.1.3 For the definition of justified sequences it suffices to say that the justification pointers exist. In fact we are only interested in well-formed sequences, and we could give these a proper mathematical meaning by encoding them as 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 (in a well-formed sequence) 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, and indeed we would need a different encoding to deal with general justified sequences which might not be well-formed, but this particular encoding matches that used in the definition of *economical form* in Section 3.1.

To avoid tedious detail, in practice we ignore issues of 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.

Remark 2.1.4 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 [Nic96] 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 [McC98]. 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 The *P-view* of a justified sequence $\vec{s}$, written $\ulcorner\vec{s}\urcorner$, is given recursively by:

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

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

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

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 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 2.1.5 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 [HO00, 4.2.5] or [McC98]. ■

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 and legal positions and their interaction with arenas and function space arenas in particular. 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 [HO00, §4.4] or [McC98].

Lemma 2.1.6 (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 2.1.7 (View Idempotency) If $\vec{s}$ is a legal position then $\lfloor\lfloor\vec{s}\rfloor\rfloor = \lfloor\vec{s}\rfloor$ and $\ulcorner\ulcorner\vec{s}\urcorner\urcorner = \ulcorner\vec{s}\urcorner$.

Definition 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 those 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 !".

In order to distinguish between views taken in different sub-arenas we write, for $\vec{s}$ a well-formed sequence in A, $\ulcorner\vec{s}\urcorner^A$, and similarly for O-views.

In the following lemmas $\vec{s}$ is a legal position of $A \Rightarrow B$, b is the initial B-move which begins $\vec{s}$, a is any initial move in A, and the last move of $\vec{s}$ is m.

Lemma 2.1.8 (O-view Projection)

(i) If m is in B then $\lfloor\vec{s}\rfloor_{A\Rightarrow B} \upharpoonright B = \lfloor\vec{s} \upharpoonright B\rfloor_B = \lfloor\vec{s}\rfloor_{A\Rightarrow B}$

(ii) If m is in the component (A, a) then $\lfloor\vec{s}\rfloor_{A\Rightarrow B} \upharpoonright (A, a)^+ = b \cdot \ulcorner\vec{s} \upharpoonright (A, a)\urcorner^A = \lfloor\vec{s}\rfloor_{A\Rightarrow B}$

Lemma 2.1.9 (Switching Convention) If m and m' are consecutive moves of $\vec{s}$ and are in different components, then m' is a P-move i.e. only Proponent may change components.

Lemma 2.1.10 (P-view Projection)

(i) If m is in B then $\ulcorner \vec{s} \urcorner^{A \Rightarrow B} \upharpoonright B \preceq \ulcorner \vec{s} \upharpoonright B \urcorner^B$

(ii) If m is in (A, a) then $\ulcorner \vec{s} \urcorner^{A \Rightarrow B} \upharpoonright (A, a)^+ \preceq b \cdot \ulcorner \vec{s} \upharpoonright (A, a) \urcorner^A$

(The order $\preceq$ is that of subsequence, which respects pointers in justified sequences.)

Lemma 2.1.11 (Projection Convention)

(i) $\vec{s} \upharpoonright B$ is a legal position in B.

(ii) $\vec{s} \upharpoonright (A, a)$ 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 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 $\vec{s} \cdot m = \vec{s} \cdot m'$. (Equality of justified sequences includes equality of justification pointers, hence the moves m and m', 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 2.1.4.

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 2.2.1 The standard definitions of strategy ([HO00], [Nic96], [McC98]) 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).

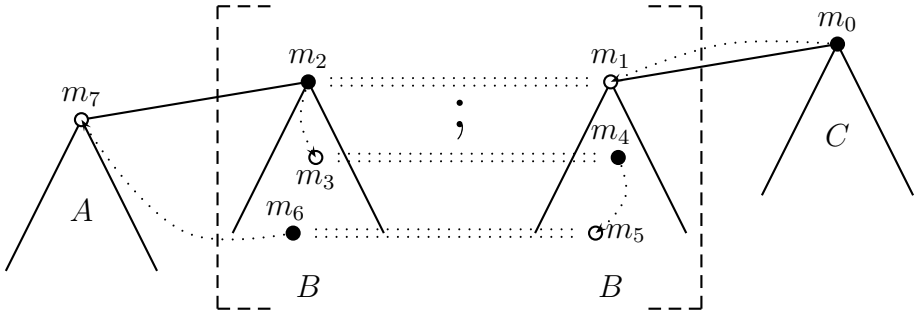
However we prefer to keep to the terminology that an initial move is labelled ε, and add some special cases to the definitions to cope with multiple-tree arenas.

Example 2.2.2 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 $\bot$. We call this the *undefined strategy* or the *empty strategy* (even though it is not actually an empty set).

Definition 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 [HO00]. We first illustrate the ideas with an example.

Example 2.2.3 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 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:

(i) $\vec{s}$ 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;

(ii) $\vec{s} \upharpoonright (B, C)$, 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 $B \Rightarrow C$;

(iii) $\vec{s} \upharpoonright (A, B, b)$, the subsequence of moves in A or B 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 $b \in \vec{s}$.

(iv) $\vec{s} \upharpoonright (A, C)$, 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 a sequence of moves in 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:

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

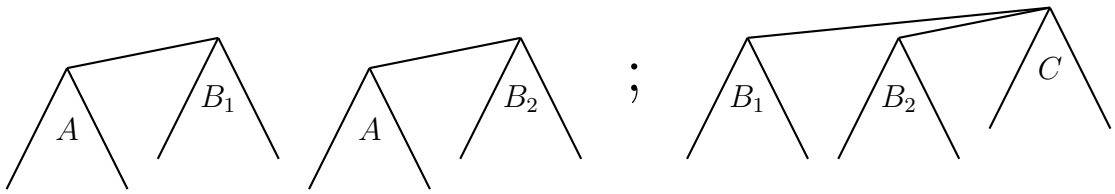
An essentially straightforward result, although very technical in proof, is that composition is well-defined and associative. For a proof see [HO00, 5.1.3].

Lemma 2.2.4

(i) If σ and τ are strategies on $A \Rightarrow B$ and $B \Rightarrow C$ respectively then $\sigma\,;\tau$ is a strategy on $A \Rightarrow C$.

(ii) If, in addition, ρ is a strategy on $C \Rightarrow D$, then $(\sigma\,;\tau)\,;\rho = \sigma\,;(\tau\,;\rho)$

Remark 2.2.5 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_1, B_2, 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 $A \Rightarrow C$.

When analysing a composition it is useful to be able to refer to the intermediate moves which were hidden. The following definition allows us to see all the moves, including the hidden ones, up to but not including the visible move about to occur after a specified legal position of the composite arena.

Definition For strategies σ and τ on $A \Rightarrow B$ and $B \Rightarrow C$ respectively, and a legal position $\vec{s}$ of $A \Rightarrow C$ the *uncovering* of $\vec{s}$ with respect to σ and τ, written $\mathbf{u}(\vec{s}, \sigma, \tau)$, is the unique maximal sequence $\vec{u}$ satisfying the following conditions:

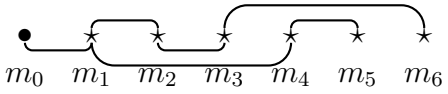
(i) $\vec{u} \in \mathrm{ias}(A, B, C)$;

(ii) $\vec{u} \upharpoonright (A, C) \leq \vec{s}$;

(iii) $\vec{u} \upharpoonright (A, B, b) \in \sigma$ for all initial B-moves $b \in \vec{u}$;

(iv) $\vec{u} \upharpoonright (B, C) \in \tau$.

Example 2.2.6 If the strategies σ and τ and moves m_i are as defined in Example 2.2.3, the uncovering of the minimal legal position (the sequence consisting only of the initial move) shows all the moves of the composition up to but not including the second visible move. That is, $\mathbf{u}(\langle m_0 \rangle, \sigma, \tau)$ is the justified sequence:



This illustrates how we typeset a justified sequence, with justification pointers represented by lines. Note that in an interaction sequence, all the moves occurring in the hidden arena are *both* O- and P-moves, depending on which strategy is looking at them, and hence they are denoted $\star$.

2.3 Innocent and Recursive Innocent Strategies

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 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,

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

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 [HO00, §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 [HO00, §5.3]) is that composition of innocent strategies is well-defined:

Lemma 2.3.1 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*, that is P-moves equipped with a justification pointer back into the argument of the function.

Definition 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 $\bot$ is trivially innocent, and that its innocent function is the everywhere undefined function.

Remark 2.3.2 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 2.1.3, 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 [HO00] and [Nic96] 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 3.1.

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

Definition 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:

(i) $\mathrm{dom}(f)$ is prefix-closed,

(ii) 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^-,

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

(i) $\varepsilon \in \sigma_f$, where ε is the initial move of A.

(ii) If $\vec{s} \in \sigma_f$, $|\vec{s}|$ is even, and $\vec{s} \cdot m$ a legal position of A then $\vec{s} \cdot m \in \sigma_f$

(iii) 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 extra justification pointer has to be relativised 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 2.3.3

(i) $f_{(\sigma_f)} = f$

(ii) $\sigma_{(f_\sigma)} = \sigma$

(iii) $f \subseteq f' \iff \sigma_f \subseteq \sigma_{f'}$

(iv) $\sigma \subseteq \sigma' \iff f_\sigma \subseteq f_{\sigma'}$

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

(i) An innocent strategy is said to be *compact* if the domain of definition of its innocent function is finite.

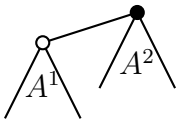
(ii) We say that an innocent strategy is *recursive* if the innocent function representing it is recursive, after some encoding of P-views and justified P-moves.

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,

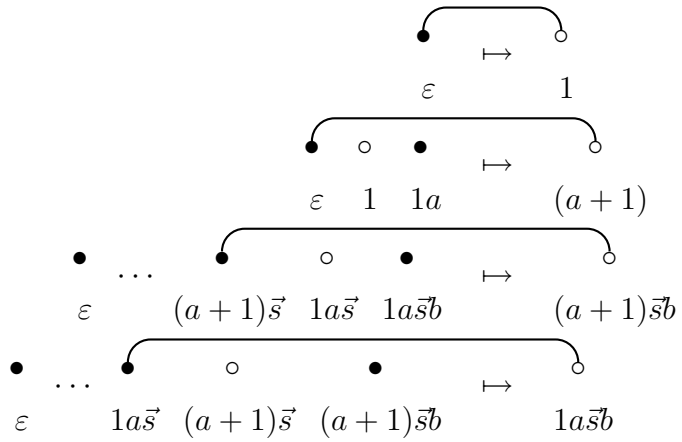
Lemma 2.3.4 The composition of two recursive innocent strategies is itself recursive.

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:



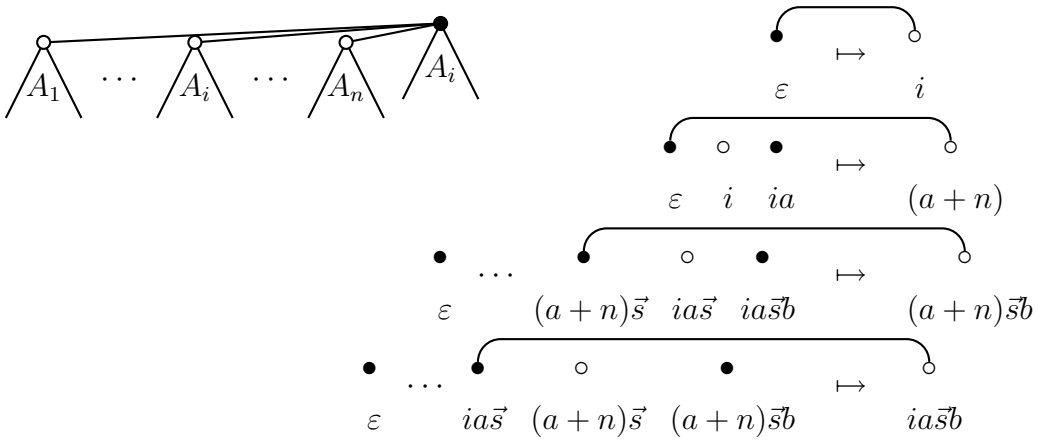
for any sequence $\vec{s}$ which is of appropriate parity of length (odd in the third clause, even in the fourth) and such that the moves $1a\vec{s}$, $1a\vec{s}b$, etc. do actually exist in the arena $A \Rightarrow A$.

We have omitted some justification pointers in the innocent function, but they can be reconstructed from the action of the function on shorter P-views, 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 (copycat strategies were so named in [AJ92]). 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 numbers of appropriate parity of length, and 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 λ-Algebras in Categories of Innocent Strategies

In this section we show that arenas and innocent strategies form a category, which is cartesian closed and has a reflexive object, hence leading to the first game model of the untyped λ-calculus, which we call $\mathcal{D}$. There is also a recursive analogue $\mathcal{D}_{\mathrm{REC}}$.

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

(i) Objects are arenas;

(ii) Morphisms $f : A \to B$ are innocent strategies on the function space arena $A \Rightarrow B$;

(iii) Composition of morphisms is composition as strategies;

(iv) The identity morphism on A is the copycat strategy id_A.

Lemmas 2.2.4, 2.3.1 and the properties of identities show that $\mathbb{A}$ is indeed a category.

Theorem 2.4.1 $\mathbb{A}$ is cartesian closed.

Proof We define products and exponentials as follows:

(i) The terminal object $\mathbf{1}$ is the arena E (defined in Example 2.1.2, the arena consisting of no trees);

(ii) The product of A and B is the product arena $A \times B$;

(iii) Projections are copycat strategies. If $A = \langle A_1, \ldots, A_m \rangle$ and $B = \langle B_1, \ldots, B_n \rangle$ then define

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

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

(iv) The exponential object $(A \Rightarrow B)$ is the function space arena $A \Rightarrow B$;

(v) 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 2.2.2. 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. ■

Definition 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 2.3.4, and the 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 $\mathbb{A}$.

Recall that the arena U, the "maximal" single-tree arena defined in Example 2.1.2, is given in sequence-subset form by $\langle \mathbb{N}^* \rangle$. Observe that $U = U \Rightarrow U$, 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.

Thus we can use the terminology of Proposition 1.6.3 to make the following definition.

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

Propositions 1.6.3, 1.6.4 and 1.5.2 then show the following.

Theorem 2.4.2 $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ form $\lambda\eta$-algebras.

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 interpretation function $[\![-]\!]_\rho$ is the same in both $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ (except that the valuation ρ may map variables to non-recursive elements in $\mathcal{D}$) and we write $[\![s]\!]$ for the strategy which denotes the closed term s in either model.

Remark 2.4.3 All arenas (or r.e. arenas in the case of $\mathbb{A}_{\mathrm{REC}}$) are not necessary to form a CCC leading to the λ-algebras $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$. It is sufficient to consider the category which we call $\mathbb{U}$, where the objects are only the arenas made up of trees which look like the arena U. That is, the objects are E and finite products U^n. This is easily seen to be a full sub-CCC of $\mathbb{A}$, and still possesses the same reflexive object, hence $\mathcal{D}$ could equally be presented as $\mathcal{M}(\mathbb{U}, U, \mathit{Fun}, \mathit{Gr})$. The recursive analogue is called $\mathbb{U}_{\mathrm{REC}}$, the full sub-CCC of $\mathbb{A}_{\mathrm{REC}}$ consisting of the same objects and recursive strategies.

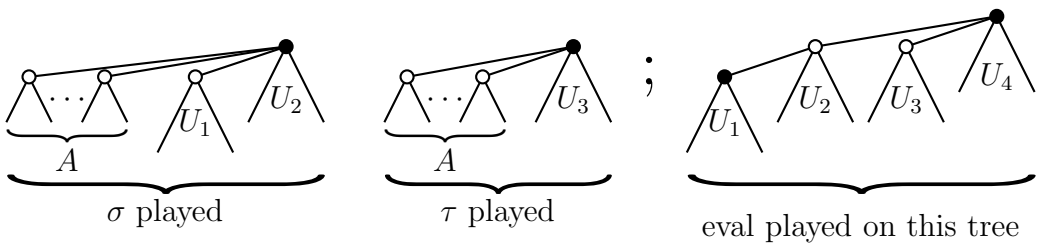
We prefer to present the full categories $\mathbb{A}$ and $\mathbb{A}_{\mathrm{REC}}$, but there are some definitions in what follows which are really aimed at the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, and which gain a certain amount of complexity when defined in full generality. We will still make the more general definitions, but comment when simpler definitions would be sufficient for $\mathbb{U}$ and $\mathbb{U}_{\mathrm{REC}}$.

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

Definition 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 2.2.5).

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 $\tau \,;\, \sigma$.

2.5 Properties of $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$

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 they 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 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. We call $\sigma\|_B$ "the restriction of σ to 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 2.5.1 If $\sigma : A \Rightarrow B$, A' is a sub-arena of A and B' is a sub-arena of B then

$$\begin{aligned} \sigma\|_{A\Rightarrow B'} &= \sigma\,;\iota_{B'} \\ \sigma\|_{A'\Rightarrow B} &= \iota_{A'}\,;\sigma \end{aligned}$$

where $\iota_{A'} : A \to A$ and $\iota_{B'} : B \to B$ are the identity strategies on A and B restricted to $A' \Rightarrow A'$ and $B' \Rightarrow B'$ respectively.

Proof A straightforward modification of the proof that $\mathrm{id}_A\,;\sigma = \sigma = \sigma\,;\mathrm{id}_B$ ■

Corollary 2.5.2 If $\sigma : A \Rightarrow B, \tau : B \Rightarrow C$, B' is a sub-arena of B and C' a sub-arena of C then

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

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

Definition

(i) $U_0 = M$, the "minimal" single-tree arena defined in Example 2.1.2,
(ii) $U_{n+1} = U_n \Rightarrow U_n$,
(iii) If σ is an innocent strategy on U then $\sigma_n = \sigma\|_{U_n}$.

Note that $U = \bigcup_{n\in\omega} U_n$. This particular definition of U_n also has the following key property:

Lemma 2.5.3 $\mathrm{eval}_{U,U}\|_{(U_{n+1}\times U)\Rightarrow U} = \mathrm{eval}_{U,U}\|_{(U\times U_n)\Rightarrow U_n}$

Proof As strategies we know that $\mathrm{eval}_{U,U}$ and $\mathrm{id}_{U\Rightarrow U}$ are identical. Further, from Lemma 2.5.1 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:

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

■

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

Lemma 2.5.4 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) $\sigma_0 = \bot$,
(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 2.5.3. ■

Theorem 2.5.5 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 [Bar84, §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 [Hyl76] and Wadsworth in [Wad76]. The properties of the approximation are used to show that labelled reduction is monotone in the model, and hence that

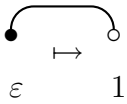
labelled β-normal forms are maximal. This gives rise to an approximation theorem — in the model any term is the union of its *approximate normal forms* (introduced by Wadsworth in [Wad71]). Finally, it is simple to show that the only approximate normal forms of unsolvable terms have denotation $\bot$. ■

Remark 2.5.6 It would now be possible to use a fairly simple and standard argument to show that the local structure of the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ is the theory $\mathcal{H}^*$. (The argument parallels that for Scott's model D_∞ given in [Bar84, §19.2].) Rather than prove this result using approximation, however, we prefer to wait until the following chapter, when the local structure of the models will be a simple corollary to the Exact Correspondence Theorem 3.3.1.

We now describe some of the less satisfying properties of the models.

Theorem 2.5.7 $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ are not universal, not extensional and not even weakly extensional, thus they are not λ-models.

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



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

We may consider $\bot$ and $\bot'$ as defining morphisms $\mathbf{1} \to U$, and so as elements of the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$. Certainly we have $\bot \neq \bot'$. However, the fact that neither $\bot$ nor $\bot'$ 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 $\sigma : \mathbf{1} \to U$,

$$\bot \bullet \sigma = \bot' \bullet \sigma = \bot.$$

Hence $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ are not extensional. We know that $[\![I]\!] = [\![1]\!]$ in $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, since they are $\lambda\eta$-algebras, and so by Lemma 1.6.2 they cannot be weakly extensional either (so they are not λ-models).

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 $[\![s]\!] = \bot$. ■

Thus $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ contain undefinable elements — large numbers of them in fact. For any term s, all of the nontrivial compact approximants $([\![s]\!])_n$ are undefinable. Finding an improved version of these models is the subject of the next chapter.

Chapter 3

Effectively Almost-Everywhere Copycat Strategies and the Model $\mathcal{D}_{\mathrm{EAC}}$

The reason that $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ failed to be universal, or even weakly extensional, is the existence of strategies like $\bot'$ defined in the proof of Theorem 2.5.7. This strategy has the same applicative behaviour as $\bot$, 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; this is the aim of this chapter.

To do so we first present an *economical* way to encode strategies, together with a representation of Nakajima trees which is *variable-free* (i.e. all bound variables are replaced by syntax-free encoded "pointers", which could 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 $\mathcal{H}^*$.

Using the characterisation of Böhm-like trees which are the Böhm trees of some term, given in Section 1.5, 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. To examine the extensionality properties of the model we prove a powerful result which we call the *Separation Lemma*. This enables us to show that $\mathcal{D}_{\mathrm{EAC}}$ is extensional (in fact order-extensional), and also to give a semantic proof of a separation result for the theory $\mathcal{H}^*$. The latter extends to a generalisation of Böhm's Theorem on separability in the theory λ.

3.1 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 2.1.3 and 2.3.2.

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

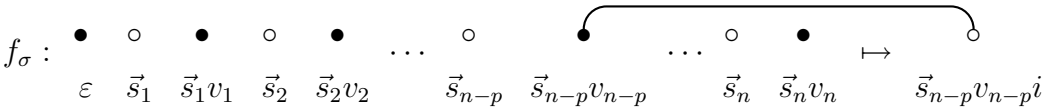
(i) We know that the domain of an innocent function is made up of P-views. Lemma 2.1.6 shows that each non-initial O-move in any P-view must be a child of the previous move, and the initial move must be ε.

(ii) 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.

(iii) 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:

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



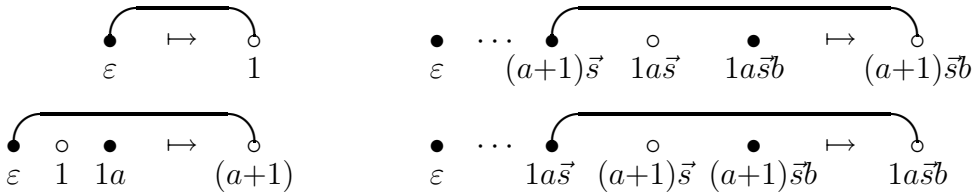
(We set $p = n$ when the resulting move in the clause of f_σ is a child of the initial move, intending "$\vec{s}_0 v_0$" to mean the sequence $\langle \varepsilon \rangle$.)

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 is encoded by the sequence $\vec{s}_i \cdot v_i$, which by definition is the v_i^{th} child of the move encoded 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 3.1.1 Recall that the following is the innocent function of the copycat strategy id_U:



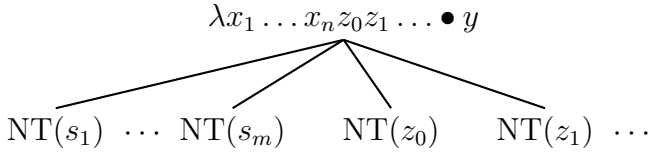
Here $\vec{s}$ ranges over sequences of appropriate parity, a and b over positive natural numbers. The reader is invited to check that the economical form of this strategy is given by: $\varepsilon \mapsto (1,0), i \mapsto (i+1,1)$ and for nonempty sequences $\vec{v}$, $\vec{v}i \mapsto (i,1)$.

3.2 Nakajima Trees and Variable-Free Form

Recall that definition of Nakajima tree from Section 1.5: for a λ-term s the *Nakajima tree* of s, written $\mathrm{NT}(s)$, is the tree, partially labelled with elements of $\{\lambda x_1 x_2 \ldots \bullet\, y \mid x_1, x_2, \ldots, y \text{ variables}\}$, defined inductively as follows.

$\mathrm{NT}(s) = \bot$ if s is unsolvable

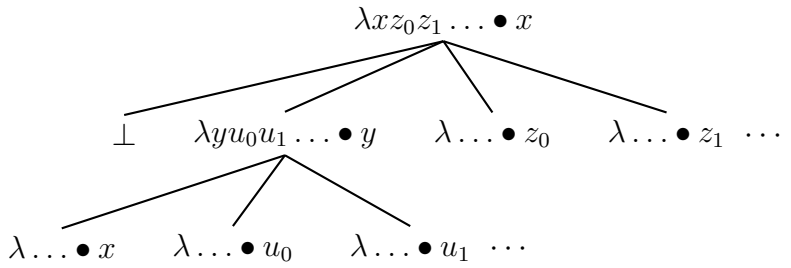
$\mathrm{NT}(s) =$ if s has HNF $\lambda x_1 \ldots x_n . y s_1 \ldots s_m$



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

In order to work modulo α-conversion, we would like to follow de Bruijn (as in [dB72]) 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 abstracted over either at the node where the head variable occurs, or 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 3.2.1 We illustrate this with an example. Recall from Example 1.5.5 that 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 $\lambda \ldots \bullet z_0$ 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_0. 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, and were we to try to build the variable-free form by breaking down the Nakajima Tree we would uncover countably many free variables under each abstraction. 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 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:

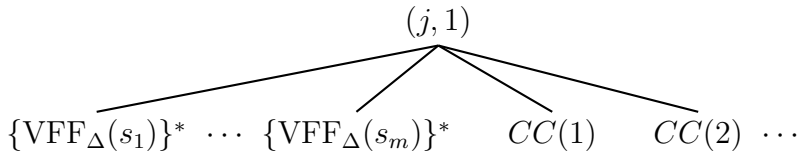
(i) those labelled (i, d) are relabelled $(i + n, d)$;

(ii) those labelled $(i, d+1)$ for $i \leq n$ are relabelled $(n - i + 1, d)$;

(iii) 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:

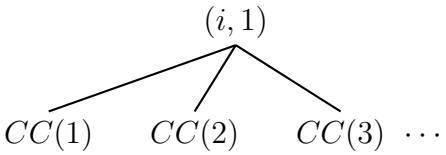
$$
\begin{array}{rcll}
\mathrm{VFF}_\Delta(s) & = & \bot \text{ , for unsolvable } s. & \\
\mathrm{VFF}_\Delta(\lambda x_1 \ldots x_n \bullet s) & = & \{\mathrm{VFF}_{\Delta\cdot\langle x_1,\ldots,x_n\rangle}(s)\}^n, & \text{if } s \text{ is of the form } y s_1 \ldots s_m, \\
 & & & \text{for some variable } y \\
\mathrm{VFF}_\Delta(v_j s_1 \ldots s_m) & = & & \\
\end{array}
$$



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

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

$CC(i) =$



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 3.2.2 This definition coincides with the informal notion of variable-free form described earlier. Formally, suppose that s is a term with free variables 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 $x^{\vec{a}}_1, x^{\vec{a}}_2, \ldots$ (in that order). Let this renamed Nakajima tree have labelling function A, and consider $\mathrm{VFF}_\Delta(s)$ also as a labelling function.

