Linear Type Theories, Semantics and Action Calculi

Andrew Graham Barber
LFCS, Department of Computing Science,
Edinburgh University

Doctor of Philosophy
University of Edinburgh
1997

To my wife, for her trust

Abstract

In this thesis, we study linear type-theories and their semantics. We present a general framework for such type-theories, and prove certain decidability properties of its equality. We also present intuitionistic linear logic and Milner's action calculi as instances of the framework, and use our results to show decidability of their respective equality judgements.

Firstly, we motivate our development by giving a split-context logic and type-theory, called *dual intuitionistic linear logic* (DILL), which is equivalent at the level of term equality to the familiar type-theory derived from intuitionistic linear logic (ILL). We give a semantics for the type-theory DILL, and prove soundness and completeness for it; we can then deduce these results for the type-theory ILL by virtue of our translation.

Secondly, we generalise DILL to obtain a general logic, type-theory and semantics based on an arbitrary set of *operators*, or general natural deduction rules. We again prove soundness and completeness results, augmented in this case by an initiality result. We introduce Milner's *action calculi*, and present example instances of our framework which are isomorphic to them. We extend this isomorphism to three significant *higher-order* variants of the action calculi, having functional properties, and compare the induced semantics for these action calculi with those given previously.

Thirdly, motivated by these functional extensions, we define *higher-order* instances of our general framework, which are equipped with functional structure, proceeding as before to give logic, type-theory and semantics. We show that the logic and type-theory DILL arise as a higher-order instance of our general framework. We then define the *higher-order extension* of any instance of our framework, and use a Yoneda lemma argument to show that the obvious embedding from an instance into its higher-order extension is conservative. This has the corollary that the embeddings from the action calculi into the higher-order action calculi are all conservative, extending a result of Milner.

Finally, we introduce relations, a syntax derived from proof-nets, for our general framework, and use them to show that certain instances of our framework, including some higher-order instances, have decidable equality judgements. This immediately shows that the equalities of DILL, ILL, the action calculi and the higher-order action calculi are decidable.

Acknowledgements

I am hugely grateful to my supervisor, Gordon Plotkin, for his insight and ability to communicate it, his encouragement, and for his detailed comments at each stage of my thesis.

I also thank Philippa Gardner, my second supervisor, for her attention, her enthusiasm and for pointing me in the direction of many interesting problems.

Other than my supervisors, I have benefited greatly during my time at Edinburgh from the insight of many people here, particularly Masahito Hasegawa, John Power, Alex Simpson, Paul-Andre Mellies and Marcello Fiore, whom I thank, and from the good-humour of my office partner Thomas Schreiber.

Finally, I thank my wife for all her support, encouragement and proof-reading–her efforts have immeasurably improved this thesis and my time in Edinburgh.

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Andrew Graham Barber*
LFCS, Department of Computing Science,
Edinburgh University)

Table of Contents

Chapter 1 Introduction **4**

1.1 Motivating Sketch . . . 5
1.2 Background . . . 11
1.3 Development . . . 24
1.4 Chapter Summary . . . 29
1.5 Historical Notes . . . 31

Chapter 2 An Alternative Formulation of ILL **32**

2.1 The Logic DILL . . . 32
2.2 The Type Theory DILL($\mathbb{C}$) . . . 35
2.3 The Type Theory ILL($\mathbb{C}$) . . . 42
2.4 Relating DILL and ILL . . . 45

Chapter 3 The Semantics of DILL **60**

3.1 The Interpretation . . . 60
3.2 Soundness . . . 65
3.3 The Term Model . . . 68
3.4 Completeness . . . 78
3.5 ILL and Other Models . . . 79

Chapter 4 Linear Type Theories **81**

4.1 Introduction . . . 81
4.2 The Generalised Logic . . . 84
4.3 The Typing System Lin($\mathbb{O}$) . . . 87
4.4 The Type Theory Lin($\mathbb{O}, \mathbb{A}$) . . . 91

Chapter 5 The Semantics of Lin($\mathbb{O}, \mathbb{A}$) **95**

5.1 The Interpretation . . . 95
5.2 Soundness . . . 99
5.3 The Term Model . . . 101

5.4 Completeness 107
5.5 Initiality 109
5.6 Output Naturality 114

Chapter 6 Action Calculi and Extensions **120**
6.1 Action Calculi 121
6.2 Generalised Linear Type Theory 124
6.3 The Translations 126
6.4 Extensions 134
6.5 Semantics 143

Chapter 7 Higher-Order Type Theories **148**
7.1 Higher-Order Type Theories 148
7.2 Higher-Order Semantics 153
7.3 Conservativity 159
7.4 Corollaries 165
7.5 Action Calculi 168
7.6 Linear Logic 170

Chapter 8 Normal Forms for $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ **178**
8.1 Proof Nets 178
8.2 Relations 185
8.3 Relational Equality 196
8.4 Terms to Relations 199
8.5 Relations to Terms 204

Chapter 9 Normal Forms and Decidability **215**
9.1 Proof-Nets 215
9.2 The F-Equalities 216
9.3 Deciding the Equality 218
9.4 The Higher-Order Case 227
9.5 Corollaries 236

Chapter 10 Conclusions **239**
10.1 Results 239
10.2 Extending $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ 241
10.3 Aims and Objectives 246
10.4 Final Remarks 249

Appendix A Basic Definitions **250**

A.1 Syntactic Preliminaries 250

A.2 Categorical Definitions 251

Bibliography **255**

Index to Notation **265**

Index to Definitions **267**

Chapter 1

Introduction

The central concept in this thesis is that of *linearity*. We can say that linearity is an attribute of systems in which unrestricted duplication of computations or information is not allowed. There are many practical situations in which this is the case—computations with side effects, where duplicating the computation also duplicates the side-effect, manipulating large data-structures, where a duplication may bear an unacceptable computational cost, or simply efficient implementations where duplicating computation is to be avoided except where necessary. Areas as diverse as optimal λ-calculus implementations, the study of process calculi and denotational semantics can all utilise linearity in this broad sense.

Although such systems have been studied for many years, a new and striking development occurred when Girard introduced *linear logic* [Gir87]. Unlike conventional intuitionistic and classical logics, linear logic is built on the underlying idea that assumptions should be treated linearly, which is to say that an assumption should not be used twice in a deduction. Starting from this basis, Girard built a complete classical linear logic, which, just as in the conventional case, has both minimal and intuitionistic fragments.

Since the introduction of linear logic, many researchers have used it as an underlying logic for systems which exhibit linearity in our general sense, for example in the study of optimal λ-calculus reductions [GAL92a], in the analysis of implementation issues such as garbage collection, references and others [Wad90, GH90] and to control interference in imperative languages [OTPT95]. It has become the logic of choice for such applications in much the same way as conventional logic underpins a huge range of theory in computer science. In the case of conventional logics, one large section of applications are those which are based on conventional intuitionistic logic as a typing system for a syntax of some sort. A variety of systems can be typed in this way, and the general principles underlying such typings are well understood.

In this thesis, we present a first step towards a similar unifying programme for systems with linearity. We give a general operator theory, logic, and type-theory over a linear underlying structure, and show how it can be used to present a number of typing systems exhibiting linearity. We also give a semantics for our theory, and prove some general results about it.

As an introduction, we give a motivating summary in a simple setting, provide brief tutorials and summaries of related work on the concepts we work with, and then outline the work contained in the thesis. A chapter summary can be found in section 1.4.