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

(i) If $\vec{a} \notin \mathrm{dom}(A)$ then $\mathrm{VFF}_\Delta(s)$ is unlabelled or undefined at $\vec{a}$,

(ii) If $A(\vec{a}) = \lambda x^{\vec{a}}_1 x^{\vec{a}}_2 \ldots \bullet v_j$ then $\mathrm{VFF}_\Delta(s)(\vec{a}) = (j, p+1)$,

(iii) If $A(\vec{a}) = \lambda x^{\vec{a}}_1 x^{\vec{a}}_2 \ldots \bullet x_j^{\langle a_1,\ldots,a_{p-r}\rangle}$ then $\mathrm{VFF}_\Delta(s)(\vec{a}) = (j, r)$.

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:

(i) If s is unsolvable then $\varepsilon \notin \mathrm{dom}(A)$ and $\mathrm{VFF}_\Delta(s)$ is unlabelled at the root.

(ii) 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

$$\begin{array}{rl} & \mathrm{VFF}_\Gamma(v_j s_1 \ldots s_m)(\varepsilon) = (j+n, 1) \\ \text{hence} & \mathrm{VFF}_\Delta(s)(\varepsilon) = \{\mathrm{VFF}_\Gamma(v_j s_1 \ldots s_m)\}^n(\varepsilon) = (j, 1). \end{array}$$

(iii) 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

$$\begin{array}{rl} & \mathrm{VFF}_\Gamma(x_j s_1 \ldots s_m)(\varepsilon) = (n-j+1, 1) \\ \text{hence} & \mathrm{VFF}_\Delta(s)(\varepsilon) = \{\mathrm{VFF}_\Gamma(v_j s_1 \ldots s_m)\}^n(\varepsilon) = (j, 0). \end{array}$$

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 rename 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:

(i) $i \leq m$ 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 $i \cdot \vec{a}$.

(ii) $i \leq m$ 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

$$\begin{array}{rl} \text{by the induction hypothesis} & \mathrm{VFF}_\Gamma(s_i)(\vec{a}) = (j+n, p+1) \\ \text{hence} & \{\mathrm{VFF}_\Gamma(s_i)\}^*(\vec{a}) = (j+n, p+2) \\ \text{hence} & \mathrm{VFF}_\Gamma(y s_1 \ldots s_m)(i \cdot \vec{a}) = (j+n, p+2) \\ \text{hence} & \mathrm{VFF}_\Delta(s)(i \cdot \vec{a}) = (j, p+2). \end{array}$$

(iii) $i \leq m$ 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

$$\begin{aligned} \text{by the induction hypothesis} \quad & \mathrm{VFF}_\Gamma(s_i)(\vec{a}) = (n-j+1, p+1) \\ \text{hence} \quad & \{\mathrm{VFF}_\Gamma(s_i)\}^*(\vec{a}) = (n-j+1, p+2) \\ \text{hence} \quad & \mathrm{VFF}_\Gamma(y s_1 \ldots s_m)(i \cdot \vec{a}) = (n-j+1, p+2) \\ \text{hence} \quad & \mathrm{VFF}_\Delta(s)(i \cdot \vec{a}) = (j, p+1). \end{aligned}$$

(iv) $i \leq m$ 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

$$\begin{aligned} \text{by the induction hypothesis} \quad & \mathrm{VFF}_\Gamma(s_i)(\vec{a}) = (j, r) \\ \text{hence} \quad & \{\mathrm{VFF}_\Gamma(s_i)\}^*(\vec{a}) = (j, r) \\ \text{hence} \quad & \mathrm{VFF}_\Gamma(y s_1 \ldots s_m)(i \cdot \vec{a}) = (j, r) \\ \text{hence} \quad & \mathrm{VFF}_\Delta(s)(i \cdot \vec{a}) = (j, r). \end{aligned}$$

(v) $i > m$ 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,

$$\begin{aligned} & \mathrm{VFF}_\Gamma(y s_1 \ldots s_m)(i) = (i-m, 1) \\ \text{so that} \quad & \mathrm{VFF}_\Delta(s)(i) = \{\mathrm{VFF}_\Gamma(y s_1 \ldots s_m)\}^n(i) = (i-m+n, 1). \end{aligned}$$

(vi) $i > m$ 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,

$$\begin{aligned} & \mathrm{VFF}_\Gamma(y s_1 \ldots s_m)(i \cdot \vec{a}) = (j, 1) \\ \text{so that} \quad & \mathrm{VFF}_\Delta(s)(i \cdot \vec{a}) = (j, 1). \end{aligned}$$

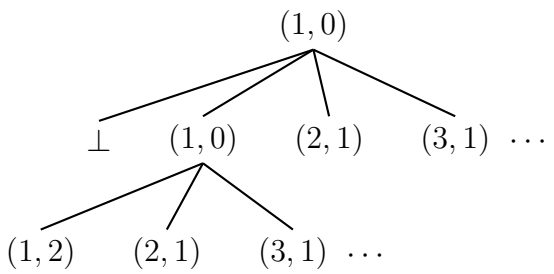
We see that the result holds in each case.

■

Remark 3.2.3 The construction of $\mathrm{VFF}(s)$ from s is not a computable procedure, because the predicate of solvability is not decidable. The same applies to the definition we have given for Böhm tree and Nakajima tree (see Remark 1.5.3). However Lemma 3.2.2 does give a recursive translation of Nakajima trees into variable-free form.

The connexion between $\mathrm{NT}(s)$ and $\mathrm{VFF}(s)$, described formally in Lemma 3.2.2, is now illustrated. Take the term $s = \lambda x.x\Omega(\lambda y.yx)$, the Nakajima tree of which is given above in Example 3.2.1. 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 3.2.2 means at each pictured node.

3.3 Exact Correspondence and Local Structure

The representation of an innocent strategy described in Section 3.1, and that of Nakajima tree in Section 3.2 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 3.3.1 (Exact Correspondence Theorem) 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.

In particular for closed terms s, $[\![s]\!]_\varepsilon = \mathrm{VFF}_\varepsilon(s)$.

Proof 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 $l+1$,

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

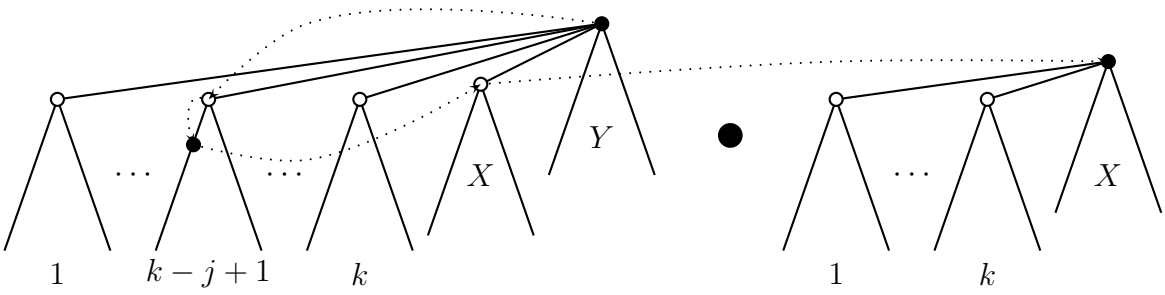
which implies the required result.

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,

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

and 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.

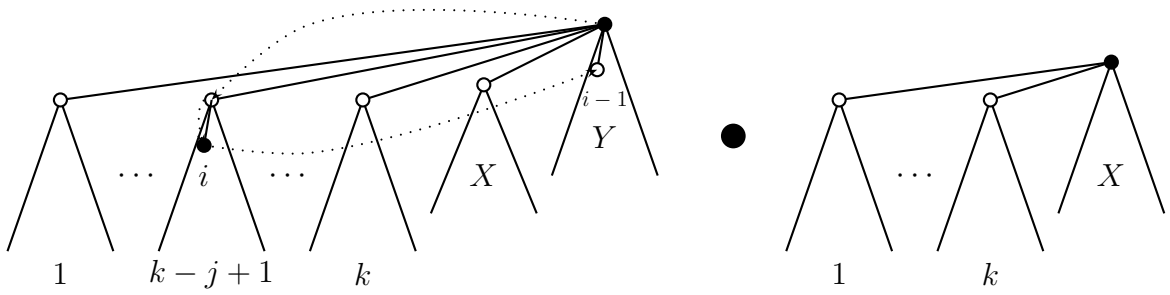
Now the induction hypothesis is that down to depth l of the economical form, the strategy σ is $\{\mathrm{VFF}_{\Delta}(s')\}^k$. By the definition of $\Pi^{\Delta}_{v_j}$ the initial move of the composition is $k-j+1$ so the root of the economical form tree is $(k-j+1, 0)$.

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 $i \leq k$. 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

$$\mathrm{VFF}_\Delta(t) = \quad (j,1) \quad \{\mathrm{VFF}_\Delta(s')\}^* \quad CC(1) \quad CC(2) \quad \cdots$$

Thus

$$\{\mathrm{VFF}_\Delta(t)\}^k = \quad (k-j+1,0) \quad \{\mathrm{VFF}_\Delta(s')\}^\times \quad CC(k+1) \quad CC(k+2) \quad \cdots$$

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

(i) nodes at depth d labelled $(i,d+1)$ for $i \le k$ are relabelled $(k-i+1,d+1)$;

(ii) 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. ■

Example 3.3.2 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:

$$\begin{array}{rcl@{\qquad}rcl} \varepsilon & \mapsto & (1,0) & \langle 4\rangle & \mapsto & (3,1) \\ \text{undefined on } \langle 1\rangle & & & \langle 21\rangle & \mapsto & (1,2) \\ \langle 2\rangle & \mapsto & (1,0) & \langle 22\rangle & \mapsto & (2,1) \\ \langle 3\rangle & \mapsto & (2,1) & \langle 23\rangle & \mapsto & (3,1). \end{array}$$

We noted in Remark 2.5.6 that the local structure of the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ can be examined using the properties of the approximation for strategies proved in Lemma 2.5.4. However we did not do so, because the local order structure of the models follows immediately from the Exact Correspondence Theorem.

Corollary 3.3.3 In the models $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, for closed terms s and t:

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

The order $\subseteq$ on the model is inclusion of strategies. That on Nakajima trees is inclusion of labelling function, modulo renaming of bound variables, which 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:

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

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

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

Equivalence 2 was proved by Nakajima in [Nak75, 3.5].

Equivalence 3 is Hyland's local structure theorem for D_∞ and a proof can be found in, for example, [Bar84, §19.2].

The fact that the local structure is the $\lambda\eta$-theory $\mathcal{H}^*$ follows immediately from Lemma 1.5.7. ■

3.4 Effectively Almost-Everywhere Copycat Strategies

Now that we have another way to see what the denotations of terms are, we can translate what we know 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 $\mathbb{A}_{\mathrm{REC}}$ using them.

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

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

(i) If $\vec{s} \in A$ then $A \mathbin{@} \vec{s} = \{\vec{t} \mid \vec{s} \cdot \vec{t} \in A\}$.

(ii) 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 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:

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

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. It is the O-move just before P is about to play as directed by the innocent function described economically by f, 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 Consider an innocent strategy in economical form $f : \mathbb{N}^* \to \mathbb{N} \times \mathbb{N}_0$, over some single-tree arena A.

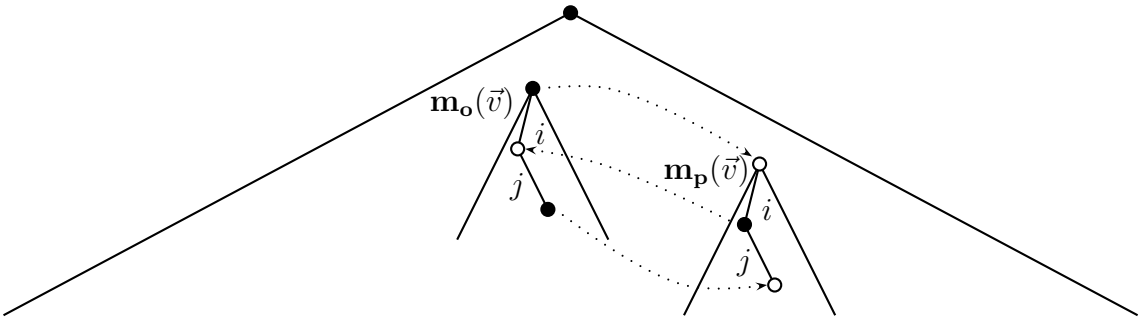
We say that f is *everywhere copycat* (EC) at $\vec{v} \in \mathbb{N}^*$ if f is undefined at $\vec{v}$ or the following hold:

(i) The arenas $A \mathbin{@} \mathbf{m_o}(\vec{v})$ and $A \mathbin{@} \mathbf{m_p}(\vec{v})$ are order-isomorphic (with respect to the prefix ordering, considered as subsets of $\mathbb{N}^*$);

(ii) 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, $f(\vec{w} \cdot i) = (i, 1)$;

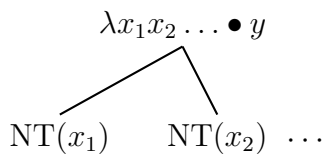
(iii) If $f(\vec{v}) = (i, p)$ then $p > 0$.

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 $A \mathbin{@} \mathbf{m_o}(\vec{v})$ and that on the right represents $A \mathbin{@} \mathbf{m_p}(\vec{v})$ — 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 also 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:



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

Definition 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

(i) 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;

(ii) 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 $\vec{v} \cdot i$;

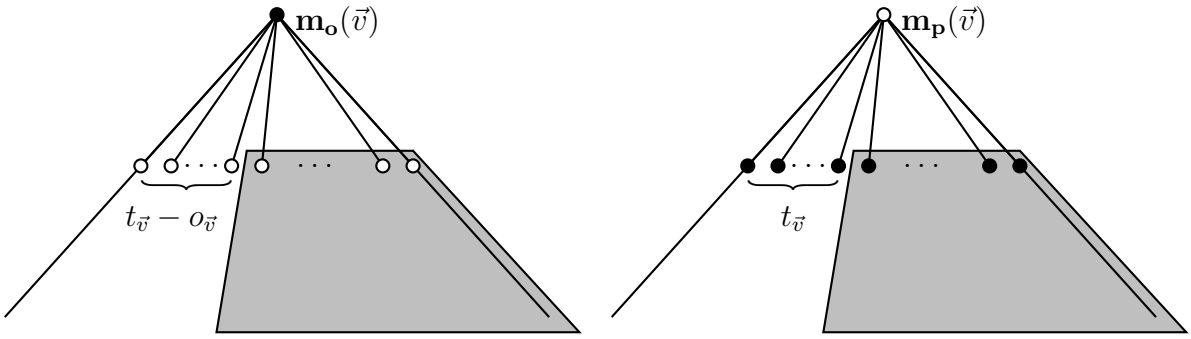
(iii) For all $\vec{w} \geq (\vec{v} \cdot k)$ with $k \leq t_{\vec{v}}$, if $f(\vec{w}) = (i,\ |\vec{w}|\ -\ |\vec{v}|\ )$ then $i \leq t_{\vec{v}} - o_{\vec{v}}$;

(iv) 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 $\vec{v}$. 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 $\mathbf{m_o}(\vec{v})$. Condition (i) says that the two shaded areas are isomorphic, and (ii) forces P to simply copy O's moves between them. Condition (iii) means that subsequent O-moves *not* in the shaded areas cannot lead to P-moves in the shaded areas, and (iv) is just that $\mathbf{m_p}(\vec{v})$ is not part of the shaded area on the left. (Note that there is nothing to say that shaded areas pictured cannot overlap.)

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_v and o_v are *valid* copycat threshold and offset if f satisfies the conditions (i)-(iv) of AC at that P-view with those particular values.

Remark 3.4.1 In practice we are only interested in EAC strategies over the arena U, in which case the definitions can be slightly simplified. For example, the conditions for a strategy to be AC at a P-view $\vec{v}$ become:

(i) For all $i > t_v$, $f(\vec{v} \cdot i) = (i - o_v, 1)$ and for all $\vec{w} \in \mathbb{N}^*$ and $j \in \mathbb{N}$, $f(\vec{v} \cdot i \cdot \vec{w} \cdot j) = (j, 1)$;

(ii) For all $\vec{w} \geq (\vec{v} \cdot k)$ with $k \leq t_v$, if $f(\vec{w}) = (i, |\vec{w}| - |\vec{v}|)$ then $i \leq t_v - o_v$;

(iii) If $f(\vec{v}) = (i, 0)$ then $i \leq t_v - o_v$.

In fact, the definition is irrelevant for finitely branching arenas, as the following lemma shows.

Lemma 3.4.2 Every strategy is AC at any P-view $\vec{v}$ such that both $\mathbf{m_o}(\vec{v})$ and $\mathbf{m_p}(\vec{v})$ have finitely many children. Thus on a finitely branching arena every strategy is EAC.

Proof Suppose that $\mathbf{m_o}(\vec{v})$ and $\mathbf{m_p}(\vec{v})$ have p and q children respectively in the arena. Then $t_v = q$ and $o_v = q - p$ are a valid threshold and offset for any innocent strategy to be AC at $\vec{v}$, because both the arenas in (i) are empty, none of the moves in question for condition (ii) exist, and the conditions $i \leq t_v - o_v$ in (iii) and (iv) hold automatically. ■

Lemma 3.4.3 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_v and o_v respectively, then for any $t' \geq t_v$, f is also AC at the P-view $\vec{v}$ with threshold and offset t' and o_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).

Proof It is easy to check that increasing the threshold only relaxes the conditions (i), (ii) and (iv) on f in the definition of AC. To check that condition (iii) still holds, suppose that $\vec{w} \geq \vec{v} \cdot k$. When $k \leq t_v$, (iii) is just a weaker condition on f than before. When $t_v < k \leq t'$ there are two cases:

1. If $\vec{w} = \vec{v} \cdot k$ then by condition (ii) of the fact that t_v is a valid threshold for f at $\vec{v}$, $f(\vec{w}) = (k - o_v, 1)$. Hence $f(\vec{w}) = (i, |\vec{w}| - |\vec{v}|)$ with $i = k - o_v \leq t' - o_v$ as required.

2. If $\vec{w} = \vec{v} \cdot k \cdot \vec{v'} \cdot i$ then f is EC at $\vec{v} \cdot k \cdot \vec{v'}$, so $f(\vec{w}) = (i, 1)$. Hence $f(\vec{w}) \neq (i, |\vec{w}| - |\vec{v}|)$.

In either case (iii) holds. ■

This means that the copycat thresholds of an EAC strategy are not unique, and indeed we will return to this in much greater detail in the next chapter.

Remark 3.4.4 At each P-view of an EAC strategy there will be a *least copy-cat threshold*, the least value for t_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.

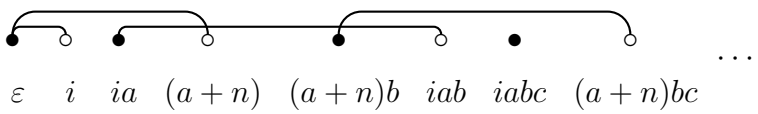
The following proof is important for what follows, and also serves as an illustration of the definition of EAC strategies.

Lemma 3.4.5 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:

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

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



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

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. ■

The following requires a highly technical proof. We direct the reader to a result from Chapter 4, namely Theorem 4.2.2. That result certainly belongs to the later chapter and it would be pointless to repeat its proof here, but we apologise for the nonlinearity.

Lemma 3.4.6 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 The relevant part of Theorem 4.2.2 is the following:

Let σ and τ be as above, and suppose the copycat threshold and offset of σ at the minimal P-view are t_σ and o_σ, and those of τ are t_τ and o_τ. As long as $t_\sigma - o_\sigma \geq |A|$ and $t_\tau - o_\tau \geq |B|$ then there is a recursive way to calculate valid thresholds and offsets for each P-view of $\sigma\,;\tau$ from those of σ and τ. (Here $|A|$ means the number of trees in the arena A).

Now it may be that the above inequalities are not satisfied, but we can define new thresholds for σ and τ at the minimal P-view as follows: $t'_\sigma = \min(t_\sigma, o_\sigma + |A|)$ and $t'_\tau = \min(t_\tau, o_\tau + |B|)$. Lemma 3.4.3 shows that these are still valid thresholds, and we can apply Theorem 4.2.2 — since the procedure for computing the composite threshold and offset is recursive this also ensures that the composition's threshold and offset functions are recursive. Finally, Lemma 2.3.4 tells us that the composite innocent strategy is recursive. ■

Definition 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 3.4.7 $\mathbb{A}_{\mathrm{EAC}}$ is a cartesian closed category, in fact a lluf subcategory[1] 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 3.4.5 shows that the same identity strategies are morphisms of $\mathbb{A}_{\mathrm{EAC}}$. Lemma 3.4.6 completes the proof that it is a category.

Lemma 3.4.5 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 $\mathbb{A}_{\mathrm{REC}}$. ■

Remark 3.4.8 In the same way that we can define the full subcategories $\mathbb{U}$ and $\mathbb{U}_{\mathrm{REC}}$ of $\mathbb{A}$ and $\mathbb{A}_{\mathrm{REC}}$ (see Remark 2.4.3), we can consider the full subcategory $\mathbb{U}_{\mathrm{EAC}}$ of $\mathbb{A}_{\mathrm{EAC}}$ which has as objects only arenas U^n for $n \in \mathbb{N}^0$. Working in this category we could use the marginally simpler definition of EAC given in Remark 3.4.1.

3.5 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:

[1] by a *lluf subcategory* of a category $\mathbb{C}$ we mean a subcategory of $\mathbb{C}$ which has the same class of objects as $\mathbb{C}$. See e.g. [Cro93, §2.3].

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

As for $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, $\mathcal{D}_{\mathrm{EAC}}$ could equivalently be defined as $\mathcal{M}(\mathbb{U}_{\mathrm{EAC}}, U, \mathit{Fun}, \mathit{Gr})$.

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 $[\![-]\!]_\rho$ of $\mathcal{D}_{\mathrm{EAC}}$ is the same as that of $\mathcal{D}_{\mathrm{REC}}$ (except that in $\mathcal{D}_{\mathrm{EAC}}$ the valuation ρ may only map variables to EAC strategies). Hence,

Theorem 3.5.1 $\mathcal{D}_{\mathrm{EAC}}$ 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 3.3.3).

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 3.5.2 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 $\mathbb{N}^* \rightharpoonup \mathbb{N} \times \mathbb{N}_0$) $\mathrm{VFF}_\varepsilon(s) = f$.

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

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

$$\begin{array}{l} \varepsilon \in X \\ \text{if } \vec{v} \in X \text{ and } \vec{v} \in \mathrm{dom}(f) \text{ then } 1 \leq i \leq t_v \implies \vec{v} \cdot i \in X. \end{array}$$

Then X has the following properties:

(i) If $\vec{v} = \vec{u} \cdot i$ with $\vec{u} \in X$ and $\vec{v} \notin X$ then $i > t_u$, so $f(\vec{v}) = (i - o_u, 1)$ (by clause (ii) of the definition of AC).

(ii) 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).

The intuition is that X is a subset of $\mathbb{N}^*$, coding a subset of the possible P-views of U, and at every such P-view f's behaviour is *not* determined by the conditions of EAC. We say that f is given *explicitly* at every P-view in X, as opposed to the P-views not in X, when its value is forced by the copycat thresholds and offsets of f. We will return to this idea in Chapter 4.

We define the labelling function of Böhm-like tree A (by what we call *copycat collapse*) as follows. The shape of the tree A (i.e. the set of nodes at which either its labelling function is defined or the node is $\perp$) is the set X.

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

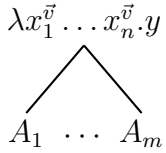
(i) If $\vec{v} \notin \mathrm{dom}(f)$ then $A(\vec{v})$ is undefined (the partially-labelled tree has label $\bot$ at this node). Note that if $\vec{v} \notin \mathrm{dom}(f)$ then $\vec{v}$ codes a terminal node of A.

(ii) 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 $n = t_v - o_v$.

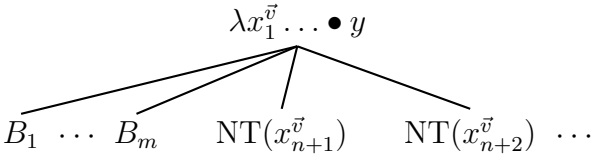
Now there are no free variables in the Böhm-like tree A (since in the second clause above, if $f(\langle v_1, \ldots, v_p\rangle) = (i, r)$ then $r \leq p$). Since the functions $\vec{v} \mapsto t_v$ and $\vec{v} \mapsto o_v$ are recursive, and f is recursive, and our procedure for computing $A(\vec{v})$ from these is clearly recursive, A itself must be a recursive labelling function, i.e. the Böhm-like tree A is r.e. Thus we can appeal to Theorem 1.5.4 to show that 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 3.2.2. 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 $\bot$, so is $\mathrm{NT}(s)$. At a node of the tree labelled by A of the form



we can deduce that the tree labelled by B has the corresponding subtree of the form



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

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

(i) $\vec{v} \in X$ 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 3.2.2 $\vec{v} \notin \mathrm{dom}(\mathrm{VFF}_\varepsilon(s))$.

(ii) $\vec{v} \in X$ and $f(\vec{v}) = (i, r)$. Then we know that $A(\vec{v}) = \lambda x_1^{\vec{v}} \ldots x_n^{\vec{v}}.x_i^{\langle v_1,\ldots,v_{p-r}\rangle}$, so $B(\vec{v}) = \lambda x_1^{\vec{v}} \ldots \bullet\, x_i^{\langle v_1,\ldots,v_{p-r}\rangle}$. Hence, by Lemma 3.2.2, $\mathrm{VFF}_\varepsilon(s)(\vec{v}) = (i, r)$.

(iii) $\vec{v} \notin X$ but $\vec{v} = \vec{u} \cdot i$ with $\vec{u} \in X$. Then we know that $f(\vec{v}) = (i - o_u, 1)$. Also we know that $A(\vec{u}) = \lambda x_1^{\vec{u}} \ldots x_n^{\vec{u}}.y$, and has m descendents, for some variable y and where $m = t_u$ and $n = t_u - o_u$. Therefore by examining the diagrams comparing nodes of Böhm and Nakajima trees above, we see that $B(\vec{v}) = \lambda x_1^{\vec{v}} \ldots \bullet\, x_{i-m+n}^{\vec{u}}$. Hence, by Lemma 3.2.2, $\mathrm{VFF}_\varepsilon(s)(\vec{v}) = (i - m + n, 1) = (i - o_u, 1)$.

(iv) $\vec{v} \notin X$ 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 $B(\vec{v}) = \lambda x_1^{\vec{v}} \ldots \bullet x_i^{\vec{u}}$. Hence, by Lemma 3.2.2, $\mathrm{VFF}_\varepsilon(s)(\vec{v}) = (i, 1)$.

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

■

Combining Lemma 3.5.2 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 3.5.3 $\mathcal{D}_{\mathrm{EAC}}$ forms a universal $\lambda\eta$-algebra.

Remark 3.5.4 In order to construct a term which is denoted by a specified element of the model, i.e. an EAC strategy on U, we need to know a valid copycat threshold and offset for each P-view. If only given the moves of the strategy finding thresholds cannot be done effectively. We will return to this question in Chapter 4.

3.6 The Separation Lemma

This section is devoted to the proof and illustration of the *Separation Lemma*, which will be of key importance in our semantic proof of extensionality properties of the model $\mathcal{D}_{\mathrm{EAC}}$. An intuitive description of the force of the lemma is as follows: for any clause of the innocent function of an innocent strategy, we can find a compact innocent O-strategy which activates this clause when played against it.

Lemma 3.6.1 (The Separation Lemma) Given an EAC innocent P-strategy σ on U and any odd legal position $\vec{s}$ such that $\vec{s} \in \sigma$, there exists a compact innocent O-strategy τ such that the play of σ against τ, $\sigma \cap \tau$, contains a finite legal position $\vec{t}$ with $\vec{s} \preceq \vec{t}$ (the subsequence order respects justification pointers) and $\ulcorner \vec{s} \urcorner = \ulcorner \vec{t} \urcorner$.

(An example follows this proof; the reader may wish to follow this to illustrate the details of the construction.)

Proof Let $\vec{s} = \langle m_0 = \varepsilon, \langle m_1, p_1 \rangle, \ldots, \langle m_{2n}, p_{2n} \rangle \rangle$ where the p_i are the standard encodings of justification pointers. For $0 \leq k \leq 2n$ write $\vec{s}_k = \langle \varepsilon, \langle m_1, p_1 \rangle, \ldots, \langle m_k, p_k \rangle \rangle$.

Then for $0 \leq k < n$ we can take the P-view of $\vec{s}_{2k+1}$ and extract a sequence $\vec{q_k}$, which codes both the P-move m_{2k+1} and the O-move m_{2k+2} with respect to σ.

So we can define $T_k = t_{\vec{q_k}}$, $O_k = o_{\vec{q_k}}$, the copycat threshold and offset of σ at the move m_{2k+1}. Also set $F = \max\{i \mid f_\sigma(\vec{q_k}) = (i, r),\ 0 \leq k < n\} + \max\{O_k \mid 0 \leq k < n\}$. We then set

$$T_k^* = \max\left(\{T_i \mid m_{2i+1} = m_{2k+1},\ 0 \leq i < n\} \cup \{F\}\right)$$

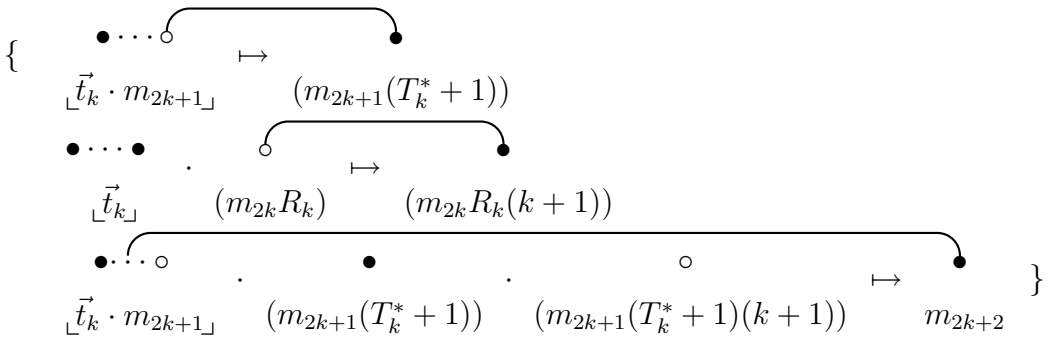
and write $R_k = T_k^* + 1 - O_k$ for tidiness.

We have guaranteed that $T_k^* \geq T_k$ so the EAC property of σ means that any O-move $m_{2k+1}(T_k^* + 1)$ will be responded to by the R_k^{th} child of the O-move before m_{2k+1}, say $\vec{u}R_k$. Further, any subsequent O-move $\vec{u}R_k l$ will be responded to by the l^{th} child of the move before $\vec{u}$.

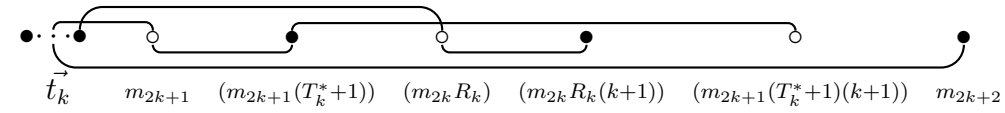
Define a sequence of innocent functions of compact O-strategies $\langle f_{\tau_0}, \ldots, f_{\tau_n} \rangle$ and a sequence of legal positions $\langle \vec{t}_0, \ldots, \vec{t}_n \rangle$ inductively by

$$f_{\tau_0} = \emptyset,\ \vec{t}_0 = \langle \varepsilon \rangle,$$

$f_{\tau_{k+1}} = f_{\tau_k} \cup$



$\vec{t}_{k+1} =$



Where the moves m_{2k+1} and m_{2k+2} are justified by the moves as specified in the original sequence $\vec{s}$. Finally we set $f_\tau = f_{\tau_{2n}}$ and $\vec{t} = \vec{t}_{2n}$.

We show by induction on k that

(i) The extra clauses for the innocent function of τ_{k+1} are consistent with those of τ_k.

(ii) $\vec{s}_{2k} \preceq \vec{t}_k$ and $\ulcorner \vec{s}_{2k} \urcorner = \ulcorner \vec{t}_k \urcorner$.

(iii) $\vec{t}_k \in \sigma \cap \tau_k$.

For (i) we look at the 3 extra clauses. We can simplify the problem by counting the lengths of the sequences involved. Certainly $|\vec{t}_k| = 1 + 6k$. Also, in $\vec{t}_{k+1}$ the move m_{2k+1} must be justified by the move m_{2j} for some $j \leq k$. Hence taking the O-view of $\vec{t}_{k+1}$ cuts out all the moves $m_{2j}R_j$ and $m_{2j}R_j(j+1)$ for $j \leq k$ and some

number of complete sets of the six moves added at each stage of the process. So $|\lfloor \vec{t}_k \rfloor| = 1 + 4p$ for some p.

Similarly considering justification pointers we can see that $\lfloor \vec{t}_k \cdot m_{2k+1} \rfloor = \lfloor \vec{t}_j \rfloor \cdot m_{2k+1}$ for some $j \leq k$. Thus $|\lfloor \vec{t}_k \cdot m_{2k+1} \rfloor| = 2 + 4q$ for some q. Hence the sequences mapped in the first two clauses have length 2 (mod 4), and in the last clause have length 0 (mod 4). So we can never be adding inconsistent clauses by conflicting between the first and third or second and third clause.

There are a number of remaining possibilities to cover:

1. It might be the case that $\lfloor \vec{t}_k \cdot m_{2k+1} \rfloor = \lfloor \vec{t}_l \cdot m_{2l+1} \rfloor$ for some $l < k$ but since in that case we must have $m_{2k+1} = m_{2l+1}$ so $T_l^* = T_k^*$ and so the extra clause is still consistent.

2. If the first clause conflicts with a second clause from some τ_l then we have $\lfloor \vec{t}_k \cdot m_{2k+1} \rfloor = \lfloor \vec{t}_{\lfloor} \cdot m_{2l} R_l$. So we must have $l < k$.

   But then $m_{2k+1} = m_{2l} R_l$ and $m_{2k} = m_{2l}$, hence $f_\sigma(\vec{q_k}) = (R_l, p)$ for some p. But $R_l \geq F + 1 - O_l > R_l$ by the definition of F, hence a contradiction.

3. If the second clause conflicts with a second clause from a different τ_l we have $\lfloor \vec{t}_{k\rfloor} \cdot (m_{2k} R_k) = \lfloor \vec{t}_{l\rfloor} \cdot (m_{2l} R_l)$. But these O-views end in an O-move so we can see the penultimate moves, which are $m_{2k+1}(T_k^* + 1)(k+1)$ and $m_{2l+1}(T_l^* + 1)(l+1)$. Hence $k = l$.

4. If the third clause conflicts with a third clause from a different τ_l we have, similarly to above, $m_{2k+1}(T_k^* + 1)(k+1) = m_{2l+1}(T_l^* + 1)(l+1)$ and hence $k = l$.

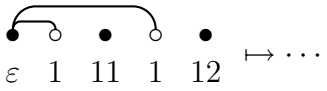
(ii) is trivial when we examine the sequence $\vec{t}_{k+1}$.

(iii) is also easy to see. The O-moves all come directly from the new clauses in τ_{k+1}. The first P-move is from the strategy σ (it is guaranteed to occur since $\ulcorner s \urcorner \in \sigma$). The other two P-moves are responses to moves beyond the copycat threshold and so are automatic. ■

We remark that the number F is required to avoid inconsistencies between the first clause of f_{τ_k} and the second of f_{τ_l} for some $l < k$, but that this problem can only occur in special circumstances. This method of finding a suitable F is rather like using a sledgehammer to crack a nut; a more economical value could be found, at the expense of a more complicated proof.

Since the proof of the Separation Lemma looks a bit of a mess at first sight, we give an example.

Example 3.6.2 Suppose we are given an EAC strategy σ, of which one clause of the innocent function is something like:



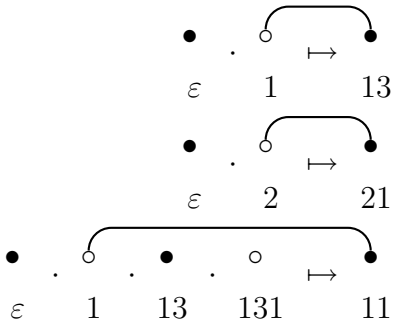
and that $t_\varepsilon = 1, t_1 = 2, o_\varepsilon = o_1 = 0$.

We are trying to construct an innocent O-strategy τ such that the play of $\sigma \cap \tau$ contains as a P-view the left-hand side of the above clause. This looks tricky, because the moves 11 and 12 seem to have been played after identical O-view, which would be impossible for an innocent O-strategy. The solution is that large amounts of the play might be hidden by taking the P-view.

Applying the above technique gives that $F = 1$ and $T_0^* = T_1^* = 2$ hence,

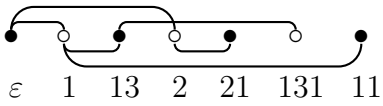
$$f_{\tau_0} = \{\,\}$$
$$\vec{t_0} = \langle \varepsilon \rangle$$

$f_{\tau_1} = f_{\tau_0} \cup \{$

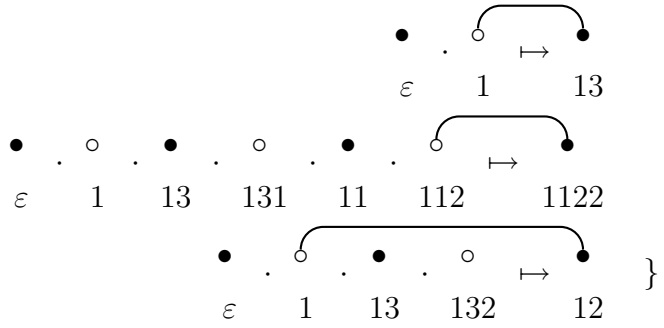


$\}$

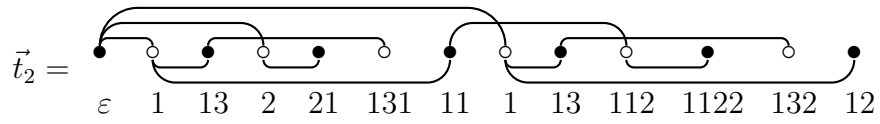
$\vec{t_1} =$



$f_{\tau_2} = f_{\tau_1} \cup \{$

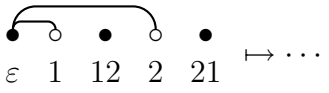


$\}$



And it is clear that the P-view of $\vec{t}_2$ is as required.

That example showed why we must use T^* and not just T, else the first clause of f_{τ_1} would have been inconsistent with that of f_{τ_2}. To show why the F variable is necessary, the reader is invited to apply the same method to the sequence



and any strategy containing that sequence, with $t_\varepsilon = 1, t_2 = 0, o_\varepsilon = o_2 = 0$.

To end this section we remark that the condition in the lemma that the sequence $\vec{s}$ must be odd may be relaxed — in order to construct an even-length P-view we proceed as if constructing (any) sequence of longer length and just stop after the adding the first move of the last sequence $\vec{t}_n$.

3.7 Consequences of the Separation Lemma

Using the Separation Lemma we will be able to prove a variety of results about the extensionality properties of $\mathcal{D}_{\mathrm{EAC}}$. We first need a few lemmas.

Lemma 3.7.1 If σ and τ are innocent strategies on U, considered as subsets of legal positions of U, then the application $\sigma \bullet \tau$ is the strategy

$$\{\vec{s} \restriction U_2 \mid \vec{s} \in \sigma \ \wedge \ \vec{s} \restriction (U_1, u)^+ \in \tau^* \text{ for any initial } U_1\text{-move } u \text{ in } \vec{s}\}$$

where the arena U is decomposed as $U_1 \Rightarrow U_2$, and τ^* is the innocent O-strategy $\{\varepsilon\} \cup \{1 \cdot \vec{s} \mid \vec{s} \in \tau\}$. (Recall the definitions of $\vec{s} \restriction B$ and $\vec{s} \restriction (A, a)^+$ from Section 2.1.)

The proof of this involves a fairly detailed examination of the interaction sequences of $\langle \sigma \, ; \mathit{Fun}, \tau \rangle \, ; \mathrm{eval}_{U,U}$ but it is straightforward and we omit it.

Corollary 3.7.2 Application is monotonic in both arguments. That is, $\sigma_1 \subseteq \sigma_2$ implies $\sigma_1 \bullet \tau \subseteq \sigma_2 \bullet \tau$ and $\tau \bullet \sigma_1 \subseteq \tau \bullet \sigma_2$.

Lemma 3.7.3 Decompose the arena U as $A \Rightarrow B$ where $A = U^n$ and $B = U$. Then given a compact innocent O-strategy τ on U, of which the non-initial O-moves are contained in the A-component of U, there exists a sequence of compact innocent P-strategies $\langle \rho_1, \ldots, \rho_n \rangle$ such that for any P-strategy σ on U,

$$\{\vec{s} \upharpoonright B \mid \vec{s} \in \sigma \ \wedge \ \vec{s} \upharpoonright (A, a)^+ \in \tau \text{ for any initial } A\text{-move } a \text{ in } \vec{s}\} = (\sigma \bullet \rho_1) \cdots \bullet \rho_n.$$

Proof The proof is by induction on n, for all strategies τ simultaneously.

For the base case we take $n = 0$. Although not part of the lemma, we extend to the case $n = 0$, which means the following: decompose the arena U as $A \Rightarrow B$ with $A = \mathbf{1}$ and $B = U$. Then for any compact innocent O-strategy τ, of which the non-initial O-moves are contained in A, we have $\{\vec{s} \upharpoonright B \mid \vec{s} \in \sigma \ \wedge \ \vec{s} \upharpoonright (A, a)^+ \in \tau \text{ for any initial } A\text{-move } a \text{ in } \vec{s}\} = \sigma$. But there are no initial A-moves, since it is the arena E with no trees at all, so the equations just says that $\{\vec{s} \upharpoonright U \mid \vec{s} \in \sigma\} = \sigma$, which is clearly true.

For the inductive step decompose the arena U into $(A \times C) \Rightarrow B$, with $A = U^n$, $B = C = U$. Suppose that τ is a compact innocent O-strategy on U, of which the non-initial O-moves are contained in the $(A \times C)$-component of U.

Set $\tau' = \{\varepsilon\} \cup \{i \cdot \vec{s} \mid i \leq n, i \cdot \vec{s} \in \tau\}$ and $\tau'' = \{\varepsilon\} \cup \{i \cdot \vec{s} \mid i = n + 1, i \cdot \vec{s} \in \tau\}$. Clearly τ' and τ'' are compact innocent O-strategies, and by hypothesis we know that $\tau = \tau' \cup \tau''$.

Applying the inductive hypothesis to τ', with the arena U decomposed as $A \Rightarrow (C \Rightarrow B)$, we have that there exists a sequence of innocent P-strategies $\langle \rho_1, \ldots, \rho_n \rangle$ such that for any P-strategy σ on U,

$$\begin{array}{rcl} S & = & \{\vec{s} \upharpoonright (C \Rightarrow B) \mid \vec{s} \in \sigma \ \wedge \ \vec{s} \upharpoonright (A, a)^+ \in \tau' \text{ for any initial } A\text{-move } a \text{ in } \vec{s}\} \\ & = & (\sigma \bullet \rho_1) \cdots \bullet \rho_n. \end{array}$$

Set $\rho_{n+1} = \{\vec{s} \mid (n+1)\vec{s} \in \tau''\}$, it is easy to see that ρ_{n+1} is a compact innocent P-strategy. Then by Lemma 3.7.1 (decomposing U as $C \Rightarrow B$) and the above equation we know that

$$\begin{array}{rcl} & & ((\sigma \bullet \rho_1) \cdots \bullet \rho_n) \bullet \rho_{n+1} \\ & = & \{\ \vec{t} \upharpoonright B \mid \vec{t} \in S \ \wedge \ \vec{t} \upharpoonright (C, c)^+ \in \rho_{n+1}^* \text{ for any initial } C\text{-move } c \text{ in } \vec{t}\ \} \\ & = & \{\ (\vec{s} \upharpoonright (C \Rightarrow B)) \upharpoonright B \ \mid \ \vec{s} \in \sigma \\ & & \ \wedge \ \vec{s} \upharpoonright (A, a)^+ \in \tau' \text{ for any initial } A\text{-move } a \text{ in } \vec{s} \\ & & \ \wedge \ (\vec{s} \upharpoonright (C \Rightarrow B)) \upharpoonright (C, c)^+ \in \rho_{n+1}^* \text{ for any initial } C\text{-move } c \text{ in } \vec{s} \upharpoonright (C \Rightarrow B)\ \} \end{array}$$

But $(\vec{s} \upharpoonright (C \Rightarrow B)) \upharpoonright (C, c)^+ \in \rho_{n+1}^*$ is equivalent to $\vec{s} \upharpoonright (C, c)^+ \in \tau''$, by the definition of ρ_{n+1}. Also clearly $(\vec{s} \upharpoonright (C \Rightarrow B)) \upharpoonright B = \vec{s} \upharpoonright B$ so if we write $D = A \times C$ then

$$\begin{array}{l} ((\sigma \bullet \rho_1) \cdots \bullet \rho_n) \bullet \rho_{n+1} = \\ \qquad \{\vec{s} \upharpoonright B \mid \vec{s} \in \sigma \ \wedge \ \vec{s} \upharpoonright (D, d)^+ \in \tau \text{ for any initial } D\text{-move } d \text{ in } \vec{s}\} \end{array}$$

which establishes the inductive step. ■

Note that in the above induction we defined ρ_i for $i = 1, \ldots, n$ by $\rho_i = \{\vec{s} \mid i{\cdot}\vec{s} \in \tau\}$. Thus ρ_i is precisely the P-strategy corresponding to the moves of τ in the i^{th} subtree of U.

Lemma 3.7.4 If σ_1 and σ_2 are EAC innocent strategies on U satisfying $\sigma_1 \not\subseteq \sigma_2$ then there is a compact innocent strategy ρ on U satisfying

$$\sigma_1 \bullet \rho \not\subseteq \sigma_2 \bullet \rho.$$

Proof Let $\vec{s}$ be some legal position in $\sigma_1 \setminus \sigma_2$; we can choose $\vec{s}$ to end in a P-move. Suppose the first P-move of $\vec{s}$ is m and that the copycat threshold and offset of σ_1 at ε are t_ε and o_ε respectively. Choose a large natural number n bigger than $t_\varepsilon - o_\varepsilon$. Define

$$\vec{s_1} = \underbrace{\varepsilon \quad m \quad \cdots \quad -}_{\vec{s}} \quad m(n + o_\varepsilon + 1)$$

Then $\vec{s_1} \in \sigma_1 \setminus \sigma_2$.

We apply the Separation Lemma to the sequence $\vec{s_1}$. This generates a compact O-strategy τ and a legal position $\vec{t}$ such that $\vec{t} \in \sigma_1 \cap \tau$, $\vec{s_1} \preceq \vec{t}$, and $\ulcorner \vec{s_1} \urcorner = \ulcorner \vec{t} \urcorner$. This ensures that $\vec{t} \notin \sigma_2$.

Write

$$\vec{s_2} = \underbrace{\varepsilon \quad m \quad \cdots \quad m(n + o_\varepsilon + 1)}_{\vec{t}} \quad n + 1 \qquad \text{and } \vec{s_3} = \varepsilon \quad 1$$

We can be sure that $\vec{s_2} \in \sigma_1 \cap \tau$ because the behaviour of σ_1 in response to the move $m(n + o_\varepsilon + 1)$ is forced by the properties of EAC, since $n + o_\varepsilon + 1 > t_\varepsilon$, and contingent completeness of τ. Also $\vec{t} \leq \vec{s_2}$ so $\vec{s_2} \notin \sigma_2 \cap \tau$.

Now we note that the value of n has no effect on the O-strategy τ or the legal position $\vec{t}$ until the very last stage of the Separation Lemma. Hence we can choose the value of n to be large enough to be certain that

(i) All of the non-initial O-moves of τ are contained in $A = \{i{\cdot}\vec{s} \mid i \leq n \wedge \vec{s} \in \mathbb{N}^*\}$,

(ii) The same applies to the non-initial O-moves of s_2.

Now apply Lemma 3.7.3 to the O-strategy τ. This generates a sequence of compact innocent strategies $\langle \rho_1, \ldots, \rho_n \rangle$ with the property that $\vec{s_3} \in (\sigma_1 \bullet \rho_1) \cdots \bullet \rho_n$ (from the facts shown above, and because decomposing U into $A \Rightarrow B$ with $A = U^n$ gives that $\vec{s_2} \upharpoonright B = \vec{s_3}$), and also that $\vec{s_3} \notin (\sigma_2 \bullet \rho_1) \cdots \bullet \rho_n$. So by Corollary 3.7.2, we can be certain that $\sigma_1 \bullet \rho_1 \not\subseteq \sigma_2 \bullet \rho_1$. ■

Lemma 3.7.5

(i) Given a compact P-strategy τ on U one can generate an EAC P-strategy τ' on U such that $\tau' \supseteq \tau$.

(ii) Furthermore, having constructed τ' as above and given a legal position $\vec{s} \in \tau' \setminus \tau$ there is an EAC P-strategy τ'' such that $\tau' \supseteq \tau'' \supseteq \tau$ with $\vec{s} \notin \tau''$.

Proof Suppose that the economical form of the innocent function of τ is f, and by assumption this has a finite domain.

For (i) we will construct a function f', the economical form of τ'. We first define a set $X \subset \mathbb{N}^*$ which codes a set of P-views at which f' will be given explicitly (see the proof of Lemma 3.5.2).

Let $\vec{v}$ be any sequence on which f is defined. If f is defined at $\vec{v} \cdot i$ for some i then set $t_v = \max\{i \mid \vec{v} \cdot i \in \mathrm{dom}(f)\}$, else take $t_v = 0$. Also set $b = \max\{i \mid \exists \vec{w} \geq \vec{v}\,.\, f(\vec{w}) = (i, |\vec{w}| - |\vec{v}|)\}$ and $o_v = t_v - b$. These maxima must exist, and are specified by a computable function, because the domain of f is finite.

Define the set X inductively by

$$\begin{array}{l} \varepsilon \in X \\ \text{if } \vec{v} \in X \text{ then } 1 \leq i \leq t_v \implies \vec{v} \cdot i \in X. \end{array}$$

We define the economical form of the innocent function f' of τ' as follows. For any $\vec{v} \in X$ set

$$\begin{array}{rcll} f'(\vec{v}) & = & f(\vec{v}) & \\ f'(\vec{v} \cdot i) & = & (i - o_v, 1) & \text{for } i > t_v, \\ f'(\vec{v} \cdot i \cdot \vec{s} \cdot j) & = & (j, 1) & \text{for } i > t_v \text{ and any sequence } \vec{s}. \end{array}$$

We also specify that $t_v = o_v = 0$ for any $\vec{v} \in \mathrm{dom}(f') \setminus X$.

It is clear that the domain of f' is prefix-closed and satisfies the other conditions for it to be the economical form of an innocent function, and also that $f'(\vec{v}) = f(\vec{v})$ for all $\vec{v} \in \mathrm{dom}(f)$. It is routine to check that f' is EAC with copycat thresholds and offsets specified by $\vec{v} \mapsto t_v$ and $\vec{v} \mapsto t_o$.

For (ii), write $\vec{s} = \vec{t_1} \cdot m \cdot \vec{t_2}$ with $\vec{t_1} \in \tau$ but $\vec{t_1} \cdot m \notin \tau$ (the move m must be a P-move by contingent completeness of τ). Since $\vec{t_1} \in \tau$, $\ulcorner \vec{t_1} \urcorner$ is coded by some sequence $\vec{b} = \langle b_1, \ldots, b_n \rangle$, and $\vec{b} \in \mathrm{dom}(f') \setminus \mathrm{dom}(f)$ (since $\tau \subseteq \tau'$). Set $\vec{b'} = \langle b_1, \ldots, b_{n-1} \rangle$.

Now proceed to construct f'' exactly as f' was, but modify $t_{b'}$ to be b_n (do not change $o_{b'}$) before constructing the set X, and continue to the end of (i) above. We must have increased the value of $t_{b'}$ in this way, otherwise it could not have been the case that the construction of τ' added the sequence $\vec{s}$.

It is easy to see that the only change being an increased value of $t_{b'}$ means that $f'' \subseteq f'$, and nothing changed the fact that $f \subset f''$ as before. Since $\vec{b} \notin \mathrm{dom}(f)$ and with the new value of $t_{b'}$ we have do not have $\vec{b} = \vec{b'} \cdot i$ for $i > t_{b'}$, we have not assigned a value for $f''(\vec{b})$. Hence $\vec{t_1} \cdot m \notin \tau''$, so $\vec{s} \notin \tau''$. ■

We can now prove the main result of this section.