We assume a basic understanding of elementary intuitionistic logic, its type theory, simple category theory, and the various relationships of a familiar triangle, particularly including the Curry-Howard correspondence between logic and type-theory:

$$
\begin{tikzcd}
& \text{Logic} \arrow[dl, leftrightarrow, "\text{[GLT89]}"'] \arrow[dr, leftrightarrow, "\text{[Lam89, Sol90]}"] & \\
\text{Type Theory} \arrow[rr, leftrightarrow, "\text{[LS86]}"'] & & \text{Category Theory [Mac71]}
\end{tikzcd}
$$

1.1 Motivating Sketch

Consider the typed λ-calculus. Although it forms the basis of many developments in computer science, from functional programming languages to foundational semantics, it is very frequently augmented with additional constructs, for example those for pairing, sums, or other data types, or that for recursion. In the case of each such extension the definition of terms must be extended, and new typing rules given, and as a result of these extensions, many simple definitions and results must be at least rechecked. Such a procedure not only repeats work, but also guarantees that many slightly differing presentations of each alternative calculus exist, each with its own development. We might try to avoid this by using a general theory of operators, due to Aczel [Acz80, Acz78].

Minimal Intuitionistic Logic

We start by considering minimal intuitionistic logic over a given set of formulae or proposition letters (ranged over by $A, B \ldots$) and given by the abstract grammar:

$$A ::= p \in \mathsf{P} \mid A \to A$$

with the rules:

$$\frac{}{\Gamma, A \vdash A}\ (Ax)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (Abs) \qquad\qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (App)$$

where the turnstile $\vdash$ is a relation holding between sets of formulae (denoted $\Gamma, \Delta \ldots$) and formulae. We call the axiom rule, along with the (admissible) cut and weakening rules:

$$\frac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash B}\ (Cut) \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ (Weak)$$

the *structural rules* of the logic. Note that the rules for the arrow connective occur as an *introduction-elimination pair* (or I/E-pair); the abstraction introduces the $\to$ and the application rule eliminates it. Such I/E-pairs are characteristic of natural deduction.

Now there are many possible ways to strengthen this logic, by adding new connectives and rules for those connectives. Let us try to give a general schema for such natural deduction rules. As a first attempt, we might say that an arbitrary rule, given a number of deductions $\Gamma \vdash A_i$ for $i = 1 \ldots r$, gives us a deduction $\Gamma \vdash B$ for some new conclusion B:

$$\frac{\Gamma \vdash A_1 \quad \ldots \quad \Gamma \vdash A_r}{\Gamma \vdash B}$$

This is a reasonable attempt, as all but one of the rules of minimal intuitionistic logic are instances of this schema, as are the rules for pairing amongst others. However, the abstraction rule is not; this is because an assumption is discharged in this rule. Given a deduction $\Gamma, A \vdash B$, we end up with a deduction in which A is no longer an assumption. Hence we could refine our schema; given a number of deductions $\Gamma, \Delta_i \vdash A_i$ for $i = 1 \ldots r$, we have a deduction $\Gamma \vdash B$ in which all the assumptions Δ_i for each $i = 1 \ldots r$ have been discharged. This gives a general rule:

$$\frac{\Gamma, \Delta_1 \vdash A_1 \quad \ldots \quad \Gamma, \Delta_r \vdash A_r}{\Gamma \vdash B}$$

Now all the rules of minimal intuitionistic logic and many others arise as instances of this schema.

It is important to note, however, that some do not, because of implicit or explicit side-conditions. One common situation in which a side-condition is imposed is when a modality is used; for example in the presence of a unary connective $\Box$, we might have an introduction rule:

$$\frac{\Gamma \vdash A}{\Gamma \vdash \Box A}$$

only where each formula in Γ has the form $\Box B$ for some B.

Another (apparent!) possible behaviour which is not captured by our rule schema is that where an assumption is *introduced* in a deduction, as for example in an alternative formulation of the $\rightarrow$-elimination rule:

$$\frac{\Gamma \vdash A \rightarrow B}{\Gamma, A \vdash B}$$

Although rules of this form are not instances of our rule schema, in the presence of the structural rules we can show that they are interderivable with the instances of our rule schema. In the case of this alternative rule for $\rightarrow$, the appropriate instance is exactly the normal $\rightarrow$-elimination rule.

Given our general description of a natural deduction rule, we can see that any particular instance is characterised by the number r, the sets of formulae Δ_i which are discharged, the conclusions A_i and the final conclusion B'. We might then write all this information as the *arity* of the rule:

$$\frac{(\vec{A}_1)B_1 \quad \ldots \quad (\vec{A}_r)B_r}{B'}$$

where the set of elements in the sequence $\vec{A}_i$ is Δ_i for each $i = 1 \ldots r$. As an example, consider the sequent-style presentation of the natural deduction rule for choice:

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ (\vee E)$$

where formulae A and B are discharged from the assumptions. This would then have the arity:

$$\frac{()A \vee B \quad (A)C \quad (B)C}{C}$$

We can characterise any extension of minimal intuitionistic logic with rules of this general form by giving just the basic formulae, and the arities of the rules. Together, we call these two bits of information a *signature* for the logic. In fact, as we shall see, the most important feature of minimal intuitionistic logic is not the rules for the connective $\rightarrow$, but rather the admissible structural rules. Given these structural rules, we can express the underlying behaviour of any connective such as $\rightarrow$ in the arity of its rules.

Using the definition of proofs in a logic with general rules, we can also define a sensible notion of equivalence between proofs based on eliminating certain redundant proof sequences. For example, in minimal logic, any proof which ends:

$$\dfrac{\Gamma \vdash A \quad \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}}{\Gamma \vdash B}$$

should be equivalent to the proof obtained from the admissible cut rule applied to the deductions $\Gamma \vdash A$ and $\Gamma, A \vdash B$. We could extend such a notion of proof-equivalence to the extensions of minimal logic with instances of our rule schema by allowing arbitrary equivalences over proofs of the extended logic. However, this is more easily done in the framework of the type-theory we will present in the next subsection.

The Typed λ-calculus

We now recall that via the Curry-Howard correspondence, proofs of minimal logic can be represented by the typed λ-calculus, using the annotations:

$$\dfrac{}{\Gamma, x{:}A \vdash x{:}A}\ (Ax)$$

$$\dfrac{\Gamma, x{:}A \vdash t{:}B}{\Gamma \vdash \lambda x{:}A.t{:}A \to B}\ (Abs) \qquad\qquad \dfrac{\Gamma \vdash t{:}A \to B \quad \Gamma \vdash u{:}A}{\Gamma \vdash tu{:}B}\ (App)$$

Moreover, in the example of the previous subsection, the $(\vee - E)$ rule is often annotated with a $\mathsf{cases}_{A,B,C}$-construct

$$\dfrac{\Gamma \vdash t{:}A \vee B \quad \Gamma, x{:}A \vdash u{:}C \quad \Gamma, y{:}B \vdash v{:}B}{\Gamma \vdash \mathsf{cases}_{A,B,C}\ t\ \mathsf{of}\ x\ \mathsf{in}\ u\ \mathsf{or}\ y\ \mathsf{in}\ v{:}C}$$

which acts as a choice operator. In general, we can give an annotation for an instance of our rule schema having arity

$$\dfrac{(\vec{A}_1)B_1 \quad \ldots \quad (\vec{A}_r)B_r}{B'}$$

in an extension of the typed λ-calculus by adding a term construct $O((\vec{x})t, \ldots, (\vec{x})t)$ with the introduction rule

$$\dfrac{\Gamma, \vec{x}_1{:}\vec{A}_1 \vdash t_1{:}B_1 \quad \ldots \Gamma, \vec{x}_r{:}\vec{A}_r \vdash t_r{:}B_r}{\Gamma \vdash O((\vec{x}_1)t_1, \ldots, (\vec{x}_r)t_r){:}B'}$$

In this way we can build a typed λ-calculus over a signature of formulae and rules with associated arities. Many common extensions of typed λ-calculus arise as instances of this, including pairing constructs, choice constructs and recursion operators.

Having defined a typing system based on the logic having as rules arbitrary instances of our rule schema, we can give a notion of *equality judgement* over such a typing system. An equality judgement takes the form $\Gamma \vdash t = u : A$ for terms $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$, and asserts that the proofs denoted by the terms t and u are equivalent. Equality judgements are commonly built inductively as congruences over axiomatic equalities which arise from proof equivalences. These often take one of the general forms:

$$\dfrac{\dfrac{\vdots}{I}}{E} \equiv \vdots \qquad \dfrac{\dfrac{\vdots}{E}}{I} \equiv \vdots$$

where the I and E denote the introduction and elimination rules for a particular connective. We shall call these respectively the β- and η-*equivalences* for a given connective. They give rise respectively to β- and η-*equalities* on the proof terms.

In minimal logic, the β- and η-equivalences for the $\rightarrow$ connective are:

$$\dfrac{\dfrac{\Gamma, x{:}A \vdash t{:}B}{\Gamma \vdash \lambda x{:}A.t{:}A \rightarrow B}\,(\rightarrow I) \qquad \Gamma \vdash u{:}A}{\Gamma \vdash (\lambda x{:}A.t)u{:}B}\,(\rightarrow E) \qquad\qquad \dfrac{\dfrac{\Gamma, x{:}A \vdash t{:}B \quad \Gamma, x{:}A \vdash x{:}A}{\Gamma, x{:}A \vdash tx{:}B}\,(\rightarrow E)}{\Gamma \vdash \lambda x{:}A.(tx){:}A \rightarrow B}\,(\rightarrow I)$$

$$\equiv \qquad\qquad\qquad\qquad \equiv$$

$$\dfrac{\Gamma, x{:}A \vdash t{:}B \quad \Gamma \vdash u{:}A}{\Gamma \vdash t\{u/x\}{:}B}\,(Cut) \qquad\qquad \Gamma \vdash t{:}A \rightarrow B$$

under the condition on the second of these that x does not occur free in t (and where $t\{u/x\}$ stands for the *substitution* of u for x in t).

The equivalences then yield the familiar $\beta\eta$-equalities of the $\rightarrow$ type:

$$(\beta)\ \ \Gamma \vdash (\lambda x{:}A.t)u = t\{u/x\}{:}B \qquad (\eta)\ \ \Gamma \vdash \lambda x{:}A.(tx) = t{:}A \rightarrow B\ (x \text{ not free in } t)$$

We can extend the equality judgement in the case of logics built on general instances of our rule schema by allowing arbitrary axiomatic equality judgements over terms of the general term calculus over the logics.

Cartesian Closed Categories

We turn to the question of semantics. It is well-known that minimal intuitionistic logic has models which are cartesian closed categories. In these, a proof $\Gamma \vdash A$ is modelled as a morphism $[\![\Gamma]\!] \rightarrow [\![A]\!]$, with a suitable interpretation of the basic types. We can see that any rule of the logic takes a certain number of morphisms from the appropriate hom-sets, and returns another. Being more precise, given a rule having arity

$$\frac{(\vec{A}_1)B_1 \quad \dots \quad (\vec{A}_r)B_r}{B'}$$

to interpret it in a c.c.c. $\mathcal{C}$ we would require for each possible context Γ a function

$$\alpha_\Gamma : \prod_{i=1\dots r} \mathcal{C}([\![\Gamma]\!] \times [\![\vec{A}_i]\!], [\![B_i]\!]) \to \mathcal{C}([\![\Gamma]\!], [\![B']\!])$$

This is sufficient to give a sound interpretation of the operator theory we have described. However, when we extend this interpretation to the type theory, a complication arises. Consider the choice operator $\mathsf{cases}_{A,B,C}$ we have already introduced:

$$\Gamma \vdash \mathsf{cases}_{A,B,C}\ t\ \mathsf{of}\ x\ \mathsf{in}\ u\ \mathsf{or}\ y\ \mathsf{in}\ v{:}C$$

If Γ contains just the typing $x' : A'$, and we have a proof $\_ \vdash t' : A'$, we would expect the typing

$$\_ \vdash (\mathsf{cases}_{A,B,C}\ t\ \mathsf{of}\ x\ \mathsf{in}\ u\ \mathsf{or}\ y\ \mathsf{in}\ v)\{t'/x'\}{:}C$$

to be modelled by the composition

$$[\![\_ \vdash t'{:}A']\!]; \alpha_{x':A'}([\![x'{:}A' \vdash t{:}A \vee B]\!], [\![x'{:}A', x{:}A \vdash u{:}C]\!], [\![x'{:}A', y{:}B \vdash v{:}C]\!])$$

where $\alpha_{x':A'}$ is the function interpreting the choice construct. However, the term $(\mathsf{cases}_{A,B,C}\ t\ \mathsf{of}\ x\ \mathsf{in}\ u\ \mathsf{or}\ y\ \mathsf{in}\ v)\{t'/x'\}$ is precisely

$$\mathsf{cases}_{A,B,C}\ (t\{t'/x'\})\ \mathsf{of}\ x\ \mathsf{in}\ (u\{t'/x'\})\ \mathsf{or}\ y\ \mathsf{in}\ (v\{t'/x'\})$$

and so we would expect this term to be interpreted as

$$\alpha_\_(f; [\![x'{:}A' \vdash t{:}A \vee B]\!], f; [\![x'{:}A', x{:}A \vdash u{:}C]\!], f; [\![x'{:}A', y{:}B \vdash v{:}C]\!])$$

where $f = [\![\_ \vdash t'{:}A']\!]$. Clearly, though, we would wish these two interpretations to be equal. This is achieved by imposing the condition that $\alpha_{A'}$ be the instance at $[\![A']\!]$ of a natural transformation

$$\mathcal{C}(\_, [\![A \vee B]\!]) \times \mathcal{C}(\_ \times [\![A]\!], [\![C]\!]) \times \mathcal{C}(\_ \times [\![B]\!], [\![C]\!]) \to \mathcal{C}(\_, [\![C]\!])$$

Further, we can then model an equality judgement over general axiomatic equalities as a c.c.c. such that the interpretation of the terms is sound with respect to the axiomatic equalities.

We might summarise the situation presented here in a familiar form as follows:

$$\begin{tikzcd}
 & \text{Minimal Logic with arbitrary rules} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{Type Theory with binding operators} \arrow[rr, leftrightarrow] & & \text{Cartesian Closed Category with natural transformations}
\end{tikzcd}$$

1.2 Background

We aim to extend the account of the previous section by starting with a linear logic, λ-calculus and semantics. First, however, we need to recall some background material in a number of different areas touched on in the previous section.

Linear Logic

Linear logic was introduced by Girard in 1987 [Gir87], and is a resource logic in which contraction and weakening are allowed only on certain formulae, those labelled with the connective !.

Consider what we might call minimal intuitionistic linear logic, which is the simple logic with formulae

$$A ::= p \in \mathsf{P} \mid A \multimap A$$

and rules

$$\frac{}{A \vdash A}\ (Ax)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ (Abs) \qquad\qquad \frac{\Gamma \vdash A \quad \Delta \vdash A \multimap B}{\Gamma, \Delta \vdash B}\ (App)$$

where the turnstile $\vdash$ is now a relation holding between *multisets* of formulae and formulae. We call the axiom rule, together with the admissible cut rule

$$\frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ (Cut)$$

the *structural rules* of this logic. Note that weakening is no longer admissible.