Theorem 3.7.6 $\mathcal{D}_{\mathrm{EAC}}$ is *order-extensional*. That is for $\sigma_1, \sigma_2 \in \mathcal{D}_{\mathrm{EAC}}$ we have

$$(\forall \tau \in \mathcal{D}_{\mathrm{EAC}}.\ \sigma_1 \bullet \tau \subseteq \sigma_2 \bullet \tau) \implies \sigma_1 \subseteq \sigma_2.$$

Proof Suppose that $\sigma_1 \not\subseteq \sigma_2$. We can then apply Lemma 3.7.4 give a compact innocent strategy ρ with $\sigma_1 \bullet \rho \not\subseteq \sigma_2 \bullet \rho$.

Decompose the arena U as $A \Rightarrow B$ with $A = B = U$ and let $\vec{s}$ be a legal position of U in $(\sigma_1 \bullet \rho) \setminus (\sigma_2 \bullet \rho)$.

This means that $\vec{s} = \vec{t} \upharpoonright B$ for some legal position $\vec{t}$ such that

$$\begin{array}{l} \vec{t} \in \sigma_1 \ \wedge\ \vec{t} \upharpoonright (A, a)^+ \in \rho^* \text{ for any initial } A\text{-move } a \text{ in } \vec{t} \\ \vec{t} \notin \sigma_2 \text{ (else by the above we would have } \vec{s} \in \sigma_2 \bullet \rho). \end{array}$$

Now we will apply Lemma 3.7.5 to the compact innocent strategy ρ to get an EAC strategy $\tau \supseteq \rho$. This will ensure that $\vec{s} \in \sigma_1 \bullet \tau$. We want to show that $\vec{s} \notin \sigma_2 \bullet \tau$. There may be a problem if there is a legal position $\vec{t'}$ such that

$$\begin{array}{l} \vec{t'} \upharpoonright B = \vec{s} \\ \vec{t'} \in \sigma_2 \\ \vec{t'} \upharpoonright (A, a)^+ \in \tau^* \text{ for any initial } A\text{-move } a \text{ in } \vec{t} \\ \vec{t'} \upharpoonright (A, a)^+ \notin \rho^* \text{ for some initial } A\text{-move } a \text{ in } \vec{u}. \end{array}$$

Then we would have $\vec{s} \in \sigma_2 \bullet \tau$ even though $\vec{s} \notin \sigma_2 \bullet \rho$.

If there is such a $\vec{t'}$ it must be unique for the following reasons: its moves in the arena B are specified by $\vec{s}$, its P-moves in the arena A are forced by σ_2 and its O-moves in the arena A are forced by τ^*.

Define $\vec{u} = \vec{t'} \upharpoonright (A, a)$ for the first initial A-move a occurring in $\vec{t'}$ such that $\vec{t'} \upharpoonright (A, a)^+ \in (\tau^* \setminus \rho^*)$. Then $\vec{u} \in (\tau \setminus \rho)$.

So by (ii) of Lemma 3.7.5 there is an EAC strategy τ' such that $\rho \subseteq \tau'$ — hence $\vec{s} \in \sigma_1 \bullet \tau'$ — and $\tau' \subseteq \tau$ which together with $\vec{u} \notin \tau'$ means that $\vec{t'} \upharpoonright (A, a)^* \notin (\tau')^*$ and so $\vec{s} \notin \sigma_2 \bullet \tau'$.

Hence $\sigma_1 \bullet \tau \not\subseteq \sigma_2 \bullet \tau$. ■

Corollary 3.7.7 $\mathcal{D}_{\mathrm{EAC}}$ is extensional, and hence weakly extensional. Thus $\mathcal{D}_{\mathrm{EAC}}$ is a $\lambda\eta$-model.

Remark 3.7.8 We emphasise that we have chosen to make a semantic proof of this result even though much shorter proofs exist. We did so for two reasons: the introduction of the Separation Lemma is of interest, and we will see in the next section that we can extend the results obtained to give a semantic proof of a classical syntactic result of the λ-calculus.

In fact the universality of $\mathcal{D}_{\mathrm{EAC}}$ allows us to deduce many of its properties from well-known facts about the syntactic theory of the λ-calculus, or properties of other models. As an example of the latter, the fact that Scott's model D_∞ satisfies the *ω-rule*:

$$(\forall u \in \Lambda.\ D_\infty \models su = tu) \implies D_\infty \models s = t$$

directly implies that $\mathcal{D}_{\mathrm{EAC}}$ is extensional, given the universality property. (The original proof that D_∞ satisfies the ω-rule is due to Wadsworth [Wad76] and Nakajima [Nak75].)

For completeness we also give an alternative, syntactic, method based on techniques developed by Barendregt.

Alternative proof of Theorem 3.7.6 Let σ_1, σ_2 be elements of $\mathcal{D}_{\mathrm{EAC}}$. Then, by universality, $\sigma_1 = [\![s]\!]$ and $\sigma_2 = [\![t]\!]$, for some closed terms s and t. Assume that $\sigma_1 \not\subseteq \sigma_2$. By the Exact Correspondence Theorem, $\mathrm{NT}(s) \not\subseteq \mathrm{NT}(t)$, 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 [Bar84, 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 [Bar84, 10.2.21] for the definition of $\sim_\alpha$ and [Bar84, 10.2.31] for the proof). Then by [Bar84, 10.3.13], [Bar84, 10.4.1(ii)] and [Bar84, 10.3.4], there is a sequence of terms $u_1, \ldots, u_n$ such that $\lambda \vdash su_1 \ldots u_n = I$ and $tu_1 \ldots u_n$ is unsolvable. Hence

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

But $\bullet$ is monotone so $\sigma_1 \bullet [\![u_1]\!] \not\subseteq \sigma_2 \bullet [\![u_1]\!]$. ■

3.8 Böhm's Theorem

With a little more work we can give new proofs of some familiar results, and although results about the model $\mathcal{D}_{\mathrm{EAC}}$ are only relevant to the λ-theory $\mathcal{H}^*$ a little syntactic analysis allows us to extend the results to λ. First we make a generalisation to the Separation Lemma:

Lemma 3.8.1 Given EAC innocent P-strategies σ_1 and σ_2 on U, and even legal positions $\vec{s_1} \in \sigma_1$ and $\vec{s_2} \in \sigma_2$, with $\vec{s_1}$ and $\vec{s_2}$ of equal length, and having the same moves except for the last, there exists a compact innocent O-strategy τ and legal positions $\vec{t_1}$ and $\vec{t_2}$ such that $\vec{t_1} \in \sigma_1 \cap \tau$ and $\vec{t_2} \in \sigma_2 \cap \tau$ with $\vec{s_1} \preceq \vec{t_1}$, $\vec{s_2} \preceq \vec{t_2}$, $\ulcorner \vec{s_1} \urcorner = \ulcorner \vec{t_1} \urcorner$ and $\ulcorner \vec{s_2} \urcorner = \ulcorner \vec{t_2} \urcorner$.

Proof This is just a modification of the Separation Lemma and we do not include

the full proof. One proceeds by constructing numbers F^1, $(T_k^*)^1$ for the first sequence/strategy pair, and F^2, $(T_k^*)^2$ for the second, then set $F = \max(F_1, F_2)$ and $T_k^* = \max((T_k^*)^1, (T_k^*)^2)$ before proceeding with the rest of the construction, once for each sequence/strategy pair, making compact innocent O-strategies τ^1 and τ^2. It is routine to verify that $\tau = \tau^1 \cup \tau^2$ is a strategy satisfying the stated conditions. ■

Lemma 3.8.2 If σ_1 an σ_2 are EAC strategies on U such that there is some legal position $\vec{s} \in \sigma_1 \cap \sigma_2$ and justified P-moves m_1 and m_2 such that $\vec{s} \cdot m_1 \in \sigma_1 \setminus \sigma_2$ and $\vec{s} \cdot m_2 \in \sigma_2 \setminus \sigma_1$ then there is a sequence of compact innocent strategies $\rho_1, \ldots, \rho_n$ such that

$$(\sigma_1 \bullet \rho_1) \cdots \bullet \rho_n = [\![\lambda xy.x]\!] \qquad (\sigma_2 \bullet \rho_1) \cdots \bullet \rho_n = [\![\lambda xy.y]\!].$$

Proof We only sketch the proof, which is by analogy with Lemma 3.7.4.

Suppose the first P-move of $\vec{s}$ is m and that the copycat threshold and offset of σ_i at ε are t_ε^i and o_ε^i respectively (for $i = 1, 2$). Choose a large natural number n bigger than $t_\varepsilon^1 - o_\varepsilon^1$ and $t_\varepsilon^2 - o_\varepsilon^2$. Define

$$\vec{s_1} = \underbrace{\varepsilon \quad m \quad \cdots}_{\vec{s}} \quad m_1 \quad m(n + o_\varepsilon^1 + 1) \quad n+1$$

$$\vec{s_2} = \underbrace{\varepsilon \quad m \quad \cdots}_{\vec{s}} \quad m_2 \quad m(n + o_\varepsilon^1 + 2) \quad n+2$$

We apply the generalised version of the Separation Lemma above to the strategies with these sequences, to generate a compact innocent O-strategy τ together with legal positions $\vec{t_1} \in \sigma_1 \cap \tau$ and $\vec{t_2} \in \sigma_2 \cap \tau$, and

$$\ulcorner \vec{t_1} \urcorner = \varepsilon \quad m \quad m(n + o_\varepsilon^1 + 1) \quad n+1 \qquad \text{and} \qquad \ulcorner \vec{t_2} \urcorner = \varepsilon \quad m \quad m(n + o_\varepsilon^1 + 2) \quad n+2$$

Set $\vec{u_1} = \varepsilon \quad 1$ and $\vec{u_2} = \varepsilon \quad 2$.

As before, we can choose n sufficiently large to make all the non-initial O-moves of τ (and the moves in $\vec{t_1}$ and $\vec{t_2}$) occur in $\{i \cdot \vec{s} \mid i \leq n \ \wedge \ \vec{s} \in \mathbb{N}^*\}$.

Applying Lemma 3.7.3 to the O-strategy τ generates a sequence of compact innocent strategies $\langle \rho_1, \ldots, \rho_n \rangle$ with the property that $\vec{u_i} \in (\sigma_1 \bullet \rho_1) \bullet \cdots \bullet \rho_n$. Furthermore, since σ_i is AC at the initial P-view, and we exceed the copycat threshold

with the move $m(n + o^i_\varepsilon + i)$ we know that σ_i plays copycat after the last move of $\vec{s_i}$, hence $(\sigma_i \bullet \rho_1) \cdots \bullet \rho_n$ plays copycat immediately after the first move (all this for $i = 1, 2$). This implies the desired result. ■

We can also convert the compact innocent strategies ρ_i into EAC strategies τ_i, using Lemma 3.7.5 and the same techniques as before. The following theorem gives the consequence of this, and summarises the things we can prove about the λ-theory $\mathcal{H}^*$ using these results about the model. It amounts to a characterisation of separability in the theory $\mathcal{H}^*$.

For notational convenience, given a sequence $u_1, \ldots, u_n$ and a term s we write $s\vec{u}$ for the term $su_1u_2 \ldots u_n$.

Lemma 3.8.3 Let s and t be closed terms. Consider equality in the theory $\mathcal{H}^*$, with ordering given by inclusion of Nakajima Trees. Write $\bot$ for the least element (the unsolvables).

(i) $s = t$ iff for all sequences of closed terms $u_1, \ldots, u_n$, $s\vec{u} = t\vec{u}$.

(ii) $s \not\sqsubseteq t$ iff there exists a sequence of closed terms $u_1, \ldots, u_n$ such that $s\vec{u} = \lambda x.x$ and $t\vec{u} = \bot$.

(iii) s and t are inconsistent iff there exists a sequence of closed terms $u_1, \ldots, u_n$ such that $s\vec{u} = \lambda xy.x$ and $t\vec{u} = \lambda xy.y$.

The first reflects the extensionality of $\mathcal{D}_{\mathrm{EAC}}$, the second is a consequence of the lemma which is needed to prove order-extensionality, and the last comes from the generalisation Lemma 3.8.2.

We now try to lift the results of this theorem to the theory λ. (i) does not seem to have a sensible analogue, and it is certainly not true as it stands (in λ, $\lambda xy.xy \neq \lambda x.x$ but for all closed terms t, $(\lambda xy.xy)t = (\lambda x.x)t = t$). For (ii) and (iii) we have more success.

Theorem 3.8.4 Let s and t be closed terms. Give Nakajima trees the ordering of inclusion (modulo renaming of bound variables, this amounts to inclusion of variable-free form).

(i) $\mathrm{NT}(s) \not\sqsubseteq \mathrm{NT}(t)$ iff there exists a sequence of closed terms $u_1, \ldots, u_n$ such that $\lambda \vdash s\vec{u} = \lambda x.x$ and $t\vec{u}$ unsolvable.

(ii) $\mathrm{NT}(s)$ and $\mathrm{NT}(t)$ are inconsistent iff there exists a sequence of closed terms $u_1, \ldots, u_n$ such that $\lambda \vdash s\vec{u} = \lambda xy.x$ and $\lambda \vdash t\vec{u} = \lambda xy.y$.

Proof (i, $\Longleftarrow$): If there is a sequence $\vec{u}$ such that $\lambda \vdash s\vec{u} = \lambda x.x$ and $t\vec{u}$ unsolvable

then certainly $\mathrm{NT}(s\vec{u}) = \mathrm{NT}(\lambda x.x)$ and $\mathrm{NT}(t\vec{u}) = \bot$, so $\mathcal{H}^* \vdash s\vec{u} = \lambda x.x$ and $\mathcal{H}^* \vdash t\vec{u} = \bot$ (we also write $\bot$ for some unsolvable term). Hence, by Lemma 3.8.3, $\mathrm{NT}(s) \not\subseteq \mathrm{NT}(t)$.

(i, $\Longrightarrow$): The hypothesis allows us to apply Lemma 3.8.3 to show that there is a sequence of terms $\vec{u}$ such that $\mathcal{H}^* \vdash s\vec{u} = \lambda x.x$ and $\mathcal{H}^* \vdash t\vec{u} = \bot$. Hence $\mathrm{NT}(s\vec{u}) = \mathrm{NT}(\lambda x.x) \neq \bot$, so $s\vec{u}$ is solvable. Hence by the definition of solvability there is a sequence of terms $\vec{v}$ such that $\lambda \vdash s\vec{u}\vec{v} = \lambda x.x$. Also $t\vec{u}$ must be unsolvable, so likewise $t\vec{u}\vec{v}$.

(ii, $\Longleftarrow$): If there is a sequence $\vec{u}$ such that $\lambda \vdash s\vec{u} = \lambda xy.x$ and $\lambda \vdash t\vec{u} = \lambda xy.y$ then certainly $\mathrm{NT}(s\vec{u}) = \mathrm{NT}(\lambda xy.x)$ and $\mathrm{NT}(t\vec{u}) = \mathrm{NT}(\lambda xy.y)$, so $\mathcal{H}^* \vdash s\vec{u} = \lambda xy.x$ and $\mathcal{H}^* \vdash \lambda xy.y$ Hence, by Lemma 3.8.3, $\mathrm{NT}(s)$ and $\mathrm{NT}(t)$ are inconsistent.

(ii, $\Longrightarrow$): The hypothesis allows us to apply Lemma 3.8.3 to show that there is a sequence of terms $\vec{u}$ such that $\mathcal{H}^* \vdash s\vec{u} = \lambda xy.x$ and $\mathcal{H}^* \vdash t\vec{u} = \lambda xy.y$. Hence $\mathrm{NT}(s\vec{u}) = \mathrm{NT}(\lambda xy.x)$ and $\mathrm{NT}(t\vec{u}) = \mathrm{NT}(\lambda xy.y)$, so by the definition of Nakajima tree both $s\vec{u}$ and $t\vec{u}$ are solvable with $\lambda \vdash s\vec{u} = \lambda xyz_1 \ldots z_n.xs_1 \ldots s_m$ for some n, m and terms s_i, and $\lambda \vdash t\vec{u} = \lambda xyz_1 \ldots z_{n'}.yt_1 \ldots t_{m'}$ for some n', m' and terms t_i.

Suppose $n' \geq n$ (the other case is symmetrical). Set $X = \lambda a_1 \ldots a_m b_1 \ldots b_{n'-n} cd.c$ and $Y = \lambda a_1 \ldots a_{m'} cd.d$. It is easy to check the following:

$$\lambda \vdash s\vec{u}XYr_1 \ldots r_{n'} = \lambda xy.x$$
$$\lambda \vdash t\vec{u}XYr_1 \ldots r_{n'} = \lambda xy.y$$

where $r_1 \ldots r_{n'}$ are any terms. ■

(ii) of the above Theorem is a generalisation of *Böhm's Theorem*, which is usually stated as follows: "If s and t are different closed terms in $\beta\eta$-normal form then there is a sequence of closed terms $\vec{u}$ such that $\lambda \vdash s\vec{u} = \lambda xy.x$ and $\lambda \vdash t\vec{u} = \lambda xy.y$."

The version we have proved has the weakest possible precondition for such a sequence of terms to exist, namely $\mathrm{NT}(s)$ and $\mathrm{NT}(t)$ are inconsistent (equivalently, there is a node where both $\mathrm{NT}(s)$ and $\mathrm{NT}(t)$ are labelled, and the labels are different), because we have shown the implication both ways.

Chapter 4

Explicit EAC Strategies and the Model $\mathcal{D}_{\mathrm{XA}}$

Our aim in this chapter is to modify the model $\mathcal{D}_{\mathrm{EAC}}$ to find a game model which does not validate η-conversion. To do so, we will need to augment EAC strategies with some extra information. By examining the parts of Nakajima trees which use fresh variables, in the light of the Exact Correspondence Result, it becomes clear that the additional information we require is that which specifies copycat thresholds at each P-view.

We are thus lead to the definition of an *EXAC strategy*, and we describe the problems involved in the search for a CCC of such strategies, and the solution $\mathbb{XA}$. This category also has a reflexive object, but for which the retraction morphisms are not isomorphisms, and hence the model $\mathcal{D}_{\mathrm{XA}}$ which arises in it does not validate η-conversion. We formulate an analogue of the variable-free form for Böhm trees, and show that an Exact Correspondence for this model holds. This means that the local structure of $\mathcal{D}_{\mathrm{XA}}$ is the λ-theory $\mathcal{B}$. Unlike $\mathcal{D}_{\mathrm{EAC}}$, however, $\mathcal{D}_{\mathrm{XA}}$ is not extensional or even weakly extensional.

4.1 Specifying Copycat Thresholds

To find a model in which η-conversion is not validated, we require the terms I and 1 to be denoted differently. They have the same variable-free form of Nakajima tree, so it is not apparent how this might be achieved. The key is to make use of the fact that valid copycat thresholds are not unique — any number greater than a given valid copycat threshold is also a valid copycat threshold (Lemma 3.4.3). Different thresholds (at some P-view) may be used to distinguish I and 1.

This idea is prompted by the observation that when one compares a term with its denotation, the part of the EAC strategy which is specified by the rules of

copycat (the part which is not specified *explicitly* – see the proof of Lemma 3.5.2) corresponds precisely to the part of the Nakajima tree which has been generated by η-expansion (i.e. the part of the tree with the fresh variables as the head variables). Recall the Nakajima trees of I and 1 — the former has fresh variables appearing at every node except the root, whereas the latter is similar except that there is not a fresh variable at the first child of the root. Therefore we aim to find a model where I and 1 are represented by the strategy with the same moves, but the copycat threshold of $[\![I]\!]$ at the first P-view is 0, whereas that of $[\![1]\!]$ is 1.

However, the definition of an EAC strategy is stated in terms of the existence of some computable function which associates a pair of numbers to each P-view of the strategy and this function is *not* specified along with the strategy. (A consequence of this is that there is no computable procedure for finding valid thresholds for an EAC strategy, nor for deciding whether a given innocent strategy is EAC.)

Remark 4.1.1 It is really the thresholds (rather than the offsets) which are important because, for a certain P-view $\vec{v}$ of an EAC strategy σ, the copycat threshold t usually gives enough information to compute the offset o directly. This is clear since $f_\sigma(\vec{v} \cdot (t+1)) = (t+1-o, 1)$, as long as the move coded by $\vec{v} \cdot (t+1)$ actually exists in the arena in question. When this move does not exist, any value would do for the offset. However, in this case the value of the offset is irrelevant and we will not take this technicality into account. Similarly, when none of the moves coded by $\vec{v} \cdot i$ exist for any i, the threshold is irrelevant. We will not distinguish between strategies which only differ in such circumstances. In practice, we only consider strategies over the arena U, in which all such moves always exist, so this technicality can be ignored.

This motivates the following definition:

Definition An *effectively and explicitly almost-everywhere copycat strategy* (EXAC strategy) is given by a pair $\langle \sigma, t_\sigma \rangle$, where σ is an EAC strategy and t_σ is an effective function mapping the P-views where σ is defined to valid copycat thresholds. We sometimes write the EXAC strategy $\langle \sigma, t_\sigma \rangle$ just as σ.

We will usually refer to the first and second part of an EXAC strategy as the "(underlying) EAC strategy (part)" and the "threshold function (part)", respectively. In view of our remark above, however, we will sometimes speak of the offsets as if they too are specified by the threshold function.

This definition allows us to make the intended finer distinction between strategies: two strategies with the same moves must be equal as EAC strategies, but may have different copycat thresholds and so can be distinguished as EXAC strategies. There is an obvious forgetful map from EXAC strategies to EAC strategies, which takes only the strategy part (i.e. erasing the threshold information).

In a similar vein to the economical form of innocent strategies, using the same encoding of a P-view as a sequence of natural numbers, we can give an economical form of EXAC strategies over single-tree arenas. We can also take advantage of the fact that parts of the strategy are completely dictated by its copycat nature. Let us say that a P-view is *entirely explicit* if none of the O-moves in it exceed the copycat threshold of the P-view at which they are made. Thus if a P-view is not entirely explicit the ensuing move can be deduced from the threshold and offset of the P-view preceding the first O-move in it which did exceed the copycat threshold.

Definition The *economical form* of an EXAC strategy is a map from $\mathbb{N}^*$ to $\mathbb{N} \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{Z}$. The domain is the encoding of P-views in the same way as economical form of an EAC strategy. The map is defined at a sequence $\vec{v}$ only if the P-view encoded by $\vec{v}$ is entirely explicit, in which case

$$\vec{v} \mapsto (i, r, t, o)$$

where the resulting P-move is encoded as before — it is the i^{th} child of the move $2r$ from last of the P-view — and the copycat threshold and offset at this P-view are t and o respectively.

Example 4.1.2 We take the EXAC strategies η_0 and η_1 to be $\langle [\![I]\!], t_0 \rangle$ and $\langle [\![1]\!], t_1 \rangle$, where t_0 maps every P-view to the threshold 0 and t_1 does likewise except that the minimal P-view is mapped to the threshold 1. Since $[\![I]\!] = [\![1]\!]$, they have the same EAC strategy part, but different threshold functions. These are the suggestions we made for the denotations of I and 1 in a model not supporting η-conversion. Nearly every P-view of either is not entirely explicit, and the respective economical forms are given by:

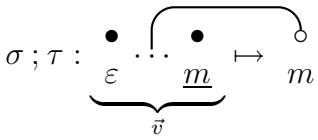
$$\begin{array}{rcl} \varepsilon & \mapsto & (1,0,0,-1) \end{array} \quad \text{and} \quad \begin{array}{rcl} \varepsilon & \mapsto & (1,0,1,-1) \\ \langle 1 \rangle & \mapsto & (2,1,0,0) \end{array}$$

4.2 Composition of EXAC Strategies

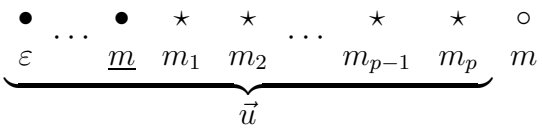
We now need a method to compose EXAC strategies. Of course the EAC strategy part will just be the standard composition of innocent strategies, and we give below an algorithm for computing the composition of the threshold functions.

Algorithm 4.2.1 (The Composition Algorithm) Let $\langle \sigma, t_\sigma \rangle$ be an EXAC strategy over $A \Rightarrow B$, and $\langle \tau, t_\tau \rangle$ be an EXAC strategy over $B \Rightarrow C$. Take a P-view $\vec{v}$ on which the strategy $\sigma ; \tau$ (which is given by the usual composition of

innocent strategies) is defined and suppose that the last move of the P-view is $\underline{m}$ and the resulting move is m:



We write $\vec{u} = \mathbf{u}(\vec{v}, \sigma, \tau)$ for the *uncovering* of the composition up up to the move m (see the definition in Section 2.2). Along with the following move m it will be of the form:



The moves m_i are the intermediate interactions which might have taken place between σ and τ before the move m became the visible outcome. They are all in the arena B and they are either O- or P-moves, depending on which strategy is looking at them (which is why they are written $\star$ as in Example 2.2.6). Possibly there are no such intermediate moves, in which case $p = 0$. We do not care about justification pointers, and for tidiness set $m_0 = \underline{m}$ and $m_{p+1} = m$.

For $1 \leq i \leq p+1$ we consider the P-view that the strategies σ or τ are faced with when making the move m_i, setting

$$\vec{u_i} = {}^{\ulcorner}\vec{u}_{\leq m_{i-1}} \upharpoonright X^{\urcorner X}$$

where X depends on which strategy is to make the move m_i (which depends on the parity of i). X will be (A, B, b) or (B, C, c), where b or c is the initial B- or C-move hereditarily justifying m_i.

Define t_i and o_i to be the copycat threshold and offset of σ, or τ as appropriate, at the P-view $\vec{u_i}$. These are specified by t_σ or t_τ. Then set:

$$
\begin{array}{rcl}
t'_i & = & \begin{cases} t_i + |A|\,, & \text{if } m_i \text{ is a root of the arena } B \\ t_i, & \text{otherwise} \end{cases} \\
o'_i & = & \begin{cases} o_i + |A|\,, & \text{if } m_i \text{ is a root of the arena } B \\ o_i, & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{rcl}
T_1 & = & t'_1 \\
O_1 & = & o'_1 \\
T_{i+1} & = & \max(T_i + o'_{i+1}, t'_{i+1}) \\
O_{i+1} & = & O_i + o'_{i+1}
\end{array}
$$

$$
\begin{array}{rcl}
\mathbf{t} & = & T_{p+1} \\
\mathbf{o} & = & \begin{cases} O_{p+1} - |A| + |B|\,, & \text{if } \underline{\mathrm{m}} \text{ is a root of the arena } C \\ O_{p+1}, & \text{otherwise} \end{cases}
\end{array}
$$

Then $\mathbf{t}$ and $\mathbf{o}$ are the copycat threshold and offset of the composition $\langle \sigma, t_\sigma \rangle ; \langle \tau, t_\tau \rangle$ at the P-view $\vec{v}$.