In fact, there are two obvious differences between this minimal intuitionistic linear logic and our presentation of minimal intuitionistic logic in the previous section. The first of these occurs in the (Ax) rules; in the linear logic, we are only allowed to assume the one formula which is deduced, whereas in minimal logic we can also make any number of other assumptions. The second is more pervasive, but is seen particularly in comparing the rules for application in the two logics. In the linear logic, each proof uses a separate multiset of assumptions and the multiset union of these proves the conclusion. Hence we are able to measure the number of uses of an assumption in a derivation by the number of times it appears in the multiset of assumptions. On the other hand, in minimal logic the same assumptions are used to prove both premises and the conclusion, and therefore we have no information about how the assumptions are used, if at all. However, this new way of handling contexts also has more wide-ranging effects. One could present minimal logic equivalently with the elimination rule for the arrow given in the linear logic. Hence the differences between these two logic are all implicit in those between the structural rules. The importance of such rules in logics has been studied: for a general overview, see [Avr94], and in the particular case of those considered here, see [Lam89, Lam90a, Sza75].

We now add to this minimal linear logic rules for the exponential, as presented for example in [Abr93]. Now the formulae are given by:

$$A ::= p \in \mathsf{P} \mid A \multimap A \mid\, !A$$

where given a multiset of formulae $\Gamma = \{A_1, \ldots, A_r\}$, we write $!\Gamma$ as shorthand for the multiset of formulae $\{!A_1, \ldots, !A_r\}$, and the rules are augmented with the *promotion*, *dereliction*, *copy* and *discard* rules:

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash\, !A}\ (Prom) \qquad\qquad \frac{\Gamma \vdash\, !A}{\Gamma \vdash A}\ (Der)$$

$$\frac{\Gamma \vdash\, !A \quad \Delta, !A, !A \vdash B}{\Gamma, \Delta \vdash B}\ (Copy) \qquad\qquad \frac{\Gamma \vdash\, !A \quad \Delta \vdash B}{\Gamma, \Delta \vdash B}\ (Disc)$$

Of these, the copy and discard rules allow us to duplicate or discard assumptions of the form $!A$ for some A. Although these rules complicate the situation, the promotion and dereliction rules can be seen as an I/E-pair for the !-connective. Altogether, these connectives and rules allow us to encode minimal logic into intuitionistic linear logic using the translation given by $(A \to B) \mapsto (!A \multimap B)$.

Other connectives in the intuitionistic fragment of linear logic are the binary *tensor*, $\otimes$, the nullary *unit*, I (which acts as a unit for the tensor), the binary *product*, & (sometimes written $\times$) and the binary *coproduct*, $\oplus$. In this thesis, we will focus on the *multiplicative* fragment of intuitionistic linear logic, which has all these connectives except the product and coproduct (the *additive connectives*). Extending techniques used in the study of the multiplicative fragment to treat the additives is known to be non-trivial, and this is also the case for our work.

Although we will work exclusively with intuitionistic linear logic, we briefly present the corresponding classical linear logic, which has sequents $\Gamma \vdash \Delta$, and two important new connectives, linear negation $(\_)^\perp$ (originally presented as a defined connective except on the atomic propositions) and the 'par' connective, written $\parr$. This has the introduction (or right) rule

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}$$

It turns out that $\parr$ is the de Morgan dual of $\otimes$, and that in this logic $A \multimap B$ can be defined $A^\perp \parr B$.

This logic is of more than passing interest to us because of its position as a predecessor of the system LU [Gir93] introduced by Girard in 1993. LU has a number of interesting features, but the most relevant to our study is its use of a split context. In LU, a general sequent has the form $\Gamma; \Delta \vdash \Delta'; \Gamma'$, where the positions occupied by the Γ, Γ' are known as *intuitionistic zones* and those occupied by the Δ, Δ' are *linear zones*. Hence the above sequent should be read (roughly): "using formulae Γ intuitionistically and formulae Δ linearly, we can prove formulae Δ' linearly or formulae Γ' intuitionistically". Although LU is essentially a classical logic, it contains as subsystems minimal logic, classical logic, intuitionistic linear logic and classical linear logic, and we will take advantage in particular of the fact that it contains intuitionistic linear logic as a subsystem to present an alternative formulation of that logic. We will sketch this later in this introduction, but for now it is interesting to note that the structural rules of LU include (amongst others) admissible weakening in both intuitionistic zones, and two flavours of cut:

$$\frac{\Gamma; \_ \vdash A; \Gamma' \quad \Gamma, A; \Delta \vdash \Delta', \Gamma'}{\Gamma; \Delta \vdash \Delta'; \Gamma'}\ (\textit{I-Cut}) \qquad \frac{\Gamma; \Delta_2 \vdash \Delta_2', A; \Gamma' \quad \Gamma; \Delta_1, A \vdash \Delta_1'; \Gamma'}{\Gamma; \Delta_1, \Delta_2 \vdash \Delta_1', \Delta_2'; \Gamma'}\ (\textit{L-Cut})$$

The presence of zones adds complexity to the structural rules, and this complexity will be crucial in capturing the interaction between linear and intuitionistic types and computations.

Linear Term Calculi

The first stage in assigning terms to proofs in a natural deduction formulation of intuitionistic linear logic is to consider the minimal intuitionistic linear logic defined above. Just as the minimal intuitionistic logic can be annotated with the typed λ-calculus, the minimal intuitionistic linear logic can be annotated with a linear typed λ-calculus, and this was done by a number of people, mostly along the lines of [Abr93]. Most of these systems also went on to annotate the I/E-pair for the tensor connective in what has since become the standard way:

$$\frac{\Delta_1 \vdash t{:}A \quad \Delta_2 \vdash u{:}B}{\Delta_1, \Delta_2 \vdash t \otimes u{:}A \otimes B}\ (\otimes I) \qquad\qquad \frac{\Delta_1 \vdash t{:}A \otimes B \quad \Delta_2, x{:}A, y{:}B \vdash u{:}C}{\Delta_1, \Delta_2 \vdash \mathsf{let}\ x \otimes y{:}A \otimes B\ \mathsf{be}\ t\ \mathsf{in}\ u{:}C}\ (\otimes E)$$

From now on we may omit type annotations in terms, where they can be reconstructed from derivations. It should be noted that in some accounts, which include those of Abramsky [Abr93] and Wadler [Wad93] the syntax $\mathsf{let}\ t\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ v$ and slight variants of it were used instead of that given here. The term construct $t \otimes v$ is self-explanatory, and the term construct $\mathsf{let}\ x \otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u$ is a pattern-matching constructor which can be thought of as decoding its argument t into two parts which are then substituted for the bound variables x and y in its argument u.

As in minimal logic, we can give an equality judgement which is a congruence over axiomatic equalities corresponding to proof-equivalences. The β- and η-equivalences for the $\otimes$ connective are easily given, and they generate β- and η-axiomatic equality judgements as follows:

$$(\beta)\ \ \Gamma \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ t \otimes u\ \mathsf{in}\ v = v\{t, u/x, y\}{:}C$$
$$(\eta)\ \ \Gamma \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ t\ \mathsf{in}\ x \otimes y = t{:}A \otimes B$$

Unlike minimal logic, however, when we consider the whole equality judgement on terms corresponding to proof equivalence and normalisation, the term calculus as presented so far already has one particular drawback which is inherent in much work on linear logic. Whereas in the type-theory arising from minimal logic with no additional rules, the equality judgement is simply based on the familiar $\beta\eta$-equality of the typed λ-calculus, in the case of linear logic equalities arise which are not β- or η-equalities, the so-called "commuting conversions". For example, consider the following proofs:

$$\dfrac{\dfrac{\Gamma_1, A \vdash B \otimes B' \quad \Gamma_2, B, B', A \vdash C}{\Gamma_1, \Gamma_2, A \vdash C}\ (\otimes E)}{\Gamma_1, \Gamma_2 \vdash A \multimap C}\ (\multimap I) \qquad \dfrac{\Gamma_1 \vdash B \otimes B' \quad \dfrac{\Gamma_2, B, B', A \vdash C}{\Gamma_2, B, B' \vdash A \multimap C}\ (\multimap I)}{\Gamma_1, \Gamma_2 \vdash A \multimap C}\ (\otimes E)$$

where the proof $\Gamma_1, A \vdash B \otimes B'$ is obtained from the proof $\Gamma_1 \vdash B \otimes B'$ by weakening. The difference between these two proofs arises because we have used the two essentially independent rules in two different orders. However, semantically and intuitively, it is sensible to ask that they be equivalent. This gives us the axiomatic equality:

$$\Gamma \vdash \lambda x.\mathsf{let}\ y \otimes y'\ \mathsf{be}\ t\ \mathsf{in}\ u = \mathsf{let}\ y \otimes y'\ \mathsf{be}\ t\ \mathsf{in}\ \lambda x.u{:}A \multimap C$$

which holds under appropriate free and bound variable conditions. There are a variety of such proof equivalences and corresponding axiomatic equalities. Categorically, these equalities are soundly modelled by virtue of the interaction of the $\otimes$-functor with composition.

Another way of thinking of them is to recall the intuitive reading given above of the let -$\otimes$ constructor as a pattern matching substitution. Then it should have the usual properties of substitution, for example commuting with other substitutions and term structure, under suitable free variable conditions. A similar situation arises in a more familiar setting; if we consider the terms:

$$\Gamma \vdash \mathsf{cases}_{A,A',B\to C}\ t\ \mathsf{of}\ x\ \mathsf{in}\ (\lambda y.u)\ \mathsf{or}\ x'\ \mathsf{in}\ (\lambda y.v){:}B \to C$$

and

$$\Gamma \vdash \lambda y.(\mathsf{cases}_{A,A',C}\ t\ \mathsf{of}\ x\ \mathsf{in}\ u\ \mathsf{or}\ x'\ \mathsf{in}\ v){:}B \to C$$

it seems very plausible that they be equal, since the functions they describe are intuitively the same. However, such an equality is not derivable from the $\beta\eta$-equalities of the construct, and hence must be added as an axiomatic equality.

Given term annotations for this system, the most difficult problem remaining is the annotation of the intuitionistic rules for the connective !. A first attempt, by Abramsky [Abr93], simply annotated the key promotion rule:

$$\frac{\Gamma \vdash t{:}A}{\Gamma \vdash !t :!A}\ (Prom)$$

where Γ is a sequence of typings $x_1 :!A_1 \ldots x_r :!A_r$. However, Wadler [Wad92] and Mitchell and Lincoln [LM92] noticed that there was a technical problem with this presentation, because of the failure of a linear substitution lemma (given a typing judgement $\Delta \vdash t{:}A$ and one $\Delta', x{:}A \vdash u{:}B$, there should be a typing judgement $\Delta, \Delta' \vdash u\{t/x\}{:}B$). This is connected to the fact that the !-connective does not have a "real" natural deduction presentation, since it has four rules rather than an I/E-pair.

In order to avoid the problem, Benton, Bierman, dePaiva and Hyland gave a syntax [BBdPH93b, BBdPH93a] which changed the form of each rule except (Der) by adding an element of substitution for each of the fresh assumptions (each of $!\Delta$ in the case of $Prom$). This gave the rule

$$\text{Promotion}\ \frac{\Delta_i \vdash M_i :!A_i \quad (i = 1..r) \quad x_1 :!A_1, \ldots, x_r :!A_r \vdash N{:}B}{\Delta'' \vdash \mathsf{promote}\ \vec{M}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ N :!B}$$

Benton proved strong normalisation for the $\beta - cc$ fragment of the equality of this system [Ben95b]. We present this system in full in chapter 2. However, this system is complex, notably in its equality judgement. Since the !-connective does not have an I/E-pair in the logic, it is not immediately clear what its axiomatic equalities should be, and in fact there are a large number of axiomatic equalities involving the term constructs associated with it.

Considering classical logics, although Abramsky [Abr93] gave an annotation for classical linear logic, it had been thought that term calculi were not the most appropriate presentation of the proofs of classical linear logic, partly because of the huge number of proof equivalences induced by the possibility of selecting more than one formula on the right of the turnstile as the primary formula of a rule. Hyland and dePaiva [HdP93] gave a system of term assignment for a linear logic with multiple conclusions, having the par constructor, which also exhibited these characteristic equivalences.

A more significant development occurred when Parigot presented a system of term assignment for classical logic [Par92, Par93] which has since been adapted by Bierman [Bie96b, Bie96a] to give a linear form. The main feature of Parigot's system and its linear derivatives is that there is an identified formula on the right which is always the active formula in introduction rules, and there are explicit structural rules allowing the exchange of this identified formula with another formula on the right, thereby recapturing the power of classical logic.

As yet there have been no attempts to give a syntax for the proofs of the unified logic LU. However, a preliminary linear version of the logical framework [HHP93] based on a natural deduction presentation of its intuitionistic fragment was given in [MPP92], based on a logic having sequents of the form $\Gamma; \Delta \vdash A$. This presentation independently inspired the annotations of [Plo93b] and [Wad93], and the linear logical frameworks of Pfenning and Cervesato [CP96, Pfe94] and Ishtiaq and Pym [IP]. We will sketch the annotation of [Plo93b] in the next section, and present a full development in chapters 2 and 3 of this thesis. This work was done in the academic year 1994-1995, published initially in [Bar96].

Meanwhile, Benton has given [Ben95a, Ben94] a type-theory in this style for a general categorical model which encompasses the monoidal adjunction models of intuitionistic linear logic, and Benton and Wadler [BW96] have used this type-theory to relate the computational λ-calculus, due to Moggi [Mog91], to linear logic.

Systems with split contexts have also proved popular in the logic programming community. A system similar to those mentioned above was presented by Miller [Mil94a, HM94] again building on the work of Miller, Plotkin and Pym [MPP92], and another by Harland and Pym [PH94, HPW96].

Proof nets

Although term calculi are a popular syntax for the proofs of many kinds of logics, mostly intuitionistic, Girard's original presentation of linear logic [Gir87] employed an alternative graphical syntax, that of *proof nets*. These are essentially graphs in which edges, labelled with formulae, represent assumptions and conclusions occuring in a derivation, and nodes, labelled with rule names, represent the application of the various rules. The main advantage of this approach is that rules are not applied globally, as a term construct is applied to a whole term, but only apply to the edges labelled with the formulae involved in the rule application.

In the intuitionistic case, therefore, proof nets have the distinct advantage over term-calculi that two proofs which differ by a proof equivalence which would induce a commuting conversion on terms are assigned the same proof net. Further, since the $\beta\eta$-equalities which hold between proofs are essentially local in the proof-net representation, we can decide them using a local rewrite system.

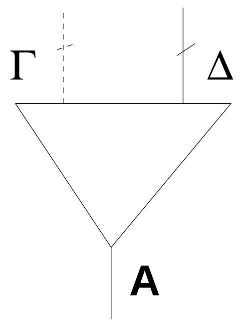


Figure 1.1: A Proof Net

To see how proof nets work, let us consider the translation from a term calculus for a split-context linear logic to proof nets for that logic, due to the author. We represent the proof net corresponding to a deduction $\Gamma; \Delta \vdash A$ as in figure 1.1, where a wire with a stroke through it represents a number of wires, the dashed wires labelled with Γ represent the intuitionistic assumptions and the plain wires labelled with Δ represent the linear assumptions.

We can see how proof nets avoid commuting conversions by considering the translation of terms into proof nets via a translation Φ. Consider the action of Φ on the tensor introduction term $\Gamma; \Delta \vdash t \otimes u : A \otimes B$, which can be seen in figure 1.2. In this proof net, we can see that the edges representing the conclusions of the two subterms t and u are connected to a node for the rule instance $\otimes I$, and a new trailing edge representing the conclusion of the whole term is added.