Now we must show that this method does indeed produce an EXAC strategy, i.e. that the composite threshold function specifies valid thresholds and offsets for the composite strategy. In fact it does so only under some restrictions, for which we need an additional definition.

Definition Let σ be an EAC strategy over a single-tree arena. If σ has a first move, then it has a copycat threshold and offset, say t and o, at the P-view consisting only of the root O-move. The *l-number* of σ is the value $t - o$, and we write it $\mathbf{l}(\sigma)$.

If $\sigma = \langle \sigma_1, \ldots, \sigma_n \rangle$ is an EAC strategy over an arena with n trees, which is defined on at least one of the minimal P-views, then we define $\mathbf{l}(\sigma) = \min^n_{i=1, \sigma_i \neq \perp} \{\mathbf{l}(\sigma_i)\}$.

Theorem 4.2.2 If $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are EAC strategies satisfying $\mathbf{l}(\sigma) \geq |A|$ (or σ is everywhere undefined) and $\mathbf{l}(\tau) \geq |B|$ (or τ is everywhere undefined) then Algorithm 4.2.1 produces valid copycat thresholds and offsets for $\sigma ; \tau$.

Proof Firstly, we may assume that C is a single tree arena, because if $C = \langle C_1, \ldots, C_n \rangle$ then we can write $\tau = \langle \tau_1, \ldots, \tau_n \rangle$ and set $\sigma ; \tau = \langle \sigma ; \tau_1, \ldots, \sigma ; \tau_n \rangle$ and since $\mathbf{l}(\tau) = \min^n_{i=1} \{\mathbf{l}(\tau_i)\}$ we can be sure that the condition $\mathbf{l}(\tau_i) \geq |B|$ holds for each i where τ_i has a first move.

We choose a P-view $\vec{v}$ of the composite strategy $\sigma ; \tau$, overloading notation so that $\vec{v}$ refers to both the P-view and the sequence of natural numbers which encodes it; we do likewise for moves.

However we must be very careful when referring to "the i^{th}" child of a move m. This may well be ambiguous — in the special case where m is the root of C, the i^{th} child of the move m in the arena $A \Rightarrow C$ is the $(i - |A|)^{th}$ child of the same move in the arena C. In all other cases there is no difference which arena the move m is looked at in. In what follows $(A \mathbin{@} m)^{>n}$ will unambiguously refer to selecting the subtree rooted at m and deleting the first n subtrees of that, relative to the arena A.

Using the notation of the Algorithm, we aim to show:

(i) The arenas $((A \Rightarrow C) \mathbin{@} \underline{m})^{>(\mathbf{t}-\mathbf{o})}$ and $((A \Rightarrow C) \mathbin{@} m)^{>\mathbf{t}}$ are order-isomorphic;

(ii) If $i > \mathbf{t}$ then $f(\vec{v} \cdot i) = (i - \mathbf{o}, 1)$ and further $f(\vec{v} \cdot i \cdot \vec{w} \cdot j) = (j, 1)$ for all sequences $\vec{w}$ and numbers $j \geq 1$;

(iii) For all $\vec{w} \geq (\vec{v} \cdot k)$ with $k \leq \mathbf{t}$, if $f(\vec{w}) = (i, |\vec{w}| - |\vec{v}|)$ then $i \leq \mathbf{t} - \mathbf{o}$;

(iv) If $f(\vec{v}) = (i, 0)$ then $i \leq \mathbf{t} - \mathbf{o}$.

This is what we need, for the strategy $\sigma ; \tau$ to be AC at the P-view $\vec{v}$. We note that condition (ii) is sufficient for the composition to be EC where necessary.

We assume that neither σ nor τ is everywhere undefined; these special cases will be dealt with at the end.

Proof of (i) and (ii):

Note that σ's thresholds and offsets are relative to the arena $A \Rightarrow B$, whereas τ's are relative to $B \Rightarrow C$. We imagine, therefore, that all these moves take place in the arena $(A \Rightarrow B) \Rightarrow C$; then consideration of the values of t'_i and o'_i shows that they are the same offsets, but always relative to this new arena, which we will henceforth shorten to ABC.

We show by induction that all T_i and O_i satisfy:

(a) $(ABC \mathbin{@} \underline{m})^{>(T_i - O_i)} \cong (ABC \mathbin{@} m_i)^{>T_i}$;

(b) If $j > T_i$ and the j^{th} child (relative to the arena ABC) of the last move in the interaction $\vec{u_i}$ (i.e. $m_i j$) is played the resulting continuation has as the next visible move the P-move $\underline{m}(j - O_i)$ (relative to the arena ABC), justified by $\underline{m}$;

(c) Furthermore a subsequent move $\underline{m}(j - O_i)w_1$ will result in an interaction ending in $m_i j w_1$, and in general for any odd-length sequence $\vec{w}$ the move $\underline{m}(j - O_i)\vec{w}$ will result in an interaction ending in $m_i j \vec{w}$ and for even-length sequences $\vec{w}$ the move $m_i j \vec{w}$ will result in an interaction ending in $\underline{m}(j - O_i)\vec{w}$.

For $i = p + 1$ the moves $m_i = m$ and $m_i j$ and their descendants are visible in the arena C.

For $i = 1$ the properties we require are just the AC property of σ or τ at $\vec{u_0}$.

Suppose that we have proved the result for T_i and O_i.

Either $T_{i+1} = T_i + o'_{i+1}$, so we must have $T_i + o'_{i+1} \geq t'_{i+1}$, say $T_i = t'_{i+1} - o'_{i+1} + l$. Then

$$\begin{array}{rlcl} & (ABC \,@\, \underline{m})^{>(T_i - O_i)} & \cong & (ABC \,@\, m_i)^{>T_i} \\ \text{by AC property of } \sigma \text{ or } \tau, & (ABC \,@\, m_i)^{>(t'_{i+1} - o'_{i+1})} & \cong & (ABC \,@\, m_{i+1})^{>t'_{i+1}} \\ \text{hence} & (ABC \,@\, m_i)^{>(t'_{i+1} - o'_{i+1} + l)} & \cong & (ABC \,@\, m_{i+1})^{>(t'_{i+1} + l)}. \end{array}$$

But $T_{i+1} - O_{i+1} = T_i - O_i$ and $T_{i+1} = t'_i + l$ hence

$$(ABC \,@\, \underline{m})^{>(T_{i+1} - O_{i+1})} \cong (ABC \,@\, m_{i+1})^{>T_{i+1}}.$$

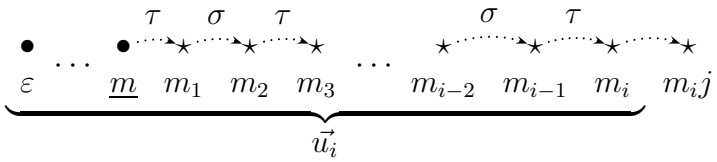
Or $T_{i+1} = t'_{i+1}$, so we must have $T_i + o_{i+1} \leq t'_{i+1}$, say $T_i = t'_{i+1} - o'_{i+1} - l$. Then

$$\begin{array}{rlcl} & (ABC \,@\, \underline{m})^{>(T_i - O_i)} & \cong & (ABC \,@\, m_i)^{>T_i} \\ \text{hence} & (ABC \,@\, \underline{m})^{>(T_i - O_i + l)} & \cong & (ABC \,@\, m_i)^{>(T_i + l)} \\ \text{by AC property of } \sigma \text{ or } \tau, & (ABC \,@\, m_i)^{>(t'_{i+1} - o'_{i+1})} & \cong & (ABC \,@\, m_{i+1})^{>t'_{i+1}}. \end{array}$$

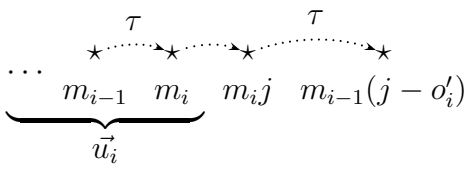
But $T_{i+1} - O_{i+1} = t'_{i+1} - o'_{i+1} - O_i = T_i - O_i + l$ hence

$$(ABC \,@\, \underline{m})^{>(T_{i+1} - O_{i+1})} \cong (ABC \,@\, m_{i+1})^{>T_{i+1}}.$$

Now for the proof of (b) and (c) suppose that the strategy dictating the first move of the interaction is τ, i.e. $\underline{m}$ is in the arena C (the other case is similar and will require some τ's to be replaced by σ's and one other difference which we note below). Suppose that the move $m_i j$ is made after the move m_i, with $j > T_i$. Then the interaction sequence looks like:

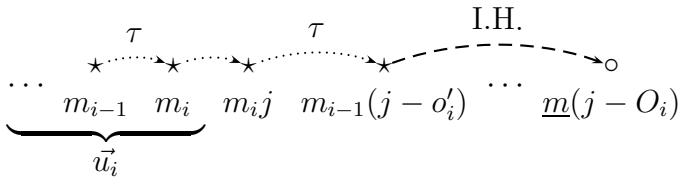


(Here and hereafter we are omitting the justification pointers, because where they should point to are determined by the property of AC.) We can predict how this interaction would continue: since $j > T_i \geq t'_i$ we know by the EAC property of τ that we must continue:



Here $m_i j$ refers to the j^{th} child of m relative to the arena ABC, and $m_{i-1}(j - o'_i)$ is the $(j - o'_i)^{th}$ child relative to the arena ABC.

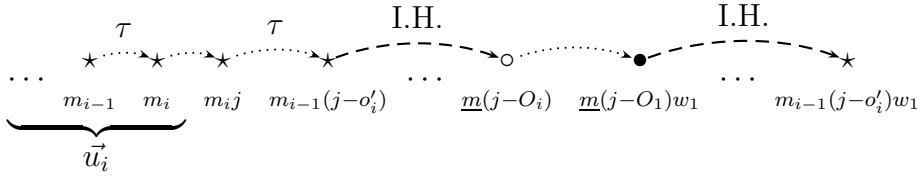
But $j - o'_i > T_i - o'_i \geq T_{i-1}$ so by the induction hypothesis we know that this interaction must continue until it reaches the visible P-move $\underline{m}(j - o'_i - O_{i-1}) = \underline{m}(j - O_i)$:



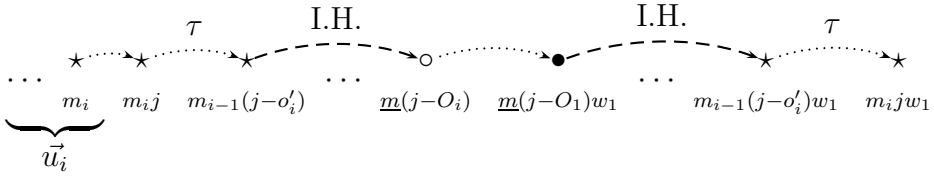
We have to be a little careful, because the move $\underline{m}(j - O_i)$ has to be visible, i.e. it is in the arena C. If the move $\underline{m}$ is a root of C then this will only be true as long as $j - O_i > |B|$. Thankfully in this case we can use that $\mathbf{l}(\tau) \geq |B|$ so that $j - O_i > T_i - O_i \geq t'_1 + \sum_{j=2}^{i-1} o'_j - \sum_{j=1}^{i-1} o'_j = t'_1 - o'_1 = t_1 - o_1 \geq |B|$.

Of course, there will be no such problem in the case that σ was the strategy to dictate the move immediately after $\underline{m}$, because $\underline{m}$ would have to be a A-move. However, in this case we have to show that the move $m_{i-1}(j - o'_i)$ is not visible, i.e. it is in the arena B. There is no problem unless m_{i-1} is a root of B. Then similarly to above, we can use the condition that $\mathbf{l}(\sigma) \geq |A|$ giving $j - o'_i > T_i - o'_i \geq t'_i - o'_i = t_i - o_i$. Now since the move m_{i-1} is a root of B this value is the l-number of σ, hence the required result.

If there is a further P-move $\underline{m}(j - O_i)w_1$, by the inductive hypothesis, the following must result:



and because τ must be EC at this view, we have:



And it is clear how subsequent moves will be copied by τ, and the induction hypothesis will always apply, giving the desired result.

Finally we note that in order to consider the children of the root of C to be labelled with respect to $A \Rightarrow C$ rather than ABC we must set $\mathbf{o} = O_{p+1}$, unless $\underline{m}$ is a root of C in which case $\mathbf{o} = O_{p+1} - |A| + |B|$. We can set $\mathbf{t} = T_{p+1}$ in any case, because the threshold refers to children of m, and m could not be a root of C.

Property (a) proves (i); (b) and (c) together give (ii).

Proof of (iii):

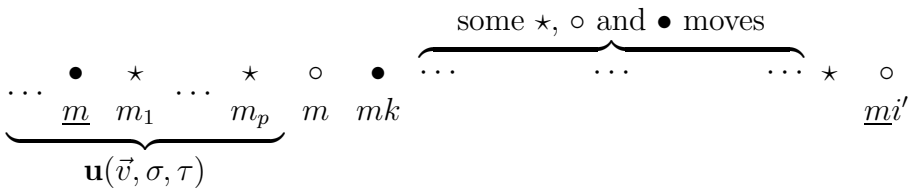
We require a little result for use later:

Directly from the definition we have that $T_{i+1} \geq T_i + o'_{i+1}$ so $T_{p+1} \geq T_n + \sum_{i=n+1}^{p+1} o'_i$. Also we have $O_{p+1} = \sum_{i=1}^{p+1} o'_i$ and $T_n \geq t'_n$. Thus

$$T_{p+1} - O_{p+1} \geq T_n - \sum_{i=1}^{n} o'_i \geq t'_n - \sum_{i=1}^{n} o'_i.$$

Now suppose that we have a sequence $\vec{w}$ such that $f_{\sigma;\tau}(\vec{v} \cdot k \cdot \vec{w}) = (i, |\vec{w}| + 1)$ with $k \leq \mathbf{t}$ (here $f_{\sigma;\tau}$ is the economical form of the innocent function of the composition). We aim to show that $i \leq \mathbf{t} - \mathbf{o}$. We would prefer to talk in terms of children of moves in the arena ABC, in which case we say that the resulting move is $\underline{m}i'$. Here $i' = i$ unless $\underline{m}$ is a root of C, in which case $i' = i + |B| - |A|$. Then we want to show that $i' \leq T_{p+1} - O_{p+1}$.

All we know about the interaction which produced this move is that it must be of the form:



Let us write $\vec{u} = \mathbf{u}(\vec{v} \cdot k \cdot \vec{w}, \sigma, \tau)$. Since the move $\underline{m}i'$ is justified by $\underline{m}$ it must be in the arena A and it was σ which caused it. So we examine the view σ saw when making that move:

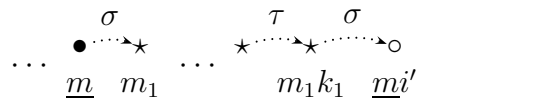
$$\vec{u_1} = \ulcorner \vec{u} \upharpoonright X \urcorner^X$$

where as before X is the appropriate component. As the move $\underline{m}$ must be in this view (because of the visibility condition) so there cannot be justification pointers from O-moves jumping over it. Hence $\vec{u_1} = \ulcorner \vec{u_{1 \leq \underline{m}}} \urcorner \cdot \vec{x} = \ulcorner \mathbf{u}(\vec{v}, \sigma, \tau)_{\leq \underline{m}} \upharpoonright X \urcorner \cdot \vec{x}$, for some sequence $\vec{x}$, and $f(\vec{u_1}) = (i, |x|)$, where f is f_σ or f_τ as appropriate. Again, in terms of the arena ABC, this resulting move is child number i' of the move justifying it.

Now we look at the first element of $\vec{x}$, call it k_1.

Either $k_1 \leq t'_1$. But then by the EAC property of σ or τ we must have $i' \leq t'_1 - o'_1 \leq T_{p+1} - O_{p+1}$ by the result shown above. So the result holds in this case.

Or $k_1 > t'_1$. But then the EAC property of σ or τ forces that $\vec{x} = \langle k_1 \rangle$ with $i' = k_1 - o'_1$. So the uncovering must look like:



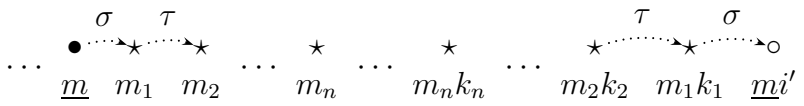
Because to whichever strategy was last to act the move of the interaction m_1 is a P-move, we can be sure it is in view when the move $\underline{m}i'$ is made.

So we know that the move m_1k_1 was made by either σ or τ. Again we look at the view it had:

$$\vec{u_2} = \ulcorner \vec{u}_{\leq m_1k_1} \upharpoonright X \urcorner^X$$

As before we can deduce that $\vec{u_2} = \ulcorner \mathbf{u}(\vec{v}, \sigma, \tau)_{\leq m_1} \upharpoonright X \urcorner \cdot \vec{x}$ for some sequence $\vec{x}$ and again we examine the first move, call it k_2. Either $k_2 \leq t'_2$, whence as before we can deduce $k_1 \leq t'_2 - o'_2$ hence $i' \leq t'_2 - o'_2 - o'_1 \leq T_{p+1} - O_{p+1}$, or $k_2 > t'_2$ so that $\vec{x} = \langle k_2 \rangle$ and $k_1 = k_2 - o'_2$.

We can use the argument inductively: at each stage the interaction looks like:



and because of the EAC property of σ or τ at $\ulcorner \mathbf{u}(\vec{v}, \sigma, \tau)_{\leq m_n} \upharpoonright X \urcorner$ either $k_n \leq t'_n$, whence $i' \leq t'_n - \sum_{i=1}^{n} o'_i \leq T_{p+1} - O_{p+1}$, or $k_{n-1} = k_n - o'_n$ and we can apply the argument again.

At the last stage, when $n = p + 1$, we actually know that the sequence $\vec{x}$ is singleton, because the move $m_{p+1} = m$ is answered directly by mk, by hypothesis. So $i' = k - \sum_{i=1}^{p+1} o'_i$ (by chasing back the definitions of k_n) and $k \leq T_{p+1}$ by hypothesis, hence $i' \leq T_{p+1} - O_{p+1}$. All cases have now been covered.

Proof of (iv):

The move m must be $\underline{m}i$ and be justified by $\underline{m}$. We need to show that the EAC properties of σ and τ force $i \leq \mathbf{t} - \mathbf{o}$. Again, relative to the arena ABC we can say that the move m is the i'^{th} child of $\underline{m}$, where i' is as above. Again, we wish to show that $i' \leq T_{p+1} - O_{p+1}$.

The proof is similar to that of (iii). We consider the sequence which each strategy had in view, when making the move $\underline{m}i'$, $\vec{u_1} = \ulcorner \mathbf{u}(\vec{v}, \sigma, \tau) \upharpoonright X \urcorner^X$. Once again the move $\underline{m}$ must be in view, so $\vec{u_1} = \ulcorner \mathbf{u}(\vec{v}, \sigma, \tau)_{\leq \underline{m}} \urcorner \cdot \vec{x}$. We look at the first element of $\vec{x}$, k_1 say, and as before either $k_1 \leq t'_1$ — exactly as before this leads to $i' \leq T_{p+1} - O_{p+1}$ — or we can deduce that the move before m is m_1k_1 and apply the argument inductively. Note that, unlike in (iii), the move before m is actually m_p.

After n steps we will be have either $k_n \leq t'_n$, hence $i' \leq T_{p+1} - O_{p+1}$, or $m_{p-n+1} = m_nk_n$. So this time we reach the final stage after $p/2$ steps (we know p must be even because $\underline{m}$ and m are in the same component) and here have no choice but that $m_{p/2+1} = m_{p/2}k_{p/2}$, hence $k_{p/2} \leq t'_{p/2} - o'_{p/2}$ and $i' \leq T_{p+1} - O_{p+1}$ follows directly from the result shown at the start of the proof of (iii).

Special Cases:

If τ is everywhere undefined and then so is the composition. If σ is everywhere undefined then for the composition to be defined on the P-view $\vec{v}$ the moves $\underline{m}$ and m must be in C, with no intermediate moves in between. Now we can treat this in the same way as the usual cases. ■

We will also need the following:

Lemma 4.2.3 Composition of EXAC strategies is associative.

Proof The proof is straightforward and we omit it. One proceeds by examining the uncovering of three strategies together (that is, the sequence including intermediate moves in both hidden arenas) and relating this to the uncoverings of the two different bracketings of the triple composition. ■

4.3 The Category $\mathbb{A}_{\mathrm{EXAC}}$

We first define the "obvious" category, which derives directly from the conditions required for the composition algorithm to work correctly. Perhaps surprisingly we can show that this does not give rise to a CCC.

Definition The *category of arenas and EXAC strategies*, written $\mathbb{A}_{\mathrm{EXAC}}$, is defined as follows.

(i) Objects are r.e. arenas.

(ii) The morphisms from A to B are the EXAC strategies on the arena $A \Rightarrow B$ which have l-number greater than or equal to $|A|$, or are everywhere undefined.

(iii) The identity morphism on A is the EXAC strategy $\langle \mathrm{id}_A, 0 \rangle$, i.e. the copycat threshold is zero everywhere. The offset at the minimal P-view is $-|A|$, and 0 everywhere else.

(iv) Composition of morphisms is given by composition of EXAC strategies via algorithm 4.2.1.

One can show that this does indeed specify a category. We already have associativity of composition, and we must also show:

Lemma 4.3.1

(i) If $\sigma : A \Rightarrow B$ satisfies $\mathbf{l}(\sigma) \geq |A|$ (or is everywhere undefined) and $\tau : B \Rightarrow C$ satisfies (or is everywhere undefined) $\mathbf{l}(\tau) \geq |B|$ then $\mathbf{l}(\sigma\,;\tau) \geq |A|$, or $\sigma\,;\tau$ is everywhere undefined.

(ii) For any morphism $\sigma : A \to B$ we have $\sigma\,;\mathrm{id}_B = \sigma = \mathrm{id}_A\,;\sigma$.

Proof Both are elementary consequences of the construction of the composition

algorithm. ■

Also, $\mathbb{A}_{\mathrm{EXAC}}$ has the obvious terminal object E (the arena consisting of no trees defined in Example 2.1.2) and products given in the usual way. However,

Theorem 4.3.2 $\mathbb{A}_{\mathrm{EXAC}}$ does not form a CCC with the usual constructions.

Proof Suppose that $\sigma : A \times B \to C$. Then we know that $\mathbf{l}(\sigma) \geq |A| + |B|$. We need a morphism $\Lambda(\sigma) : A \to B \Rightarrow C$, which must have l-number at least $|A|$, so we could take $\Lambda(\sigma)$ to be the same EXAC strategy as σ. However this choice may not be unique.

For example, consider η_0 and η_1 as defined earlier in this section. Note that the only difference between the two strategies is their threshold for the initial P-view. One can verify that both η_0 and η_1 can be considered as morphisms $U \to U \Rightarrow U$ and that in this case $\eta_0 \times \mathrm{id}_U \,;\, \mathrm{eval}_{U,U} = \eta_1 \times \mathrm{id}_U \,;\, \mathrm{eval}_{U,U} : U \times U \to U$, and that this is the same as the morphism $U \times U \to U$ described by η_1. Hence there are two candidates for $\Lambda(\eta_1)$. ■

We should perhaps also say that it is not clear that $\mathbb{A}_{\mathrm{EXAC}}$ forms a CCC with any unusual constructions either.

In order to fix this up, we made another attempt. The problem with $\mathbb{A}_{\mathrm{EXAC}}$ is that the conditions for an EXAC strategy to be a morphism $A \to (B \Rightarrow C)$ are weaker than those to be a morphism on $A \times B \to C$. One solution might be the following:

(i) An object is a pair (A, n) where $n \in \mathbb{N}_0$.

(ii) A morphism $\sigma : (A, n) \to (B, m)$ is an EXAC strategies on $A \Rightarrow B$ such that $\mathbf{l}(\sigma) \geq m + |A|$.

(iii) Composition is composition of EXAC strategies.

(iv) The identity on (A, n) is $\langle \mathrm{id}_A, t \rangle$, where t is the function mapping the minimal P-view to n and the others to zero.

Then we can set $(A, n) \Rightarrow (B, m) = (A \Rightarrow B, m + |A|)$, which gives the same set of morphisms $A \times B \to C$ and $A \to B \Rightarrow C$.

However, in this case, the identity will not work correctly! Sometimes the thresholds of $\sigma \,;\, \mathrm{id}$ come out greater than σ. To fix this, we find we must include information in the objects specifying minimal thresholds for the morphisms. But this breaks the function spaces again, and we have to add information specifying the minimum l-number for some other P-views, whereupon there are again problems with identities...

What we believe to be the least fixed point of the fixing-up process is presented in the next section.

4.4 The Category $\mathbb{XA}$

We now present a new category based on EXAC strategies, which does form a CCC. Although it is more complicated than the "almost-CCC" $\mathbb{A}_{\mathrm{EXAC}}$, it seems to be the simplest way to construct a CCC.

Firstly let us write $\mathrm{br}(A)$ for the number of branches of a tree at the root (assuming that this is finite). Then we can write $\mathrm{br}(A \mathbin{@} m)$, for any move m of a finitely-branching forest A, to mean the number of direct children of m in A. Then we make the following definition:

Definition Let A be an r.e. arena and X a finitely-branching r.e. sub-arena of A. We say that an EXAC strategy σ over A is *X-explicit* if the following holds:

Let $\sigma : \vec{v} \mapsto (i, r, t, o)$ be the economical form of any clause of the innocent function. Suppose that the sequence $\vec{v}$ codes a P-view ending in the O-move $\underline{m}$, and that the consequent P-move encoded by this clause is m. Then

(i) if $\underline{m}$ is in the sub-arena X then $t - o \geq \mathrm{br}(X \mathbin{@} \underline{m})$,

(ii) if m is in the sub-arena X then $t \geq \mathrm{br}(X \mathbin{@} m)$.

An intuitive description of this definition is the following: The subarena X determines a part of the arena A where the strategy is known to be explicitly defined, i.e. that is neither in the domain nor the range of automatic copycat forced by the threshold information of σ. This means that given a strategy σ over A which is X-explicit, any P-view of σ with moves only in X is entirely explicit.

The following lemma will be useful later.

Lemma 4.4.1 If A is an r.e. arena of which $M \Rightarrow M$ is a subarena (i.e. A is a single tree r.e. arena which is not equal to E or M) then every EXAC strategy over A is $(M \Rightarrow M)$-explicit.