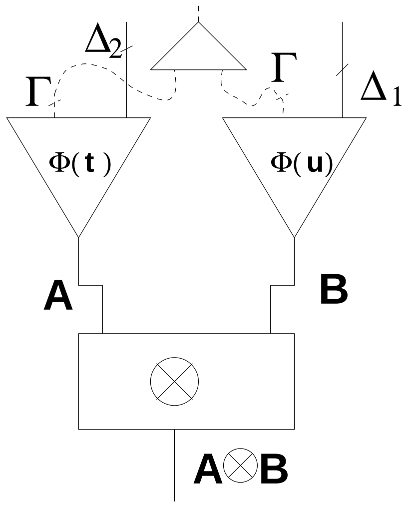


Figure 1.2: Tensor Introduction

Also, notice the treatment of the intuitionistic assumptions. Since both subterms are typed using the same set of intuitionistic assumptions Γ, we need to duplicate each assumption in Γ to provide one copy for each sub-proof. This behaviour will be repeated for the translation of any rule with two premises.

Now consider the tensor elimination rule. The derivation

$$\Gamma; \Delta \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u{:}C$$

is mapped by Φ to the proof net in figure 1.3. Again, we copy the intuitionistic

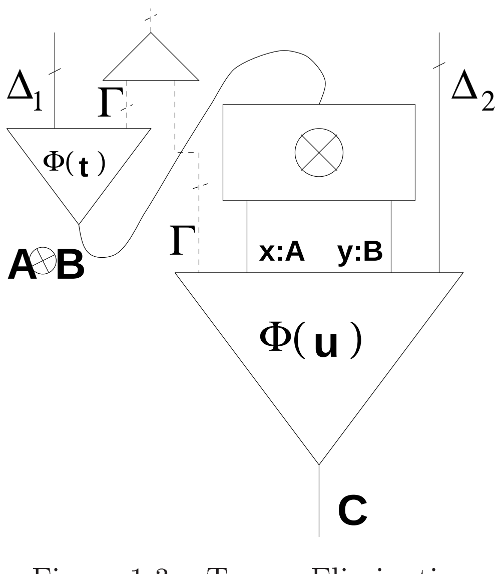


Figure 1.3: Tensor Elimination

assumptions, once for each sub-net, but in this case, the new node which is added to represent the $\otimes E$ rule is only connected to the edges labelled by $x : A$, $y : B$ and $A \otimes B$. Hence, the trailing edge representing the conclusion of the term is the same trailing edge representing the conclusion of the sub-term u. This property of the proof-net representation of the $\otimes E$ rule accounts for the fact that terms equal via commuting conversions due to $\otimes E$ are mapped to equal proof nets, as the reader may verify by drawing some sample proof nets. At this point it is worth noting that although proof nets do remove the proof-equivalences due to commuting conversions, those which arise in connection with the linear analogue of $\vee$ and its corresponding cases constructor do not map to equalities in the same way. The proper treatment of the additives is an ongoing research area.

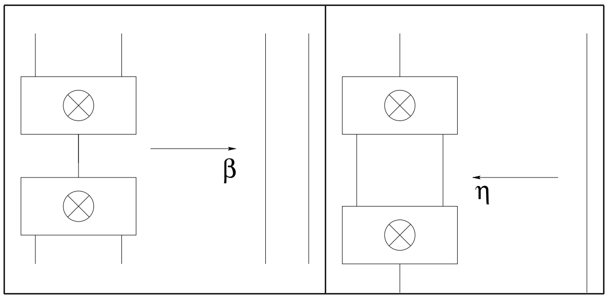


Figure 1.4: Tensor β-η-rewrites

Now, we can give rewrites capturing the $\beta\eta$-equality for the $\otimes$-construct as seen in figure 1.4. These should be read as saying that any proof net containing the redex as a subnet rewrites to the proof net having that redex replaced by the reduct. It is obvious just from the form of this presentation that these are local. It is worth noting that the η-rewrite is given the form of an expansion, rather than the more familiar contraction. This is technically in order to make the rewrite confluent, but is also motivated by work of Ghani [Gha95].

Since their introduction, proof-net systems have been given for most, if not all, linear logics, for example [GAL92b, Gir96]. Beyond these, graphical forms based on proof nets have been used in a number of areas. For example, the nets of [CS97, BCS95], used in categorical coherence reasoning, are very close neighbours of proof nets for intuitionistic linear logic, for example as given in [DR89]. More generally, Lafont has defined *interaction nets* [Laf90, Laf95], which are a general graphical syntax based on proof nets extended with arbitrary node types. This development is similar to our addition of general rules to minimal logic, and allows interaction nets to apply to a wide range of computational situations.

Possibly the most far-reaching impact of proof nets has been on the optimal λ-reduction problem [Lév80, Wad71] in the untyped λ-calculus. In defining an efficient evaluation strategy for λ-terms, it quickly becomes obvious that, for example, in a term such as $(\lambda x.xx)((\lambda x.y)z)$ we should first reduce the application $((\lambda x.y)z)$, since this will otherwise be reduced twice, one in each copy replacing an x in xx. The optimal λ-reduction problem is to efficiently mechanise the strategy for deciding which rewrite is appropriate at each stage. There is no simple solution, but Lamping [Lam90b] gave a graphical method based upon a system of λ-graphs with added connectives. Gonthier, Abadi and Lévy [GAL92a] then showed that this extra structure corresponded in the typed case to that of proof nets for intuitionistic linear logic, and that the required reduction strategy could be explained systematically under this interpretation. Mackie [Mac94] also investigated implementation issues for λ-calculus using graphical frameworks.

Action Calculi

Action calculi were first presented by Milner in [Mil93c]; the definitive reference is [Mil96]. They can be understood as a structural framework equipped with general operator rules determined by arities similar to those of binding operators. Originally designed to allow a uniform presentation of various process calculi in order to compare them, action calculi are powerful enough to represent a wide range of interesting systems. A particular action calculus is determined by a signature, which in Milner's presentation [Mil96] consists of a set of *prime arities*, similar to basic types in a term calculus, and a set of operators having arities built over those prime arities, called *controls*. Given a signature, *actions*, similar to terms, are constructed, themselves having arities which are analogous to the types assigned to terms in a type-theory. However, in contrast to the situation in a type theory, actions are constructed combinatorially from the internal language of a symmetric monoidal category with certain other constructors (in something of the sense of Curien's categorical combinators [Cur86, Cur85]). In this framework, variables, which are called *names*, are reserved for a purpose analogous to that of variables in the intuitionistic term calculus, with the linearity springing directly from the categorical language. In fact, Gardner [Gar95] has shown that each action calculus is equivalent to a *closed action calculus* which does not have free names.

A significant amount of theory supports action calculi. They are equipped with normal forms, called *molecular forms*. These, interestingly, can be given in a graphical form, as *action graphs*, which are related to proof nets. They have a semantics, which has been steadily refined since Milner's original paper introduced action structures [Mil93c]. Control structures [MMP95], which are categories with naming structure, were followed by a fibrational equivalent not requiring naming structure, due to Hermida and Power [HP95]. This was proved equivalent by Power [Pow96] to an alternative formulation, which is closely related to the structural fragment of models of linear logic. Minor variants of these models were used in the work of Gardner *et al.* [HG, BGHP97].

A higher-order extension of the action calculus was given by Milner in [Mil94b, Mil93a], and a reflexive extension in [Mil94c], providing a facility for recursion. Normal (molecular) forms [Mil93b] exist for these extensions, although not for their combination. Hasegawa and Gardner have related the higher-order extension of action calculi to Moggi's computational λ-calculus [HG].

Categorical Semantics

We now introduce and motivate the semantics we will use to model linear logic. The models we will consider are based on linear-non-linear models. These are pairs of categories $(\mathcal{C}, \mathcal{S})$ such that $\mathcal{C}$ is a cartesian closed category (CCC, or CC category) and $\mathcal{S}$ is a symmetric monoidal closed category (SMCC, or SMC category) and there is a monoidal adjunction between them:

$$\begin{tikzcd} \mathcal{C} \arrow[r, shift left=1ex, "F"] \arrow[r, phantom, "\perp"] & \mathcal{S} \arrow[l, shift left=1ex, "G"] \end{tikzcd}$$

The intention behind the construction is that the normal power of intuitionistic logic, arising from the exponential, should be modelled in the CCC, with the intuitionistic linear logic being as usual modelled in the SMCC. This idea emerged in 1993 from discussions between a number of people, including Plotkin, Benton and Hyland. However, it was only during further work by Benton [Ben95a] and Bierman [Bie95] that the details became clear, and in particular that it was necessary to impose the requirement that the adjunction be monoidal. In [Ben94] there is an extensive comparison between these models and the previously proposed models [BBdPH93b]. In fact, although Benton required that the cartesian category be closed, this is not essential for our purposes, and we take the more general definition.

Let us consider the intuition behind this in a little more detail. Imagine that the category $\mathcal{C}$ is a category of total functions, and that the category $\mathcal{S}$ is a category of 'computations', or processes—we are deliberately being vague about the precise nature of this, but we certainly mean to allow the possibility of non-termination or non-functional behaviour of other kinds. Note that the category of total functions must be cartesian: there can only be one function from any type to the unit type, that which returns the unique element of the unit type; and, further, the function $f;\langle g,h\rangle$, which evaluates f and uses the result in g and h respectively, must be the same as the function $\langle f;g,f;h\rangle$ which evaluates f twice in evaluating the functions $f;g$ and $f;h$.

By contrast, the category of computations cannot be cartesian, at least because there are potentially many computations returning no result. These might include nonterminating computations, processes which interact with the user, or ones which have other side-effects. Also, it is not the case that doing a computation once and using the result as the input to two other computations is the same as doing the first computation twice, once for each computation. This equality will fail, for example, if the first computation has any side effects.

Now consider the action of the functors in our intuitive model. The functor F takes functions to computations, intuitively saying that all functions can be computed, or that functions are a subset of processes. This is a natural requirement on any notion of computation. The existence of the functor G then corresponds to a kind of completeness; it says that for each computation or process, there must be a function simulating it on a suitable representation of its input. The adjunction $F \dashv G$ specifies that there must exist natural transformations $FG \Rightarrow Id$ and $Id \Rightarrow GF$. The first of these then implies that we can reconstruct a computation from the function representing it, and the second that from a function we can construct a function representing its computation.

Not all candidate notions of total function and computation satisfy the above intuitive representability and completeness conditions. Partly for this reason, we will later be considering an important fragment of this model, which we might call the *structural fragment*. This consists of a cartesian category $\mathcal{C}$, a (not necessarily closed) symmetric monoidal category $\mathcal{S}$ and a strong monoidal functor $F : \mathcal{C} \to \mathcal{S}$ (which may not have an adjoint):

$$\mathcal{C} \xrightarrow{\quad F \quad} \mathcal{S}$$

Now in order for notions of total function and computation intuitively to provide a model of this fragment, we require only that functions yield computations, or equivalently can be computed by processes.

The basic components of this model are all very familiar to category theorists, and substantially predated the introduction of linear logic; for example, the definitions of symmetric monoidal category (also *tensor* category), cartesian closed category and monoidal adjunction can all be found in [Mac71]. One major area of research in category theory is the study of *coherence problems* for various types of categories. In particular, coherence in monoidal categories has been the subject of much study, for example in [KM72, KL80, Mac71, Sol90].

The coherence problem for a class of categories is the problem of deciding which diagrams (made up from elementary natural transformations present in each instance of the class) commute in every category in the class. One classical result due to MacLane [Mac71, Mac63] is that in (symmetric) monoidal categories, every such diagram commutes. However, the situation is not uniformly this simple; in the case of symmetric monoidal *closed* categories, we do not have that every diagram made out of the elementary natural transformations commutes [KM72]. For these, there is a famous counter-example which we will present in chapter 8. The theories of monoidal categories and cartesian categories have further been clarified by work on general classes of categorical structure by Kelly, Power and Robinson [BKP89, PR94].

One interesting approach to such problems uses syntactic methods. Considering the case of cartesian closed categories, we know that a category built from the terms of typed λ-calculus (the *term category*) is *complete* for the class of cartesian closed categories, which is to say that any equality between elementary natural transformations holds in every cartesian closed category if and only if it holds in the term category. Since the equality of arrows in the term category is based on the equality judgement of the typed λ-calculus, to decide this is to solve the coherence problem for cartesian closed categories. More generally, to decide the equality on any accurate representation of the proofs of minimal logic is to solve this coherence problem. Along these lines, in [MRA93], Mackie, Román and Abramsky used a type theory for multiplicative linear logic to approach the problem of coherence for symmetric monoidal closed categories. Another natural question is whether proof nets help in the solution of coherence problems, as they are an efficient representation of proofs, and indeed, Seely *et al.* [CS97, BCST96] have adapted proof-net technology for the $\otimes - \parr$-fragment of classical linear logic to this end.

Since the advent of linear logic, its models have been of major interest. Barr [Bar91] initially gave a semantics for the $\otimes - \perp$ fragment of classical linear logic in his $*$-autonomous categories, which have a symmetric monoidal structure with a dualising object [Bar79] (again substantially predating linear logic itself). Then Seely [See89] and the group at Cambridge [Bie95, Ben95a, BBdPH93a] developed the underlying ideas for different fragments of the full logic.

For the purposes of this thesis, we are particularly interested in the work of Day [Day70b, Day73, Day70a] on the Yoneda embedding of a category $\mathcal{C}$ into its presheaf category $Set^{\mathcal{C}^{op}}$. He showed that given a symmetric monoidal category, the presheaf category is symmetric monoidal closed, and the Yoneda embedding functor is symmetric monoidal. The analogous result holds for cartesian categories. We will use these results to generate a model of multiplicative intuitionistic linear logic from one of the structural fragment, so obtaining conservative extension results.

1.3 Development

Having motivated our approach, and surveyed the relevant background material, we now give an overview of the original work in this thesis.

We start by presenting, in chapter 2, the reformulation of intuitionistic linear logic based on LU which was mentioned in the previous section. Known consequently as DILL, *dual intuitionistic linear logic* has sequents of the general form $\Gamma; \Delta \vdash A$, where Γ is a *set* of intuitionistic assumptions, and Δ is a *multiset* of linear assumptions.