Proof Let σ be an EXAC strategy over such an arena A. Either $\sigma = \bot$, in which case it is vacuously $(M \Rightarrow M)$-explicit, or it makes a move in response to the initial O-move ε.

The only P-view on which the conditions for σ to be $(M \Rightarrow M)$-explicit come into play is the initial P-view (when $\mathrm{br}((M \Rightarrow M) \mathbin{@} \varepsilon) = 1$), as $\mathrm{br}((M \Rightarrow M) \mathbin{@} m) = 0$ for any P-move m and $\underline{m} \notin (M \Rightarrow M)$ for any non-initial move O-move $\underline{m}$. Thus we only need to show that $\mathbf{l}(\sigma) \geq 1$. But suppose that $\mathbf{l}(\sigma) = 0$ — then by the condition (iv) of the definition of AC at the initial P-view we must have that the innocent function of σ maps ε to $(i, 0)$ with $i \leq \mathbf{l}(\sigma)$, a contradiction if $\sigma \neq \bot$. ■

We can now define a category of EXAC strategies which will form a CCC.

Definition The category $\mathbb{XA}_{\mathrm{EXAC}}$, or simply $\mathbb{XA}$, is given by the following:

(i) Objects are pairs (A, X) consisting of a r.e. arena A and a finitely-branching r.e. sub-arena X.

(ii) A morphism $\sigma : (A, X) \to (B, Y)$ is an EXAC morphism on $A \Rightarrow B$ which is $(X \Rightarrow Y)$-explicit.

(iii) Composition of morphisms is composition of EXAC strategies via Algorithm 4.2.1.

(iv) The identity strategy on (A, X), $\mathrm{id}_{(A,X)}$, is the EXAC strategy $\langle \mathrm{id}_A, t \rangle$, where id_A is the EAC identity strategy on A, and t is the function that takes the least value on every P-view which still leaves the EXAC strategy $\langle \mathrm{id}_A, t \rangle$ as $(X \Rightarrow X)$-explicit.

Theorem 4.4.2 This does form a category. We already know that composition is associative so it remains to show that:

Composition of morphisms is well-defined: If $\sigma : (A, X) \to (B, Y)$ and $\tau : (B, Y) \to (C, Z)$ then $\sigma \,;\, \tau$ is $(X \Rightarrow Z)$-explicit.

Identities work as required: If $\sigma : (A, X) \to (B, Y)$ then $\sigma \,;\, \mathrm{id}_{(B,Y)} = \sigma = \mathrm{id}_{(A,X)} \,;\, \sigma$.

Proof

Composition of morphisms is well-defined:

Take any P-view $\vec{v}$ on which the composite strategy $\sigma \,;\, \tau$ is defined. Suppose that the last move of this P-view is $\underline{m}$ and the resulting P-move is m.

Firstly suppose that $\underline{m} \in X \Rightarrow Z$. We will have to take the special case when $\underline{m}$ is a root of C separately, so first assume that this is not the case. Then we know that either $\underline{m} \in X \Rightarrow Y$ or $\underline{m} \in Y \Rightarrow Z$, depending on whether σ or τ makes the next (possibly hidden) move after $\underline{m}$. In either case, in the notation of the Composition Algorithm, we that $t_1 - o_1 \geq \mathrm{br}(X \Rightarrow Z \mathbin{@} \underline{m})$ by hypothesis. But then if we examine $\mathbf{t}$ and $\mathbf{o}$, the threshold and offset of the composition at this P-view, we see that:

$$\begin{array}{rcl} \mathbf{t} - \mathbf{o} & = & T_{p+1} - O_{p+1} \\ & \geq & (t'_1 + \sum_{i=2}^{i=p+1} o'_i) - (\sum_{i=1}^{i=p+1} o'_i) \\ & = & t'_1 - o'_1 \\ & = & t_1 - o_1 \\ & \geq & \mathrm{br}(X \Rightarrow Z \mathbin{@} \underline{m}) \end{array}$$

In the special case when $\underline{m}$ is the root of C, we have that $\mathbf{o} = O_{p+1} - |A| + |B|$, but also we know that the strategy making the first move after $\underline{m}$ is τ, and that

$\mathrm{br}(X \Rightarrow Z \mathbin{@} \underline{m}) = \mathrm{br}(Y \Rightarrow Z \mathbin{@} \underline{m}) + |A| - |B|$, so the above reasoning is still sound.

Secondly, suppose that $m \in X \Rightarrow Z$. There are no special cases; we always have that $t'_{p+1} = t_{p+1} \geq \mathrm{br}(X \Rightarrow Z \mathbin{@} m)$ by hypothesis (this is because m cannot be a root of B or C). But $\mathbf{t} = T_{p+1} \geq t'_{p+1}$ so that $\mathbf{t} \geq \mathrm{br}(X \Rightarrow Z \mathbin{@} m)$.

Hence $\sigma \,;\tau$ is $X \Rightarrow Z$-explicit.

Identities work as required:

It is simple to verify that the identity on (A, X) has the following economical form. Suppose that $A = \langle A_1, \ldots, A_n \rangle$, so that $\mathrm{id}_{(A,X)} = \langle \mathrm{id}_1, \ldots, \mathrm{id}_n \rangle$, say. Then:

$$\begin{array}{rrcl} \mathrm{id}_i : & \varepsilon & \mapsto & (i, 0, \mathrm{br}(X_i \mathbin{@} \varepsilon), -n) \\ & t & \mapsto & (t + n, 1, \mathrm{br}(X_i \mathbin{@} t), 0) \\ & \vec{s} \cdot t & \mapsto & (t, 1, \mathrm{br}(X_i \mathbin{@} (\vec{s} \cdot t)), 0) \text{ for all nonempty sequences } \vec{s} \end{array}$$

Here we have extended the definition of br slightly, to have that $\mathrm{br}(A \mathbin{@} m) = 0$ when m is not in A.

That is, the copycat threshold of $\mathrm{id}_{(A,X)}$ at a P-view ending in the O-move coded by $\underline{m}$ in either of the two copies of the arena A (which copy it will be depends on the parity of the length of $\underline{m}$) is the number of children of $\underline{m}$ in X, or zero if $\underline{m}$ is not in X.

We note that, since X is an r.e. arena, the threshold function of the identity strategy is given by a recursive function as we require for it to be EAC.

Now we know that for an EAC strategy σ on $A \Rightarrow B$, $\mathrm{id}_A \,;\sigma = \sigma = \sigma \,;\mathrm{id}_B$, where id_A and id_B are the EAC identity strategies. It remains to show that this still holds when one also includes copycat thresholds and offsets. We will only consider $\mathrm{id}_A \,;\sigma$, as the other proof is very similar and uses no additional techniques.

Let us select some P-view $\vec{v}$ on which $\mathrm{id}_A \,;\sigma$ is defined, and suppose that it ends in the O-move $\underline{m}$, with m the resulting P-move. There are four cases:

(i) Both $\underline{m}$ and m are in B.

(ii) $\underline{m}$ is in B, m is in some component (A, a).

(iii) m is in B, $\underline{m}$ is in some component (A, a).

(iv) $\underline{m}$ is in (A, a_1), m is in (A, a_2).

Let us write t_σ and o_σ for the copycat threshold and offset of σ at the same P-view $\vec{v}$. In each case we want to show that these match the copycat threshold and offset of the composition. In what follows "condition (i)" (or (ii)) refers to the fact that σ is $X \Rightarrow Y$-explicit, at the P-view $\vec{v}$.

(i) Is completely trivial.

(ii) Examining the way the EAC identity strategies work (simply copycat) we see that the uncovering must be of the form

$$\begin{array}{cccc} \cdots & \bullet & \star & \circ \\ & \underline{m} & m_1 & m \end{array}$$

where m is the same move in A as m_1, since it was arrived at by copycat. σ made the move m_1 in response to the P-view $\vec{v}$.

There are three cases. Let us first assume that $\underline{m}$ is not a root of B and m (equivalently m_1) not a root of A. Then, in the notation of the Composition Algorithm, we have:

$$\begin{array}{ll} t'_1 = t_\sigma & o'_1 = o_\sigma \\ t'_2 = \mathrm{br}(X \mathbin{@} m_1) & o'_2 = 0 \\ T_1 = t_\sigma & O_1 = o_\sigma \\ T_2 = \max(t_\sigma, \mathrm{br}(X \mathbin{@} m_1)) \stackrel{*}{=} t_\sigma & O_2 = o_\sigma \end{array}$$

(* by condition (ii)) Hence $\mathbf{t} = t_\sigma$ and $\mathbf{o} = o_\sigma$ as required.

In the case when $\underline{m}$ is not a root of B but m is a root of A, we have similarly:

$$\begin{array}{ll} t'_1 = t_\sigma + |A| & o'_1 = o_\sigma + |A| \\ t'_2 = \mathrm{br}(X \mathbin{@} m_1) & o'_2 = -|A| \\ T_1 = t_\sigma + |A| & O_1 = o_\sigma + |A| \\ T_2 = \max(t_\sigma, \mathrm{br}(X \mathbin{@} m_1)) \stackrel{*}{=} t_\sigma & O_2 = o_\sigma \end{array}$$

(* by condition (ii))

In the case when $\underline{m}$ is a root of B, we must have that m is a root of A, and the calculation above applies.

(iii) Very similar to (ii), with no special cases.

(iv) As before, we see that the uncovering must be of the form

$$\begin{array}{cccccc} \cdots & \bullet & \star & \star & \circ \\ & \underline{m} & m_1 & m_2 & m \end{array}$$

where $\underline{m}$ is the same move in A as m_1, and m_2 is the same move in A as m. The move m_2 is made by σ in response to the P-view $\vec{v}$.

There are two cases this time. First suppose that m_2 is not a root of A. Then:

$$\begin{array}{ll} t'_1 = \mathrm{br}(X \mathbin{@} \underline{m}) = \mathrm{br}(X \mathbin{@} m_1) & o'_1 = 0 \\ t'_2 = t_\sigma & o'_2 = o_\sigma \\ t'_3 = \mathrm{br}(X \mathbin{@} m_2) & o'_3 = 0 \\ T_1 = \mathrm{br}(X \mathbin{@} m_1) & O_1 = 0 \\ T_2 = \max(\mathrm{br}(X \mathbin{@} m_1) + o_\sigma, t_\sigma) \stackrel{*}{=} t_\sigma & O_2 = o_\sigma \\ T_3 = \max(t_\sigma, \mathrm{br}(X \mathbin{@} m_2)) \stackrel{**}{=} t_\sigma & O_3 = o_\sigma \end{array}$$

(* by condition (i), ** by condition (ii)). If m_2 is a root of A then:

$$
\begin{array}{ll}
t'_1 = \mathrm{br}(X \mathbin{@} \underline{m}) = \mathrm{br}(X \mathbin{@} m_1) & o'_1 = 0 \\
t'_2 = t_\sigma + |A| & o'_2 = o_\sigma + |A| \\
t'_3 = \mathrm{br}(X \mathbin{@} m_2) & o'_3 = -|A| \\
T_1 = \mathrm{br}(X \mathbin{@} m_1) & O_1 = 0 \\
T_2 = \max(\mathrm{br}(X \mathbin{@} m_1) + o_\sigma + |A|, t_\sigma + |A|) \stackrel{*}{=} t_\sigma + |A| & O_2 = o_\sigma + |A| \\
T_3 = \max(t_\sigma, \mathrm{br}(X \mathbin{@} m_2)) \stackrel{**}{=} t_\sigma & O_3 = o_\sigma
\end{array}
$$

(* by condition (i), ** by condition (ii)). So either way the result holds.

■

Theorem 4.4.3 The following constructions make $\mathbb{XA}$ into a CCC:

Terminal Object: The terminal object is (E, E).

Binary Products: The product $(A, X) \times (B, Y)$ is $(A \times B, X \times Y)$. The projection strategy $\pi^{(A,X)\times(B,Y)}_{(A,X)}$ is given by the EXAC strategy $\langle \pi^{A\times B}_A, t\rangle$, where $\pi^{A\times B}_A$ is the EAC projection strategy and t is the threshold function specifying the least value at each P-view to make this EXAC strategy $((X \times Y) \Rightarrow X)$-explicit. Similarly for the other projection.

Exponentials: The exponential object $(A, X) \Rightarrow (B, Y)$ is $(A \Rightarrow B, X \Rightarrow Y)$. The evaluation map $\mathrm{eval}_{(B,Y),(C,Z)}$ is the same EXAC strategy as $\mathrm{id}_{(B,Y)\Rightarrow(C,Z)}$.

Proof

Terminal Object:

The only morphisms with the terminal object as codomain are clearly the 0-tuples.

Binary Products:

Suppose that $A = \langle A_1, \ldots, A_m\rangle$ and $B = \langle B_1, \ldots, B_n\rangle$, so that $\pi^{(A,X)\times(B,Y)}_{(A,X)}$ is an m-tuple, $\langle \pi_1, \ldots, \pi_m\rangle$.

Then it is easy to check that π_i has the following economical form:

$$
\begin{array}{rcl}
\pi_i : \quad \varepsilon & \mapsto & (i, 0, \mathrm{br}(X_i \mathbin{@} \varepsilon), -n - m) \\
t & \mapsto & (t + n + m, 1, \mathrm{br}(X_i \mathbin{@} t), 0) \\
\vec{s} \cdot t & \mapsto & (t, 1, \mathrm{br}(X_i \mathbin{@} (\vec{s} \cdot t)), 0) \text{ for all nonempty sequences } \vec{s}
\end{array}
$$

We claim that give any morphisms $\sigma = \langle \sigma_1, \ldots, \sigma_m\rangle : (C, Z) \to (A, X)$ and $\tau = \langle \tau_1, \ldots, \tau_n\rangle : (C, Z) \to (B, Y)$, we have $\langle \sigma_1, \ldots, \sigma_m, \tau_1 \ldots, \tau_n\rangle ; \pi^{(A,X)\times(B,Y)}_{(A,X)} = \sigma$. We already know that this holds for the EAC strategy part of the EXAC strategies under consideration, and the fact that it also holds for the threshold functions is

almost entirely similar to the proof that identities work as required when composed on the right.

There is only one case where the copycat thresholds and offsets of $\pi^{(A,X)\times(B,Y)}_{(A,X)}$ differ from those of $\mathrm{id}_{(A,X)}$, namely the initial P-views, so this leaves two cases to check.

Either the first move of σ is in A or C. If the former, the interaction sequence of the first move of the composition is:

$$\begin{array}{cccccc} \cdots & \bullet & \star & \star & \circ \\ & \underline{m} & m_1 & m_2 & m \end{array}$$

where $\underline{m}$ is a root of A, m_1 the corresponding root of $A \times B$, and m_2 and m are the first A-move made by σ.

Then using the same techniques as above we can show that, if t_σ and o_σ are the threshold and offset of σ at the initial P-view, we have:

$$\begin{array}{ll} t'_1 = \mathrm{br}(X \mathbin{@} \varepsilon) + |C| = \mathrm{br}((Z \Rightarrow X) \mathbin{@} \varepsilon) & o'_1 = -n - m + |C| \\ t'_2 = t_\sigma & o'_2 = o_\sigma \\ t'_3 = \mathrm{br}(X \mathbin{@} m_2) & o'_3 = 0 \\ T_1 = \mathrm{br}(X \mathbin{@} \varepsilon) + |C| & O_1 = -n - m + |C| \\ T_2 = \max(\mathrm{br}(X \mathbin{@} \varepsilon) + |C| + o_\sigma, t_\sigma) \stackrel{*}{=} t_\sigma & O_2 = o_\sigma + |C| - n - m \\ T_3 = \max(t_\sigma, \mathrm{br}(X \mathbin{@} m_2)) \stackrel{**}{=} t_\sigma & O_3 = o_\sigma + |C| - n - m \end{array}$$

(* by condition (i), ** by condition (ii)). So, because we are in the special case where $\underline{m}$ is the root, $\mathbf{t} = t_\sigma$ and $\mathbf{o} = o_\sigma$.

In the other case, the first move made by σ is immediately visible, so the interaction sequence looks like:

$$\begin{array}{cccccc} \cdots & \bullet & \star & \circ \\ & \underline{m} & m_1 & m \end{array}$$

where $\underline{m}$ a the root of A, m_i the corresponding root of $A \times B$, and m the root of C.

$$\begin{array}{ll} t'_1 = \mathrm{br}(X \mathbin{@} \varepsilon) + |C| = \mathrm{br}((Z \Rightarrow X) \mathbin{@} \varepsilon) & o'_1 = -n - m + |C| \\ t'_2 = t_\sigma & o'_2 = o_\sigma \\ T_1 = \mathrm{br}(X \mathbin{@} \varepsilon) + |C| & O_1 = -n - m + |C| \\ T_2 = \max(\mathrm{br}(X \mathbin{@} \varepsilon) + |C| + o_\sigma, t_\sigma) \stackrel{*}{=} t_\sigma & O_2 = o_\sigma + |C| - n - m \end{array}$$

(* by condition (i)). So in either case $\mathbf{t} = t_\sigma$ and $\mathbf{o} = o_\sigma$.

Since the strategy $\langle \sigma_1, \ldots, \sigma_m, \tau_1 \ldots, \tau_n \rangle$ is clearly $Z \Rightarrow (X \times Y)$-explicit, this is sufficient to prove that this works as the tupled morphism, and also the required universal property of the projection. The proof for the other projection is, of course, entirely similar.

Exponentials:

If σ is any morphism $(A, X) \to ((B, Y) \Rightarrow (C, Z))$ we claim that the composition $\sigma \times \mathrm{id}_{(B,Y)}$; $\mathrm{eval}_{(B,Y),(C,Z)}$ is a morphism $(A, X) \times (B, Y) \to (C, Z)$ which is the same EXAC strategy as σ. Hence for $\tau : (A, X) \times (B, Y) \to (C, Z)$, the curried morphism $\Lambda(\tau)$ is uniquely given by the same EXAC strategy as τ.

As before, we already know this result to be true as far as the moves of the strategies go, and it remains to check the thresholds and offsets. The proof contains a considerable amount of tedious detail, all in a similar vein to that shown above, of which we give an outline.

Firstly, we note that $\sigma \times \mathrm{id}_{(B,Y)}$ is shorthand for

$$\langle \pi^{(A,X)\times(B,Y)}_{(A,X)} ; \sigma, \pi^{(A,X)\times(B,Y)}_{(B,Y)} ; \mathrm{id}_{(B,Y)} \rangle.$$

Let us write this as $\langle \sigma', \pi^{(A,X)\times(B,Y)}_{(B,Y)} \rangle$. Recall that the thresholds and offsets of the projection morphisms do not differ very much from those of the relevant identity morphisms: routine calculation shows that σ' has exactly the same thresholds as σ, and also the same offsets, except that the offset at any minimal P-view is decreased by $|B|$.

Then it remains to show that, at any P-view $\vec{v}$ of the composition $\langle \sigma', \pi^{(A,X)\times(B,Y)}_{(B,Y)} \rangle$; $\mathrm{eval}_{(B,Y),(C,Z)}$, the copycat threshold and offset matches that of σ. As before, suppose that the P-view ends in the move $\underline{m}$ and has resulting move m, and write t_σ and o_σ for the copycat threshold and offset of $\sigma\ \vec{v}$.

There are nine cases, corresponding to whether $\underline{m}$ and m are in the arenas A, B or C. Each gives a different interaction sequence, and requires a separate proof. Some cases have special subcases, when $\underline{m}$ or one of the intermediate moves is a root of some tree. All cases, although sometimes long, are very similar to the proofs already given for identities and projections. ■

Remark 4.4.4 The definition of EXAC strategy is really intended for arenas of the form U^n. We can define the full subcategory $\mathbb{XU}$, of which the objects are pairs (A, X) with $A = U^n$ for some n and X a finitely-branching r.e. arena with n trees, which has the same cartesian closed structure as $\mathbb{XA}$.

4.5 The Model $\mathcal{D}_{\mathrm{XA}}$

Recall the single-tree arenas U and M defined in Example 2.1.2. The former is "maximal", in that it is (countably) infinitely branching and infinitely deep. The latter is "minimal", consisting of one a single node.

Let us write U_0 for the object (U, M) of $\mathbb{XA}$, and U_1 for the object $U_0 \Rightarrow U_0$. The reader may wish to verify that, for example, the concrete representation of U_1 is given by $(U, \langle\{\varepsilon, \langle 1\rangle\}\rangle)$.

Recall also the EXAC strategy η_1 from Example 4.1.2. We repeat the definition for convenience: the EAC strategy part is the same as the strategy id_U (since $U = U \Rightarrow U$ this does define an EAC strategy over U). The copycat threshold is zero at every P-view except the initial P-view, when it is 1. Thus the economical form is given by:

$$\begin{array}{rcl} \varepsilon & \mapsto & (1,0,1,-1) \\ \langle 1 \rangle & \mapsto & (2,1,0,0) \end{array}$$

It is routine to check that η_1 specifies two morphisms of $\mathbb{XA}$, $f : U_0 \to U_1$ and $g : U_1 \to U_0$. (In fact it is the case that these morphisms could equally be specified as the EXAC strategies which have EAC strategy part id_U, and the least copycat thresholds to make them explicit in the necessary sub-arenas to be morphisms of that type. The definition of morphisms by EAC strategy and "least threshold function" to make them explicit in the appropriate sub-arenas seems to be a recurring theme.)

We can now show that f and g form a retraction from U_1 into U_0: we know that that EAC strategy part of f, g, id_{U_0} and id_{U_1} are all the same as id_U, so that same holds for $f\,;g$ and $g\,;f$. Thus it remains only to check threshold functions. Simple applications of the Composition Algorithm show that the thresholds of both $f\,;g$ and $g\,;f$ are 1 at the minimal P-view, and zero elsewhere. The same holds for id_{U_1}, whereas id_{U_0} has threshold which is zero at every P-view. Hence, as EXAC strategies and thus as morphisms in $\mathbb{XA}$,

$$g\,;f = \mathrm{id}_{U_1}, f\,;g \neq \mathrm{id}_{U_0}.$$

Thus allows us to identify a λ-algebra $\mathcal{M}(\mathbb{XA}, U_0, f, g)$, which we shall write as $\mathcal{D}_{\mathrm{XA}}$. To distinguish the denotation of a term as an EAC strategy in $\mathcal{D}_{\mathrm{EAC}}$ (which is the same as the denotation in $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$), from the denotation as an EXAC strategy in this model we write it $[\![s]\!]^{\mathrm{XA}}$.

Remark 4.5.1 The λ-algebra $\mathcal{D}_{\mathrm{XA}}$ could equivalently have been presented as $\mathcal{M}(\mathbb{XU}, U_0, f, g)$ (see Remark 4.4.4).

We will want to lift some results about EAC strategies and the model $\mathcal{D}_{\mathrm{EAC}}$ to EXAC strategies and the model $\mathcal{D}_{\mathrm{XA}}$. We make that connexion precise with the following.

Theorem 4.5.2 Let $E : \mathbb{XA} \to \mathbb{A}_{\mathrm{EAC}}$ be given on objects by $E((A, X)) = A$ and on morphisms by $E(\langle \sigma, t_\sigma \rangle) = \sigma$. Then

(i) E is a full, exact cartesian closed, functor, and

(ii) E preserves the reflexive object and the retraction morphisms.

Hence for any term s and valuation ρ, $[\![s]\!]_\rho = E([\![s]\!]^{\mathrm{XA}}_\rho)$.

Proof (i) is straightforward verification (note that for example the projections

in $\mathbb{XA}$ are defined to be precisely the projections in $\mathbb{A}_{\mathrm{EAC}}$, along with the least copycat threshold to make them explicit in the appropriate subarena).

To show that E is full suppose that $\sigma : A \to B$ is an arrow of $\mathbb{A}_{\mathrm{EAC}}$. (Suppose that neither A nor B are equal to the empty arena E, these are special cases dealt with below). Then σ is an EAC strategy on $A \Rightarrow B$ and hence has some recursive valid threshold function t_σ, so $\langle \sigma, t_\sigma \rangle$ is an EXAC strategy over $A \Rightarrow B$. Furthermore by Lemma 4.4.1 this EXAC strategy is $(M \Rightarrow M)$-explicit hence it is a morphism $(A, M) \to (B, M)$ of $\mathbb{XA}$. (The special cases are as follows: if A is the empty arena E then the EXAC strategy above will certainly be $(E \Rightarrow M)$-explicit, symmetrically for $B = E$, and if $A = B = E$ then $\sigma = \bot$, which is $(E \Rightarrow E)$-explicit.)

For (ii), clearly $E(U_0) = U$, $E(f) = \mathit{Fun}$ and $E(g) = \mathit{Gr}$. It remains to observe that the λ-algebra $\mathcal{M}(\mathbb{C}, R, \mathit{Fun}, \mathit{Gr})$ (where R is a reflexive object via Fun and Gr in the CCC $\mathbb{C}$, for the definition see Section 1.6) is completely determined by cartesian closed structure of $\mathbb{C}$, along with R, Fun and Gr. ■

Corollary 4.5.3 $\mathcal{D}_{\mathrm{XA}}$ is sensible. $[\![s]\!]^{\mathrm{XA}} = \bot$ if and only if s is unsolvable.

Note that the functor E restricts to a functor $E' : \mathbb{XU} \to \mathbb{U}_{\mathrm{EAC}}$ with the same properties.

4.6 Böhm Trees in Variable-Free Form and Exact Correspondence

In Chapter 3 we showed that the denotation of term, in the models $\mathcal{D}$, $\mathcal{D}_{\mathrm{REC}}$ and $\mathcal{D}_{\mathrm{EAC}}$, has a very close connexion with the term's Nakajima tree. Here we will show that the same connexion exists between the denotation of a term in $\mathcal{D}_{\mathrm{XA}}$ and the Böhm tree of the term.

Recall that the *Böhm tree* of term s, written $\mathrm{BT}(s)$ is given (informally) by the following: if s is unsolvable then $\mathrm{BT}(s) = \bot$. If s has HNF $\lambda x_1 \ldots x_n . y s_1 \ldots s_m$ then

$$\mathrm{BT}(s) = \quad \lambda x_1 \ldots x_n . y$$
$$\mathrm{BT}(s_1) \quad \cdots \quad \mathrm{BT}(s_m)$$

Thus each node of the Böhm tree has a finite number of abstractions, a head variable, and a finite number of children. We consider Böhm tree modulo α-conversion, so that the the following information is sufficient to describe a Böhm tree of a closed term: for each node we have numbers specifying the number of abstractions and children, and information to describe where head variable was abstracted in the tree in the same was as we did for Nakajima trees in Section 3.2.

We encode this information into a *variable-free form* in the following manner:

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

Similarly the tree $\{p\}^n$, for $n \in \mathbb{N}_0$, is labelled identically except that firstly the node at the root (i, r, t, o) is first relabelled to $(i, r, t, o - n)$, and then nodes of depth d are relabelled as follows:

(i) those labelled (i, d, t, o) are relabelled $(i + n, d, t, o)$;

(ii) those labelled $(i, d + 1, t, o)$ for $i \leq n$ are relabelled $(n - i + 1, d, t, o)$;

(iii) those labelled $(i, d + 1, t, o)$ for $i > n$ are relabelled $(i - n, d + 1, t, o)$.

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

$$\begin{aligned}
\mathrm{VFBT}_\Delta(s) &= \bot, && \text{the empty tree, for unsolvable } s.\\
\mathrm{VFBT}_\Delta(\lambda x_1 \ldots x_n.s) &= \{\mathrm{VFBT}_{\Delta\cdot\langle x_1,\ldots,x_n\rangle}(s)\}^n, && \text{if } s \text{ is of the form } v_j s_1 \ldots s_m.\\
\mathrm{VFBT}_\Delta(v_j s_1 \ldots s_m) &= && (j, 1, m, m)\\
& && \quad\diagup\qquad\diagdown\\
& && \mathrm{VFBT}_\Delta(s_1)^* \qquad \mathrm{VFBT}_\Delta(s_m)^*\\
& && \text{where } \Delta = \langle v_k, \ldots, v_1\rangle \text{ (note the reverse order).}
\end{aligned}$$

Lemma 4.6.1 The encoding of head variables in VFBT matches that in VFF. Precisely, for any term s with free variables within Δ and any sequence $\vec{v} \in \mathbb{N}^*$ at which (the labelling function of) $\mathrm{VFBT}_\Delta(s)$ is defined we have $\mathrm{VFBT}_\Delta(s)(\vec{v}) = (i, r, t, o)$ implies $\mathrm{VFF}_\Delta(s)(\vec{v}) = (i, r)$.

Furthermore, for any k, $\{\mathrm{VFBT}_\Delta(s)(\vec{v})\}^k = (i, r, t, o)$ implies $\{\mathrm{VFF}_\Delta(s)(\vec{v})\}^k = (i, r)$.

Proof Trivial when one compares the definitions of VFBT and VFF. ■

This definition is at first sight rather opaque, and indeed it could have been stated in a clearer fashion but that this would have complicated Theorem 4.6.5 below. The effect of the definition is illustrated by the following lemma, which is analogous to Lemma 3.2.2.

Lemma 4.6.2 Let s be a term with all free variables occurring within $\Delta = \langle v_k, \ldots, v_1\rangle$. Construct the Böhm tree of s, and rename all the bound variables so that if $\vec{a} \in \mathbb{N}^*$ codes a node of $\mathrm{BT}(s)$ then the i^{th} abstracted variable at this

node is $x_i^{\vec{a}}$. Let this renamed Böhm tree have labelling function A, and consider $\mathrm{VFBT}_\Delta(s)$ also as a labelling function.

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

(i) If $\vec{a} \notin \mathrm{dom}(A)$ then $\mathrm{VFBT}_\Delta(s)$ is unlabelled or undefined at $\vec{a}$,

(ii) If $A(\vec{a}) = \lambda x_1^{\vec{a}} \ldots x_n^{\vec{a}}.v_j$, and the node coded by $\vec{a}$ has m children, then $\mathrm{VFBT}_\Delta(s) = (j, p+1, m, m-n)$.

(iii) If $A(\vec{a}) = \lambda x_1^{\vec{a}} \ldots x_n^{\vec{a}}.x_j^{\langle a_1,\ldots,a_{p-r}\rangle}$, and the node coded by $\vec{a}$ has m children, then $\mathrm{VFBT}_\Delta(s) = (j, r, m, m-n)$.

Proof Entirely similar to the proof of Lemma 3.2.2. ■

Example 4.6.3 One may check that $\mathrm{VFBT}(I)$ and $\mathrm{VFBT}(1)$ are:

$$(1,0,0,-1) \qquad \text{and} \qquad \begin{array}{c}(1,0,1,-1)\\ \mid \\ (2,1,0,0)\end{array}$$

The node $\lambda xy.x$, in the Böhm tree of 1, corresponds to the node of VFBT(1) labelled $(1,0,1,-1)$, which is so labelled because the head variable is the first abstracted variable zero levels up the tree (namely x), the node has one child, and the number of abstractions at this level is $1-(-1)=2$.

The following result will be useful in what follows.

Lemma 4.6.4 If $\sigma : (A,X) \to U_1$ is a morphism in $\mathbb{XA}$, for any object (A,X), then $\sigma\,;g : (A,X) \to U_0$ is the same EXAC strategy as σ.

If $\tau : (A,X) \to U_0$ is a morphism in $\mathbb{XA}$, for any object (A,X), then $\tau\,;f : (A,X) \to U_1$ is the same EXAC strategy as τ, unless the threshold and offset of τ at the minimal P-view, t and o, satisfy $t - o = |A|$. In this case $\tau\,;f$ is the same EXAC strategy as τ except that the threshold at the minimal P-view is $t+1$.

Proof Recall that both f and g are the EXAC strategy η_1, which is the same as the EXAC strategy id_{U_0}, except that the threshold at the minimal P-view is 1 rather than 0. Hence we can use the proof that identities work as required in Theorem 4.4.2 to show that $\sigma\,;f$ and $\tau\,;g$ are the same EXAC strategies as σ and τ respectively, except at the minimal P-view.

It remains to examine the minimal P-view. Suppose that the first move of σ is in the arena A rather than U, with copycat threshold and offset t_σ and o_σ. Then, in the notation of the Composition Algorithm,

$$\begin{array}{ll} t_1' = 1 + |A| & o_1' = -1 + |A| \\ t_2' = t_\sigma & o_2' = o_\sigma \\ T_1 = 1 + |A| & O_1 = -1 + |A| \\ T_2 = \max(1 + |A| + o_\sigma, t_\sigma) \stackrel{*}{=} t_\sigma & O_2 = o_\sigma - 1 + |A| \end{array}$$

Recall that $U_1 = (U, M \Rightarrow M)$. Then $(*)$ is because σ must be $(X \Rightarrow (M \Rightarrow M))$-explicit, hence $t_\sigma - o_\sigma \geq |A| + 1$.

Thus $\mathbf{t} = t_\sigma$ and $\mathbf{o} = o_\sigma$, i.e. the threshold and offset of $\sigma\,;f$ at the minimal P-view match that of σ. A similar calculation applies when the first move of σ is in the arena U.

The same figures occur in the calculation of $\tau\,;g$ at the minimal P-view, except that in this case we only know that τ, as a morphism $(A, X) \to U_0$, must be $(X \Rightarrow M)$-explicit, so that $t_\tau - o_\tau \geq |A|$. When equality holds, $\mathbf{t} = t_\tau + 1$, when it is a strict inequality, $\mathbf{t} = t_\tau$ as for σ. ■

We use this to show a powerful connexion between the denotations of terms in $\mathcal{D}_{\mathrm{XA}}$ and Böhm trees.

Theorem 4.6.5 (Exact Correspondence for $\mathcal{D}_{\mathrm{XA}}$) If $s \in \Lambda$ with free variables in $\Delta = \langle v_k, \ldots, v_1 \rangle$ then $[\![s]\!]^{\mathrm{XA}}_\Delta = \{\mathrm{VFBT}_\Delta(s)\}^k$ when the former is considered as an EXAC strategy in economical form and the latter as a labelling function.

In particular for closed terms s, $[\![s]\!]_\varepsilon = \mathrm{VFBT}_\varepsilon(s)$

Proof We show by induction on the length of $\vec{\alpha}$, for all terms s and contexts Δ simultaneously, that

(i) $\vec{\alpha} \cdot i \in \mathrm{dom}([\![s]\!]^{\mathrm{XA}}_\Delta)$ if and only if $\vec{\alpha} \cdot i \in \mathrm{dom}(\mathrm{VFBT}_\Delta(s))$. The latter is trivially equal to $\mathrm{dom}(\{\mathrm{VFBT}_\Delta(s)\}^k)$.

(ii) If $[\![s]\!]^{\mathrm{XA}}_\Delta(\vec{\alpha}) = (i, r, t, o)$ and $\{\mathrm{VFBT}_\Delta(s)\}^k(\vec{\alpha}) = (i', r', t', o')$ then $i' = i, r' = r$ and $t' = t$.

We will be able to use the Exact Correspondence Theorem for $\mathcal{D}$, together with Lemma 4.6.1 and Theorem 4.5.2 to prove that $i' = i$ and $r' = r$, and then show $t' = t$ by considering the composition algorithm. In view of Remark 4.1.1 this will ensure that $o' = o$ too (in the base case $o' = o$ comes for free, but the proof of the inductive step is easier without having to consider offsets).

Base Case: If s is unsolvable then both $[\![s]\!]^{\mathcal{M}}_\Delta$ and $\mathrm{VFBT}_\Delta(s)$ are everywhere undefined.

Otherwise s has a head normal form $\lambda x_1 \ldots x_n.y s_1 \ldots s_m$. Then in the notation above $t' = m$ and $o' = m - n - k$.

We must return to the definition of $[\![-]\!]^{\mathrm{XA}}$ to discover the copycat thresholds and offsets.

$$[\![s]\!]^{\mathrm{XA}}_\Delta = \underbrace{\Lambda(\cdots\Lambda(\Lambda}_{n\ \Lambda\text{'s}}([\![y s_1 \ldots s_m]\!]^{\mathrm{XA}}_\Gamma)\,;g)\,;g\cdots)\,;g$$

where $\Gamma = \Delta \cdot \langle x_1, \ldots, x_n \rangle$. Since $\Lambda(\sigma)$ is the same EXAC strategy as σ, and by Lemma 4.6.4 so is $\sigma\,;g$, this has the same thresholds and offsets as $[\![y s_1 \ldots s_m]\!]^{\mathrm{XA}}_\Gamma$. Let $V = \underbrace{U_0 \times \cdots \times U_0}_{k+n \text{ times}}$.

$$[\![ys_1 \dots s_m]\!]^{\mathrm{XA}}_{\Gamma} = (\Pi^{\Gamma}_y \bullet [\![s_1]\!]^{\mathrm{XA}}_{\Gamma}) \cdots \bullet [\![s_m]\!]^{\mathrm{XA}}_{\Gamma}$$

We will show, by induction on m, that the threshold of this strategy at the minimal P-view is m and the offset is $m - n - k$. For convenience we also include the fact that the first P-move of the composite strategy is played in the arena V in the induction hypothesis.

Case $m = 0$*:* The strategy is just Π^{Γ}_y. Depending on y, this is just one of the projections $V \to U_0$, which we know has threshold 0 and offset $-n - k$ at the minimal P-view. We also know that the first P-move is played in the arena V.

Inductive Case Suppose that we have a strategy $\sigma : V \to U_0$ with threshold m and offset $m - n - k$ at the minimal P-view, and which makes the first P-move in the arena V. Then for any strategy $\tau : V \to U_0$,

$$\sigma \bullet \tau = \langle \sigma \, ; f, \tau \rangle \, ; \mathrm{eval}_{U_0,U_0}.$$

First examine $\sigma \, ; f$ — we are in the special case of Lemma 4.6.4, so that the threshold of $\sigma \, ; f$ is $m + 1$ and the offset $m - n - k$.

It is now simple to use the Composition Algorithm to examine the threshold and offset of $\langle \sigma \, ; f, \tau \rangle \, ; \mathrm{eval}_{U_0,U_0}$ at the minimal P-view. Since we know that the first P-move of σ is in the arena V, there is one intermediate move, which is the root of $V \Rightarrow U_1$ at which σ is to play. Thus the strategy τ is irrelevant for this calculation and in the notation of the Composition Algorithm the resulting threshold and offset is calculated as follows:

$$\begin{array}{lcllcl}
t'_1 & = & 1 + k + n & o'_1 & = & -1 + k + n \\
t'_2 & = & m + 1 & o'_2 & = & m - n - k \\
T_1 & = & 1 + k + n & O_1 & = & -1 + k + n \\
T_2 & = & \max(m + 1, 1 + k + n + m - n - k) & O_2 & = & -1 + k + n + m - n - k \\
 & = & m + 1 & & = & m - 1
\end{array}$$

Hence $\mathbf{t} = m + 1$ and $\mathbf{o} = m - 1 - (n + k) + 2 = m + 1 - n - k$. One can also see that the first move of the composition is in the arena V, which completes the inductive step of this claim.

This completes the proof that the threshold and offset of $[\![s]\!]^{\mathrm{XA}}_{\Delta}$ at the minimal P-view are m and $m - n - k$ respectively.

This shows that $\langle i \rangle \in \mathrm{dom}([\![s]\!]^{\mathrm{XA}}_{\Delta})$ if and only if $1 \leq i \leq m$ and s_i is solvable. On the other hand, $\langle i \rangle \in \mathrm{dom}(\mathrm{VFBT}_{\Delta}(s))$ if and only if $1 \leq i \leq m$ and s_i is solvable. This completes the proof of (i).

Suppose that $[\![s]\!]^{\mathrm{XA}}_{\Delta}(\varepsilon) = (i, r, t, o)$ and $\{\mathrm{VFBT}_{\Delta}(s)\}^k(\varepsilon) = (i', r', t', o')$. Now Lemma 4.6.1 means that $\{\mathrm{VFF}_{\Delta}(\varepsilon)\}^k(\varepsilon) = (i', r')$. On the other hand, Theorem 4.5.2 means that $[\![s]\!]_{\Delta}(\varepsilon) = (i, r)$, and the Exact Correspondence Theorem for $\mathcal{D}_{\mathrm{EAC}}$ gives that $i' = i$ and $r' = r$.

Finally, by the definition of VFBT, $t' = m = t$ and $o' = m - n - k = o$, completing the base case of the outer induction.

Inductive Step: Again if s is unsolvable then both functions are everywhere undefined, so assume that s has HNF $\lambda x_1 \ldots x_n.ys_1 \ldots s_m$ and again write $\Gamma = \Delta \cdot \langle x_1, \ldots, x_n \rangle$. We assume the results (i) and (ii) for each of the terms $s_1, \ldots, s_m$, each with the context Γ, for all sequences $\vec{\alpha}$ of length up to l.

Let $\vec{\alpha}$ be any sequence of length l. Take $1 \leq j \leq m$. We show that (i) and (ii) hold for the term s and context Δ, for the sequence $j \cdot \vec{\alpha}$. We already know it holds for the sequence ε, and (also by the base case) that the domain of both functions is contained in the set $\{j \cdot \vec{\alpha} \mid 1 \leq j \leq m, \vec{\alpha} \in \mathbb{N}^*\}$. This will therefore establish the inductive step.

Suppose that $[\![s]\!]^{\mathrm{XA}}_{\Delta}(j \cdot \vec{\alpha}) = (i, r, t, o)$ and $\{\mathrm{VFBT}_{\Delta}(s)\}^k(j \cdot \vec{\alpha}) = (i', r', t', o')$. We know by result (i) of the inductive hypothesis that one is defined if and only if the other is. Lemma 4.6.1 means that $\{\mathrm{VFF}_{\Delta}(s)\}^k(j \cdot \vec{\alpha}) = (i', r')$. On the other hand, Theorem 4.5.2 means that $[\![s]\!]_{\Delta}(j \cdot \vec{\alpha}) = (i, r)$, and the Exact Correspondence Theorem for $\mathcal{D}_{\mathrm{EAC}}$ gives that $i' = i$ and $r' = r$. We next show that $t' = t$, completing the proof of (ii).

Now by the definition of VFBT,

$$\begin{array}{rl} & \{\mathrm{VFBT}_{\Delta}(s)\}^k(j \cdot \vec{\alpha}) = (i', r', t', o') \\ \text{if and only if} & \{\mathrm{VFBT}_{\Delta}(s)\}^{(k+n)}(s_j)(\vec{\alpha}) = (i'', r'', t', o'') \end{array}$$

for some irrelevant numbers i'', r'', o'' (this is simple to verify). Then by the inductive hypothesis $[\![s_j]\!]^{\mathrm{XA}}_{\Gamma}(\vec{\alpha}) = (i'', r'', t', o'')$

With this fact in hand we examine $[\![s]\!]^{\mathrm{XA}}_{\Delta}$, aiming to calculate $[\![s]\!]^{\mathrm{XA}}_{\Delta}(j \cdot \vec{\alpha})$ from this fact.

As we found in the base case, $[\![s]\!]^{\mathrm{XA}}_{\Delta} = [\![ys_1 \ldots s_m]\!]^{\mathrm{XA}}_{\Gamma} = (\Pi^{\Gamma}_y \bullet [\![s_1]\!]^{\mathrm{XA}}_{\Gamma}) \bullet \cdots \bullet [\![s_m]\!]^{\mathrm{XA}}_{\Gamma}$, which with the $\bullet$'s decoded is

$$\langle \cdots \langle \langle \Pi^{\Gamma}_y ; f, [\![s_1]\!]^{\mathrm{XA}}_{\Gamma} \rangle ; \mathrm{eval} ; f, [\![s_2]\!]^{\mathrm{XA}}_{\Gamma} \rangle ; \mathrm{eval} ; f \cdots, [\![s_m]\!]^{\mathrm{XA}}_{\Gamma} \rangle ; \mathrm{eval}$$

where eval is eval_{U_0,U_0}. What follows is not a rigorous analysis, as such a thing would be impossible to typeset as well as extremely tedious.

We already know, from proof of the Exact Correspondence Theorem for $\mathcal{D}_{\mathrm{EAC}}$, that if the first O-move made against this strategy is j then the result of this multiple composition is to copy moves made by and against σ_j from the components where they are hidden into ones where they are not; this composite strategy makes moves which are (a small translation of) those of σ_j. What is important is that between visible moves, except between the initial move and the first P-move, all of the intermediate moves are *not* roots of the arenas they occur in (the reader is invited to draw a picture — on a large piece of paper — and demonstrate this for themselves). The reason this is important is because for Π^{Γ}_y and eval_{U_0,U_0} the

copycat thresholds and offsets are always zero except at the initial P-view (this is very simple to check).

Thus when we work out the threshold and offset of $[\![s]\!]^{\mathrm{XA}}_{\Delta}$ at the P-view coded by $j \cdot \vec{\alpha}$ using the composition algorithm and the fact that the threshold and offset of $[\![s_j]\!]^{\mathrm{XA}}_{\Gamma}$ at the P-view coded by $\vec{\alpha}$ are t' and o'', the calculation will be either of the form

$$\begin{array}{llcll} t'_1 = 0 & o'_1 = 0 & & t'_1 = t' & o'_1 = o'' \\ t'_2 = 0 & o'_2 = 0 & & t'_2 = 0 & o'_2 = 0 \\ & \vdots & & & \vdots \\ t'_p = 0 & o'_p = 0 & & t'_p = 0 & o'_p = 0 \\ t'_{p+1} = t' & o'_{p+1} = o'' & & t'_{p+1} = 0 & o'_{p+1} = 0 \end{array} \quad \text{or}$$

depending on which component the visible moves appear in. In the first case $T_1 = T_2 = \cdots = T_p = 0$ and $\mathbf{t} = T_{p+1} = \max(t', o'') = t'$, In the second case, $T_1 = t'$ so $t' = T_2 = \cdots = T_{p+1} = \mathbf{t}$.

Thus in either case we have shown that the copycat threshold of $[\![s]\!]^{\mathrm{XA}}_{\Delta}$ at the P-view coded by $j \cdot \vec{\alpha}$ is t', but by assumption it is also t. Hence $t = t'$.

Finally, we show (i) as follows: $j \cdot \vec{\alpha} \cdot i \in \mathrm{dom}([\![s]\!]^{\mathrm{XA}}_{\Delta})$ if and only if $\vec{\alpha} \cdot i \in \mathrm{dom}([\![s_j]\!]^{\mathrm{XA}}_{\Gamma})$, this is because of the way the multiple composition which defines $[\![s]\!]^{\mathrm{XA}}_{\Delta}$ copies the moves of s_j after the first P-move j. But $\vec{\alpha} \cdot i \in \mathrm{dom}([\![s_j]\!]^{\mathrm{XA}}_{\Gamma})$ if and only if $\vec{\alpha} \cdot i \in \mathrm{dom}(\mathrm{VFBT}_{\Gamma}(s_j))$ (by (i) of the inductive hypothesis) if and only if $j \cdot \vec{\alpha} \cdot i \in \mathrm{dom}(\mathrm{VFBT}_{\Delta}(s))$ by the definition of VFBT.

This completes the inductive step of the outermost induction. ■

Corollary 4.6.6 For closed terms s and t,

$$[\![s]\!]^{\mathrm{XA}} \subseteq [\![t]\!]^{\mathrm{XA}} \iff \mathrm{BT}(s) \subseteq \mathrm{BT}(t).$$

The order on $\mathcal{D}_{\mathrm{XA}}$ is inclusion of EXAC strategies, namely inclusion of both EAC strategy part and threshold function. The order on Böhm trees is inclusion of labelling function, modulo renaming of bound variables, which amounts to inclusion of variable-free form. Thus the local structure of $\mathcal{D}_{\mathrm{XA}}$ is the λ-theory $\mathcal{B}$.

Example 4.6.7 Applying the Exact Correspondence Theorem to the variable-free forms of the Böhm trees we looked at in Example 4.6.3, we can deduce that the economical forms of $[\![I]\!]$ and $[\![1]\!]$ are, as we hoped, the EXAC strategies η_0 and η_1 described in Example 4.1.2.

As with the model $\mathcal{D}_{\mathrm{EAC}}$, the Exact Correspondence Theorem allows us to prove the powerful result of universality holds for $\mathcal{D}_{\mathrm{XA}}$.

Theorem 4.6.8 $\mathcal{D}_{\mathrm{XA}}$ is a universal λ-algebra.

Proof The proof is an easier analogue of that of Lemma 3.5.2. Suppose that

$\sigma' = \langle \sigma, t_\sigma \rangle$ is an EXAC strategy on U (to be an element of the model it must be a member of the homset $\mathrm{Hom}_{\mathbb{XA}}(\mathbf{1}, U_0)$). Let f be the economical form of the innocent function of σ, and write t_v and o_v for the threshold and offset of σ at the P-view coded by $\vec{v}$.

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

$$\begin{array}{l} \varepsilon \in X \\ \text{if } \vec{v} \in X \text{ then } 1 \leq i \leq t_v \implies \vec{v} \cdot i \in X. \end{array}$$

This has the property that the domain of the economical form of σ' is a subset of X (this is straightforward to verify).

We define the labelling function of Böhm-like tree A as follows. The domain of 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:

(i) If $\vec{v} \notin \mathrm{dom}(f)$ then $A(\vec{v})$ is undefined (the partially-labelled tree has label $\perp$ at this node).

(ii) 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 $n = t_v - o_v$.

Now there are no free variables in the Böhm-like tree A which is also clearly r.e., hence by Theorem 1.5.4 there is some term s with $\mathrm{BT}(s) = A$.

Now we prove that $\mathrm{VFBT}_\varepsilon(s)$, as a labelling function, is the same as the economical form of σ'. Take any sequence $\vec{v} \in X$. If f is not defined on this sequence, neither is the economical form of σ', but on the other hand A is not labelled at $\vec{v}$ by (i) above. If $f(\vec{v}) = (i, r)$ then we know that the node in X coded by $\vec{v}$ has t_v children, hence similarly for A, and also that $A(\vec{v}) = \lambda x_1^{\vec{v}} \ldots x_n^{\vec{v}}.x_i^{\langle v_1, \ldots, v_{p-r} \rangle}$, where $n = t_v - o_v$, by (ii) above. Thus $\mathrm{VFBT}_\varepsilon(s)(\vec{v}) = (i, r, t_v, o_v)$ by Lemma 4.6.2. On the other hand, by definition the economical form of σ' also maps $\vec{v}$ to (i, r, t_v, o_v).

Finally, we appeal to the Exact Correspondence Theorem for $\mathcal{D}_{\mathrm{XA}}$, which shows that $[\![s]\!]_\varepsilon$ has economical form given by the labelling function of $\mathrm{VFBT}_\varepsilon(s)$, which by the above is the economical form of σ'. ■

The universality result for $\mathcal{D}_{\mathrm{XA}}$ tells us immediately about its extensionality properties. Recall that in Section 3.7 we showed that $\mathcal{D}_{\mathrm{EAC}}$ was order-extensional and hence extensional.

The same is not true for $\mathcal{D}_{\mathrm{XA}}$; note that $[\![I]\!]^{\mathrm{XA}} \neq [\![1]\!]^{\mathrm{XA}}$. Since for all terms s and t, $Ist = 1st$, we can be sure that $\mathcal{D}_{\mathrm{XA}}$ is not extensional, but in fact it is not even weakly extensional (hence not a λ-model).

A simple proof uses the strong correspondence between the $\mathcal{D}_{\mathrm{XA}}$ and the closed term model of the λ-calculus. The general result is as follows.

Lemma 4.6.9 If $\langle \mathcal{A}, \bullet, [\![-]\!]_- \rangle$ is a universal and weakly-extensional λ-algebra then $[\![I]\!] = [\![1]\!]$.

Proof Consider $s = x$ and $t = \lambda y.xy$. For any $a \in \mathcal{A}$ we have a closed term u

such that $[\![u]\!] = a$. Hence $[\![s]\!]_{(x:=a)} = [\![u]\!]$ and $[\![t]\!]_{(x:=a)} = [\![\lambda y.uy]\!]$.

But since u is closed we know that $u \equiv \lambda z.v$ for some term v, hence $[\![\lambda y.uy]\!] = [\![\lambda y.v[z := y]]\!] = [\![\lambda z.v]\!] = [\![u]\!]$. So by weak extensionality, we must have $[\![I]\!] = [\![\lambda x.s]\!] = [\![\lambda x.t]\!] = [\![1]\!]$. ■

Corollary 4.6.10 $\mathcal{D}_{\mathrm{XA}}$ is not weakly extensional.

Chapter 5

Conclusions

5.1 Connexions with Other Work

We turn to the work of Di Gianantonio *et al.*, who have recently been looking at history-free game models of untyped λ-calculus.

A game model is introduced in [DFH99]. It has interesting parallels with the work presented here — it is a modification of the history-free game setting used in [AJM94] with the distinction between questions and answers removed. As usual with history-free games, there is a decomposition of the function space into a linear function space and exponential, and some extra work needs to be done to construct a cartesian closed category.

Rather than giving a reflexive object directly, however, the authors use a complete partial ordering and take the limit of a chain generated by iterating a functor. Thus the construction of the reflexive object is reminiscent of the construction of Scott's D_∞ models [Sco69]. (The use of iteration to find fixed points is described in [McC98], and indeed it was this technique that Abramsky and McCusker used for their history-free game model of the lazy λ-calculus). By introducing an indexing for terms, the authors use results of labelled reduction and approximate normal forms (see [Wad76]) to show directly that the local structure of all game models arising in this way is $\mathcal{H}^*$.

In a follow-up paper [DF98] a type assignment system is given, and a connexion between the denotation of a term and the set of types which it can be assigned is made. This is some sort of correspondence result, but it does not seem at all related to our Exact Correspondence Theorem. Its significance is difficult to judge, although it does present a finitary description of a denotational semantics for the untyped λ-calculus.

A third paper on this subject, [DF00], shows how different reflexive objects may be constructed, using modifications of the techniques of [DFH99]. It is shown that such reflexive objects fall into three classes, which must give models with local structure

equal to $\mathcal{B}$, $\mathcal{H}^*$, or the theory of equality of Lévy-Longo trees $\mathcal{L}$, depending on the properties of the retraction morphisms.

The work presented in these three papers is in a different style to that presented here, and not only because it uses the history-free style of games instead of the innocent approach. The use of chains of iterates to construct reflexive objects, as opposed to our setting in which the reflexive object is simple to define, seems to make the model rather more cumbersome to work with. Local structure results are proved using approximations to terms and strategies; there is no Exact Correspondence Theorem and, importantly, no universality result. The models constructed are in fact λ-algebras and not λ-models (although the authors refer to them as λ-models) because they do not have extensionality properties.

The characterization of precisely which equational theories can be constructed in this history-free setting is interesting. However the construction of models of theories other than $\mathcal{H}^*$ is not related to our construction of the model $\mathcal{D}_{\mathrm{XA}}$, which arises from a quite different category to $\mathbb{A}_{\mathrm{EAC}}$, so we would not expect it to apply to our style of game. The parallel construction in our setting would instead be the use of different retraction morphisms in the category $\mathbb{A}$. That construction of a model which invalidates η-conversion is possible in this way has been known to the author for some time. Take, for example, morphisms Fun' and Gr' defined to be the appropriate type and by the strategies which are given by $[\![\lambda x.xx]\!]$ and $[\![\lambda xy.x]\!]$ respectively: it is routine to check that these still form a retraction, but it is not an isomorphism. One can then form a λ-algebra $\mathcal{M}(\mathbb{A}, U, Fun', Gr')$ which does not validate η-conversion. However it is not clear that its local structure is precisely $\mathcal{B}$, and the Exact Correspondence is broken by the use of retraction morphisms which shuffle moves as well as types.

5.2 Further Directions

We conclude with some areas in which the thesis is limited, and further research which is indicated.

Observational Quotient; Towards an Algebraic Definition of EAC

One limitation of this work is that the definition of an EAC strategy is rather pulled from thin air; it is more or less a translation of Theorem 1.5.4 into the language of economical forms of innocent strategies. Although it makes for a very satisfactory model (and the non-uniqueness of copycat thresholds leads us to consider the EXAC strategies) it would be nice to arrive at the EAC model in an algebraic way. Copycat strategies are a recurring theme in all styles of game semantics, and similar ideas

also appear in other parts of computer science. We believe that there are strong connexions between copycat strategies and *data independence*, see e.g. [LR96]. A better understanding of the EAC condition would be very worthwhile for these reasons. An algebraic characterization might also simplify the proof that EAC strategies compose.

A first attempt would be to fix a notion of observable and take the *observational quotient* (see Chapter 3 of [HO00]) of the category $\mathbb{A}_{\mathrm{REC}}$ (or just of the model $\mathcal{D}_{\mathrm{REC}}$). A suitable definition of the latter might be:

Definition The partial order $\preceq$ on $\mathcal{D}$ is defined by $\sigma \preceq \tau$ if for all $\rho \in \mathcal{D}$,

$$\rho \bullet \sigma : \varepsilon \mapsto 1 \quad \Longrightarrow \quad \rho \bullet \tau : \varepsilon \mapsto 1$$

and we write $\sigma \approx \tau$ if $\sigma \preceq \tau$ and $\tau \preceq \sigma$.

Then the following are true: $\preceq$ is a congruence with respect to $\bullet$; $\sigma \subseteq \tau$ implies $\sigma \preceq \tau$; $\perp$ is the unique least element; the definition is not altered if we restrict attention to quantification over compact elements ρ (and *a fortiori* to recursive elements); for any λ-terms s and t, if $[\![s]\!] \neq [\![t]\!]$ then $[\![s]\!] \not\approx [\![t]\!]$ (and hence no two different EAC elements of $\mathcal{D}$ are equivalent).

Although the EAC strategies all inhabit different equivalence classes of $\approx$, there are some elements not equivalent to any EAC strategy. If one takes the pseudo-term "$\lambda x_1 x_2 x_3 \ldots \bullet x_2 x_1 x_4 x_3 x_6 x_5 x_8 x_7 \ldots$", and gives it a denotation consistent with the Exact Correspondence Theorem, it is fairly clear that such a strategy is observationally inequivalent to any EAC strategy.

Now the observational quotient is of interest for a variety of reasons in its own right but it also suggests the question of whether there is a simple, hopefully less syntactic, restriction one can make on strategies such that there is precisely one EAC strategy in each equivalence class of the observational quotient of what is left. Taking our ideas on infinitary λ-calculus into consideration — see below — it appears that the definition of EAC is actually imposing two conditions on terms: finiteness and extensionality. Can the conditions be factored?

Failing that, perhaps there is a separate algebraic characterisation of the EAC strategies which fully reflects their parametric nature.

Connexions with Infinitary Lambda Calculus

We return to the observational quotient of $\mathcal{D}_{\mathrm{REC}}$. Since $\mathcal{D}_{\mathrm{REC}}$ is arguably the most natural game model of untyped computation, it is of interest to study its properties.

Consideration of strategies which are observationally inequivalent to EAC strategies, such as that which intuitively denotes "$\lambda x_1 x_2 x_3 \ldots \bullet x_2 x_1 x_4 x_3 x_6 x_5 x_8 x_7 \ldots$", leads us towards an infinitary version of the λ-calculus.

Nakajima introduced a brand of infinite terms when presenting the η-expanded trees in [Nak75] which we used for the Exact Correspondence Theorem. Perhaps this approach can be used for our aims, although his method (each term is a sequence of terms which approximate it) is based on the D_∞ model rather than syntax.

Infinitary λ-calculi are a topic of current research interest. In [KKSdV97] 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 [Ber96], but the focus is on infinite term re-writing and again the set of infinite terms seems not to be what we want. Infinite re-writing would certainly have to play a part in our language, and we would hope that infinite trees such as the Nakajima trees would be the normal forms with respect to a notion of (possibly transfinite) head reduction.

John Longley has commented that the space of increasing sequences of Böhm trees might correspond to the recursive innocent strategies. However we would prefer to identify a new language with infinitary terms, a suitable equational theory, and which is universally modelled by the observational quotient of $\mathcal{D}_{\mathrm{REC}}$. This language might have some interesting features (solvability is not equivalent to the existence of head normal forms, for a start). Finding such a language has the important justification of providing a (hopefully) compact syntax to reason about innocent strategies. This is something which has hitherto been lacking: certainly the innocent functions or economical forms are sufficient to describe strategies, but not in any elegant way.

Models of Other Theories

It should be possible to modify these game models to capture other λ-theories. In view of the work of Di Gianantonio *et al.* we should perhaps start with equality of Lévy-Longo Trees of [Lév75] and [Lon83] (the theory generated in this way is called $\mathcal{L}$). By slightly weakening the definition of EXAC strategies, to allow a threshold and/or offset at a node where the innocent function is undefined, it seems likely that the EXAC model can be refined into a universal model with equational theory equal to $\mathcal{L}$. Hopefully there will be no extra complications in defining a CCC for these strategies, such as we found in moving from EAC to EXAC strategies.

On the other hand, we could perhaps define a new set of undefined strategies $\bot_0, \bot_1, \ldots, \bot_\infty$ which have slightly special composition rules and are intended to denote the unsolvables of order $0, 1, \ldots, \infty$. This could be placed in the EAC setting, and we would expect to arrive at a model which validates η-conversion. Presumably there is some theory which is to $\mathcal{L}$ as $\mathcal{H}^*$ is to $\mathcal{B}$; hopefully the model would be a universal λ-algebra inducing this theory. It would certainly be of interest to find a game model of some theory other than $\mathcal{H}^*$, $\mathcal{B}$ or $\mathcal{L}$, in view of the results of [DF00].

It has to be said that these ideas make for rather contrived models, even more so than the Böhm tree model presented, since we are tagging our semantic models with thinly disguised syntax in order to induce the required theories. An algebraic definition of EAC would probably also lead to algebraic definitions for other models.

Much more speculatively, we wonder if it would be possible to attack the problem of describing a syntax-free model of the "plain" theory λ, a problem which has hitherto remained unsolved. A reason for optimism is that game semantics is quite intensional in nature; on the other hand, our gut feeling is that λ is just *too* intensional even for game models. If there are limitations, it would be very interesting to find out why. The work of [DF00] suggests that the composition algorithm of all game models makes them "strongly biased towards head reduction".

Realizability Over Game Models

We have seen that $\mathcal{D}_{\mathrm{EAC}}$ and $\mathcal{D}_{\mathrm{XA}}$ provide very satisfactory models of the untyped λ-calculus. However it is $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$ which are the compellingly "natural" innocent game models of untyped computation, and we can examine them as combinatory algebras in their own right. Of particular interest are the *realizability* categories over these combinatory algebras.

The subject of realizability first arose in mathematical logic, with the work of Kleene in [Kle45], but has since grown into a useful tool for denotational semantics. One takes a fairly simple structure — usually a (partial) combinatory algebra — and uses the elements of it to "realize" a richer structure. A survey of the use of realizability for denotational semantics can be found in Longley's thesis [Lon95]. The realizability construction we are interested in is the category of *modest sets* generated by a combinatory algebra, which is equivalent to the perhaps more familiar category of *partial equivalence relations* which appears in [Gir72] and derives from [Kre59]. For any partial combinatory algebra $\mathcal{A}$, the category of modest sets over $\mathcal{A}$, written $\mathrm{Mod}(\mathcal{A})$, is cartesian closed, regular, and has a natural numbers object which lives in a hierarchy of high order functional objects (we call the hierarchy of functional objects generated by the lifted NNO $\mathbb{N}_\bot$ and exponentiation the *simple type structure*).

In the original study of realizability the combinatory algebra used for realizers was

the set of Turing machines (encoded as Kleene's first model K_1) but the work of Phoa ([Pho90] and [Pho92]) suggests that insight may be gained by allowing other (partial) combinatory algebras. In particular, although all such structures will be untyped and Turing complete, it turns out that they may generate realizability categories with quite different properties. In his thesis [Lon95], Longley lays out a very general setting for constructing realizability categories, and amongst other things shows how different denotational models of PCF arise from realizability constructions carried out with different models of untyped computation. For example, take the term combinatory algebra Λ/β and Kleene's first model K_1. In [Lon95] it is shown that the simple type structures in $\mathrm{Mod}(\Lambda/\beta)$ and $\mathrm{Mod}(K_1)$ are not the same; in particular the latter contains "parallel" elements not present in the former. It turns out that the $\mathrm{Mod}(K_1)$ describes a fully abstract model of PCF extended by a parallel-conditional operator described in [Plo77]. This reflects the fact that K_1 as a partial combinatory algebra performs "parallel" computations. Since the λ-calculus performs only "sequential" computations, similarly to PCF, this leads to the *Longley-Phoa conjecture*: for any sensible λ-theory T, the interpretation of PCF in $\mathrm{Mod}(\Lambda/T)$ is fully abstract. (The conjecture was made implicitly for the case $T = \mathcal{B}$ in [Pho91]; Longley made it explicit and showed that the conjecture is independent of T — at least in the cases when $\mathcal{H} \subseteq T \subseteq \mathcal{B}$ — in [Lon95]).

In [Lon98] Longley used van Oosten's decision tree combinatory algebra $\mathcal{B}$ (from [vO97]) and realizability to define the *Sequentially Realizable Functionals* (SR), and an analogue which has effectiveness built in ($\mathrm{SR}_{\mathit{eff}}$). (We will refer to this combinatory algebra as $\mathfrak{B}$ to avoid confusion with the Böhm tree λ-theory.) One of Longley's aims is to study higher type computability; it turns out that there are many candidates for the class of effective functionals, including the PCF-definable functionals, $\mathrm{SR}_{\mathit{eff}}$, the functionals which inhabit the Scott domains, and so on. Longley describes a number of alternative ways in which $\mathrm{SR}_{\mathit{eff}}$ arises, apart from realizability over $\mathfrak{B}$: the extensional collapse of the effective sequential algorithms, the extensional collapse of the programming languages μPCF and PCF+catch, the type structure in a presheaf category, the closed term model of PCF extended with a combinator called H, and so on. Longley argues that, due to the ubiquity of this class, $\mathrm{SR}_{\mathit{eff}}$ provides a good definition of the sequential effective functionals.

Longley continues this programme in [Lon99]. Of particular interest here is that he describes a game combinatory algebra $\mathcal{A}$, due to Abramsky (the style is that of the history-free strategies of [AJM94]). $\mathcal{A}$ is shown quite easily to be equivalent to $\mathfrak{B}$ (in the sense that there are PCA homomorphisms between them which are iso) and hence the simple type structure in $\mathrm{Mod}(\mathcal{A})$ is SR. $\mathcal{A}$ also has an effective analogue which realizes $\mathrm{SR}_{\mathit{eff}}$ in the same way. Interestingly, these combinatory algebras also have "well-bracketed" subalgebras, and the well-bracketed effective version of $\mathcal{A}$ realizes a fully-abstract model of PCF.

This leads to our first conjecture: that the same holds for innocent game models. In particular, that the category of modest sets over $\mathcal{D}$ (respectively $\mathcal{D}_{\mathrm{REC}}$) has SR

($\mathrm{SR}_{\mathit{eff}}$) as simple type structure, and that $\mathcal{D}_{\mathrm{REC}}$ has a subalgebra, which arises by imposing a reading of question and answer onto moves and enforcing well-bracketing, which realizes a universal and fully-abstract model of PCF.

This would be of interest for a number of reasons. Firstly it gives more weight to Longley's claim that SR and $\mathrm{SR}_{\mathit{eff}}$ are natural classes of functionals. It would show that the same higher order functionals are realized by both history-free and innocent game models. Potentially the result could be of more interest than merely paralleling the history-free models, since it is certain that $\mathcal{D}$ is not as easily equivalent to $\mathfrak{B}$ as $\mathcal{A}$ is — the latter equivalence is really rather trivial. In fact it seems quite likely that $\mathcal{D}$ is not equivalent to $\mathfrak{B}$ at all, although the conjecture is that they have the same simple type structure. This would highlight an essential difference between the history-free and innocent game models. Finally, we expect that the proof of the conjecture would involve a better understanding of $\mathcal{D}$ and $\mathcal{D}_{\mathrm{REC}}$, and their observational quotient.

This conjecture appears to be difficult to prove. Some initial work suggests that something stronger may be true: that the simple structure in $\mathcal{D}$ is that same as in the observational quotient of $\mathcal{D}$ and that the latter is equivalent to $\mathfrak{B}$ (similarly for $\mathcal{D}_{\mathrm{REC}}$ and the effective analogue of $\mathfrak{B}$).

Proving facts about type structures in realizability models can be very hard. We would hope to use the methodology of *applicative morphisms* developed in [Lon95] to attack the problem. Applicative morphisms allow one to prove equivalence between PCAs by defining a translation between them and showing that certain maps on terms of the two PCAs are realizable. Details can be found in Chapter 2 of [Lon95]. An "applicative equivalence" implies equivalence of PCAs, and there is also the weaker property of "applicative inclusion" between two PCAs which (as long as the natural numbers objects behave properly) implies that their simple type structure is the same. We hope to exhibit an applicative inclusion between $\mathcal{D}$ and its observational quotient, and an applicative equivalence between the latter and $\mathfrak{B}$.

The work of Laird [Lai98] is relevant here. We have already noted that Laird studies the same class of strategies that we use for our models, where the well-bracketing condition is removed, although for the different purpose of constructing a fully-abstract of μPCF. In Chapter 5 of [Lai98], Laird studies the observational quotient of the category of unbracketed innocent strategies (equivalent to $\mathbb{A}$) and shows, by an involved syntactic proof, that this is equivalent the category of sequential algorithms [BC82]. This implies that the extensional collapse of the category is equivalent to SR. It appears that the first part of our conjecture follows from this work. However we would prefer to prove it directly because Laird's proof is rather involved, and we expect to gain considerable insight into innocent strategies in doing so.

A particular angle that has arisen in early work on this conjecture is that some

strategies are "redundant" in that they play moves which gain them no extra information, assuming that they are playing against an innocent opponent. If the definition of redundancy can be made precise, we expect that there is precisely one irredundant strategy in each equivalence class of the observational quotient.

Another conjecture in [Lon98] is that "any reasonable class of strategies that is 'sufficiently unconstrained' [sufficiently many of the conditions on strategies which make for the universal fully-abstract PCF model are relaxed] would yield SR as its extensional collapse". Investigation of this, at least in the innocent game setting, would be worthwhile.

Finally, there is some hope that these game models may shed light on the difficult questions raised by Longley in [Lon95] and subsequently. For example, it remains unproven that the simple type structures in $\mathrm{Mod}(\Lambda/\mathcal{B})$ and $\mathrm{Mod}(\Lambda/\mathcal{H}^*)$ are equal. Perhaps the universal game models for these theories will provide some help. Longley comments that "it seems likely that the [Longley-Phoa conjecture] could be made much easier by a different choice of PCA". Using the techniques of applicative morphisms, the conjecture becomes one of relating the well-bracketed strategies of [HO00] with the EAC strategies developed here. Although certainly still difficult, this question may prove more tractable than the original.

A New Composition Algorithm for Bohm Trees

The Exact Correspondence Result and Composition Algorithm for EXAC strategies give a new method for finding the Böhm tree of the composition of two terms whose Böhm trees are known. The standard way to find $\mathrm{BT}(st)$ from $\mathrm{BT}(s)$ and $\mathrm{BT}(t)$ is to take the finite approximants to the trees to be composed and use these to form finite approximants to the composition (see [Bar84, §18.3]). Instead one could translate the two Böhm trees into EXAC strategies and use the algorithms described in this work to find the composite strategy, and translate back into a Böhm tree. This indirect method has an advantage over the standard method: one can find a particular node of the composite tree in a way which only examines those nodes of the composed trees which are relevant. This "lazy" method for composing Böhm trees might have practical uses.

Index

Bibliography

[Abr90] S. Abramsky. The lazy λ-calculus. In D. A. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison Wesley, 1990.

[AJ92] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. In R. Shyamsundar, editor, *Foundations of Software Technology and Theoretical Computer Science*, pages 291–301. Springer-Verlag, 1992.

[AJM94] 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.

[AM95a] 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.

[AM95b] S. Abramsky and G. A. McCusker. Games for recursive types. In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, *Theory and Formal Methods of Computing 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*. Imperial College Press, 1995.

[AM97] S. Abramsky and G. McCusker. Game semantics. Lecture notes for 1997 Marktoberdorf Summer School, 1997.

[AM99] S. Abramsky and G. A. McCusker. Call-by-value games. In Mogens Nielsen and Wolfgang Thomas, editors, *Computer Science Logic: 11th International Workshop Proceedings*, volume 1414 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[AO93] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.

[Bar77] H. P. Barendregt. The type free lambda calculus. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 1091–1132. North-Holland, Amsterdam, 1977.

[Bar84] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1984.

[BC82] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20(3):265–321, 1982.

[Ber96] 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.

[Bla92] A. Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56, 1992.

[Böh68] C. Böhm. Alcune proprietà delle forme β-η-normali nel λ-K calcolo. *Pubblicazioni dell' Istituto per le Applicazioni del Calcolo*, 696, 1968.

[Chu32] A. Church. A set of postulates for the foundation of logic. *Annals of Mathematics (2)*, 33:346–366, 1932. Second paper with same title in Vol. 33, pages 839–864, of same journal.

[Cro93] R. L. Crole. *Categories For Types*. Cambridge mathematical textbooks. Cambridge University Press, 1993.

[Cur30] H. B. Curry. Grundlagen der kombinatorischen Logik. *Amer. J. Math.*, 52:509–536, 789–834, 1930.

[Cut80] N. J. Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, 1980.

[dB72] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation. *Indag. Math.*, 41:381–392, 1972.

[DF98] P. Di Gianantonio and G. Franco. A type assignment system for the game semantics. *Proceedings of the Italian Conference on Theoretical Computer Science 98, World Scientific*, pages 37–47, 1998.

[DF00] P. Di Gianantonio and G. Franco. The fine structure of game lambda-models. In *Proceedings of the Conference on Foundations of Software Technology and Theoretical Computer Science FSTTS'00*, Lecture Notes in Computer Science. Springer-Verlag, 2000. to appear.

[DFH99] 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.

[Fel86] W. Felscher. Dialogues as a foundation for intuitionistic logic. In D. Gabbay and F. Guenthener, editors, *Handbook of Philosophical Logic, Volume III*, pages 341–372. D. Reidel Publishing Company, 1986.

[Gir72] J.-Y. Girard. *Interprétation functionelle et élimination des coupures de l'arithmétique d'order superior*. PhD thesis, Paris, 1972.

[Har00] R. S. Harmer. *Games and Full Abstraction for Non-Deterministic Languages*. PhD thesis, University of London, 2000.

[HO00] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. To appear in *Information and Computation*, 2000.

[Hug00] D. H. D. Hughes. *Games and Full Completeness for System F*. PhD thesis, University of Oxford, 2000.

[HY97] K. Honda and N. Yoshida. Game theoretic analysis of call-by-value computation. Personal Communcation, 1997.

[Hyl76] 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.

[KKSdV97] J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. Infinitary lambda calculus. *Theoretical Computer Science*, 175(1):93–125, 1997.

[Kle45] S. C. Kleene. On the interpretation of intuitionistic number theory. *Journal of Symbolic Logic*, 10, 1945.

[KNO99] A. D. 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.

[KNO00] A. D. Ker, H. Nickau, and C.-H. L. Ong. A universal innocent model of the Böhm tree lambda theory. Technical Report TR-10-00, Oxford University Computing Laboratory, 2000.

[KNO01] A. D. Ker, H. Nickau, and C.-H. L. Ong. Innocent game models of untyped λ-calculus. To appear in *Theoretical Computer Science*, 2001.

[Koy82] C. P. J. Koymans. Models of the lambda calculus. *Information and Control*, 52(3):306–332, 1982.

[Kre59] G. Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland, Amsterdam, 1959.

[Lai98] J. Laird. *A Semantic Analysis of Control*. PhD thesis, University of Edinburgh, 1998.

[Lév75] J. J. Lévy. An algebraic interpretation of the λ-β-k-calculus and a labelled λ-calculus. In C. Böhm, editor, *Proceedings of the Symposium on λ-calculus and Computer Science Theory*, volume 37 of *Lecture Notes in Computer Science*, pages 147–165. Springer-Verlag, Berlin, 1975.

[LL78] P. Lorenzen and K. Lorenz. *Dialogische Logik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1978.

[Lon83] G. Longo. Set-theoretic models of λ-calculus: Theories, expansions, isomorphisms. *Annals of Pure and Applied Logic*, 24:153–188, 1983.

[Lon95] J. R. Longley. *Realizability Toposes and Language Semantics*. PhD thesis, University of Edinburgh, 1995.

[Lon98] J. R. Longley. The sequentially realizable functionals. Technical Report ECS-LFCS-98-402, LFCS, Division of Informatics, University of Edinburgh, 1998. Submitted to Annals of Pure and Applied Logic.

[Lon99] J. R. Longley. Realizability models for sequential computation. Notes of a talk given at the 1998 APPSEM workshop, Pisa, 16 September, 1999.

[LR96] R. S. Lazić and A. W. Roscoe. Using logical relations for automated verification of data independent CSP. In *Proceedings of the Workshop on Automated Formal Methods (Oxford, U.K., June 1996), to appear in Electronic Notes on Theoretical Computer Science*, 1996.

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, Berlin, 1971.

[McC98] G. A. McCusker. *Games and Full Abstraction for a Functional Metalanguage with Recursive Types*. Distinguished Dissertations in Computer Science. Springer-Verlag, 1998.

[Nak75] R. Nakajima. Infinite normal forms for the λ-calculus. In C. Böhm, editor, *Proceedings of the Symposium on λ-calculus and Computer Science Theory*, volume 37 of *Lecture Notes in Computer Science*, pages 62–82. Springer-Verlag, Berlin, 1975.

[Nic96] Hanno Nickau. *Hereditarily Sequential Functionals: A Game-Theoretic Approach to Sequentiality.* Shaker-Verlag, 1996. Dissertation, Universität Gesamthochschule Siegen.

[Ong88] C.-H. L. Ong. *The Lazy Lambda Calculus: An Investigation into the Foundations of Functional Programming.* PhD thesis, University of London, 1988.

[Pho90] W. K.-S. Phoa. *Domain Theory and Realizability Topses.* PhD thesis, University of Cambridge, 1990. Available as CST-82-91, Department of Computer Science, University of Edinburgh.

[Pho91] W. K.-S. Phoa. From term models to domains. In *Proceedings of Theoretical Aspects of Computer Software, Sendai*, volume 526 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[Pho92] W. K.-S. Phoa. An introduction to fibrations, topos theory, the effective topos and modest sets. Technical Report ECS-LFCS-92-208, Department of Computer Science, University of Edinburgh, 1992.

[Plo72] G. D. Plotkin. A set-theoretical definition of application. School of Artificial Intelligence, Memo MIP-R 95, University of Edinburgh, 1972.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–225, 1977.

[Plo78] G. D. Plotkin. $\mathbb{T}^\omega$ as a universal domain. *J. Comput. Syst. Sci.*, 5:223–257, 1978.

[Sch24] M. Schönfinkel. über die Bausteine der mathematischen Logik. *Math. Annalen*, 92:305–316, 1924.

[Sco69] D. S. Scott. Models for the λ-calculus. Unpublished manuscript, 1969.

[Sco74] D. S. Scott. The language LAMBDA (abstract). *Journal of Symbolic Logic*, 39:425–427, 1974.

[Sco93] D. S. Scott. A type-theoretical alternative to CUCH, ISWIM and OWHY. *Theoretical Computer Science*, 121:411–440, 1993.

[vO97] J. van Oosten. A combinatory algebra for sequential functionals of finite type. Technical Report 996, University of Utrecht, 1997. To Appear in Proc. Logic Colloquium, Leeds.

[Wad71] C. P. Wadsworth. *Semantics and Pragmatics of the λ-calculus*. PhD thesis, University of Oxford, 1971.

[Wad76] 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.