The basic axioms are:

$$\frac{}{\Gamma, A; \_ \vdash A}\ (I\text{-}Ax) \qquad \frac{}{\Gamma; A \vdash A}\ (L\text{-}Ax)$$

which, together with the admissible rules:

$$\frac{\Gamma; \Delta \vdash B}{\Gamma, A; \Delta \vdash B}\ (Weak)$$

$$\frac{\Gamma; \_ \vdash A \quad \Gamma, A; \Delta \vdash B}{\Gamma; \Delta \vdash B}\ (I\text{-}Cut) \qquad \frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, A \vdash B}{\Gamma; \Delta_1, \Delta_2 \vdash B}\ (L\text{-}Cut)$$

make up the *structural rules* of the logic. As might be expected from our previous discussion of the relationship between the structural rules and those for the connectives, in rules having two premises the intuitionistic contexts are shared and the linear contexts are kept separate; for example, consider the tensor I/E-pair:

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2 \vdash B}{\Gamma; \Delta_1, \Delta_2 \vdash A \otimes B}\ (\otimes I) \qquad \frac{\Gamma; \Delta \vdash A \otimes B \quad \Gamma; \Delta_2, A, B \vdash C}{\Gamma; \Delta_1, \Delta_2 \vdash C}\ (\otimes E)$$

Now, the rules for the !-connective are presented as an I/E-pair, in conventional natural deduction fashion:

$$\frac{\Gamma; \_ \vdash A}{\Gamma; \_ \vdash !A}\ (!I) \qquad \frac{\Gamma; \Delta_1 \vdash !A \quad \Gamma, A; \Delta_2 \vdash B}{\Gamma; \Delta_1, \Delta_2 \vdash B}\ (!E)$$

The rules of the logic, and particularly this I/E-pair for the !-connective, make the proof structure much simpler than that of intuitionistic linear logic in its conventional presentation. One characterising proof equivalence of the logic is as follows:

$$\frac{\Gamma; !A, \Delta \vdash B}{\overline{\Gamma, A; \Delta \vdash B}}$$

In the same way as we did above for minimal logic, we proceed in chapter 2 to give a type-theory in which the rules of the logic become typing rules, using the Curry-Howard correspondence. We build this type theory over a signature of primitive types and constants, and given a typing judgement $\Gamma; \Delta \vdash t\colon A$ we have that variables typed in Γ may occur many times (or not at all) free in t, whereas variables typed in Δ must occur exactly once free in t. The variables typed in Γ must be disjoint from those typed in Δ. The most significant difference between this type theory and that due to [BBdPH93b], presented above, occurs in the treatment of the !-connective and its rules. The straightforward term assignment

$$\frac{\Gamma; \_ \vdash t\colon A}{\Gamma; \_ \vdash !t :!A}\ (!I) \qquad \frac{\Gamma; \Delta_1 \vdash t :!A \quad \Gamma, x\colon A; \Delta_2 \vdash u\colon B}{\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ u\colon B}\ (!E)$$

allows us to prove the two substitution lemmas corresponding to the cut rules. Further, the β- and η-equivalences for the !-connective arising from its presentation with an I/E-pair generate β- and η-axiomatic equalities for it:

$$(\beta)\ \ \Gamma; \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ !t\ \mathsf{in}\ u = u\{t/x\}\colon A \qquad (\eta)\ \ \Gamma; \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ !x = t :!A$$

This is in contrast to the complex treatment of the connective forced in the case of the single-context linear logic presented above.

We then complete the linear version of the familiar triangle by giving, in chapter 3, a categorical semantics for the type-theory of DILL, which is based on the adjunction model of linear logic described earlier. We show that the type theory is sound and complete with respect to these models in the standard way, using a term model construction.

Now, just as we sketched the construction of a general theory of operators over the structural fragment of minimal logic, it is profitable to consider a general theory of rules and hence operators built on the structural fragment of DILL, but with one slight alteration. Instead of assuming that there is just one set of primitive propositions, as in DILL, we assume two sets; a set of primitive *linear* propositions (or types) and a set of primitive *intuitionistic* propositions (or types). We further assume that any primitive intuitionistic type is also a primitive linear type. In this definition, we are intuitively saying that some linear types are in fact value types, for example natural numbers or booleans, elements of which can be copied.

We construct this logic in chapter 4; a sequent has the general form $\Gamma; \Delta \vdash A$ where Γ is a set of primitive intuitionistic propositions, Δ is a multiset of primitive linear propositions, and A is a primitive linear proposition. Because we have replaced the set of primitive propositions of DILL with two sets, the new logic has another structural rule, the I-L-rule, which holds for any *intuitionistic proposition* Q:

$$\frac{\Gamma; \Delta_1 \vdash Q \quad \Gamma, Q; \Delta_2 \vdash A}{\Gamma; \Delta_1, \Delta_2 \vdash A}\ (I\text{-}L)$$

We build a theory of general operators on this structural foundation, and proceed to annotate it in chapter 4, as we did for minimal logic. Considering equivalence between proofs in this two-zoned setting raises substantial naturality issues, which we discuss at length before presenting the equality judgement of the type-theory. We parameterise this equality judgement over a set of axiomatic equalities, giving us an expressive theory of binding operators in the setting of a linear type-theory, which we call *general linear type-theory*.

We then consider the categorical semantics of this general framework, using natural transformations, in chapter 5. The basis of the model is the structural fragment of models of linear logic mentioned above.

Having developed the theory of this general linear type theory, it is helpful to consider an example, and in fact we provide a set of examples by showing that each instance of Milner's (static) action calculi, presented earlier, is equivalent to an instance of our general type theory in which the controls correspond to binding operators of a certain restricted form. In chapter 6, we first define action calculi and show the equivalence to instances of our theory, and then go on to consider three higher-order extensions of action calculi, one of which is due to Milner [Mil94b], one of which is largely folklore, and one of which is due to the author. We show that these extensions, and various structure-preserving translations which exist between them, correspond to extensions of the type-theories corresponding to action calculi and translations between them in our theory. Finally in chapter 6, we consider the semantics for action calculi inherited from that of the instances of our general theory, and compare it to the semantics of action calculi given by Power [Pow96].

Inspired by the generality of the higher-order extensions of action calculi, in chapter 7 we define *higher-order* general linear type theories, which are a subset of general linear type-theories having operators and equalities based on those of linear logic for the $\multimap$, $\otimes$ and $!$ connectives. We can then define the *higher-order extension* of a given instance of our theory as that theory with the connectives, operators and the corresponding equalities 'freely' added. It is not immediately clear that this addition does not yield a trivial structure (as is the case with some extensions of action calculi, see [Mil94b]), but we are able via the semantics to use a presheaf construction to show that the obvious embedding of a theory into its higher-order extension is conservative. As a corollary, this shows the same conservativity property for the morphisms of action calculi described earlier.

We complete our development in chapters 8 and 9 by considering decidability issues for the type-theories and equalities we have introduced. Historically, deciding equality judgements in linear type-theories has been complex, because of the commuting conversions mentioned earlier, and because of confluence problems with the reductive orientation of the axiomatic equalities. The situation is much clearer in the case of proof nets, since it is not complicated by the commuting conversions.

We might, therefore, consider a system of proof nets for our general linear type-theory. However, to define such a graphical system and prove the delicate results on its rewrites would be cumbersome. Hence we present a system of *relations*, which are non-graphical analogues of proof nets for general linear type-theories. It is worth noting that the proof nets which we use for intuition can be formally presented, and in fact that this has been done by the author [Bar] for a closely-related system of proof nets for the logic DILL.

A proof net labelling a derivation of $\Gamma; \Delta \vdash A$ corresponds to a relation D relating variables $\vec{x}$ of type Γ, $\vec{y}$ of type Δ to a variable z of type A. Each proof-rule corresponds to a primitive relation, and cut is modelled by a multiary relational composition. Let us take, for example, the proof net (for DILL) in figure 1.3.

This corresponds to the term

$$\_; x'{:}A \otimes C, y{:}B \vdash \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ x_1 \otimes (y \otimes x_2){:}A \otimes (B \otimes C)$$

and the relation

$$R(x', y, z) = \exists x_1{:}A, x_2{:}C, y'{:}B \otimes C.(z = x_1 \otimes y') \wedge (y' = y \otimes x_2) \wedge (x_1 \otimes x_2 = x')$$

If we were then to abstract y from this, to give the term

$$\_; x'{:}A \otimes C \vdash \lambda y{:}B.\mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ x_1 \otimes (y \otimes x_2){:}B \multimap (A \otimes (B \otimes C))$$

this would have the relation

$$S(z', x') = \forall y{:}B.\exists z{:}(A \otimes (B \otimes C)).(z'y = z) \wedge R(x', y, z)$$

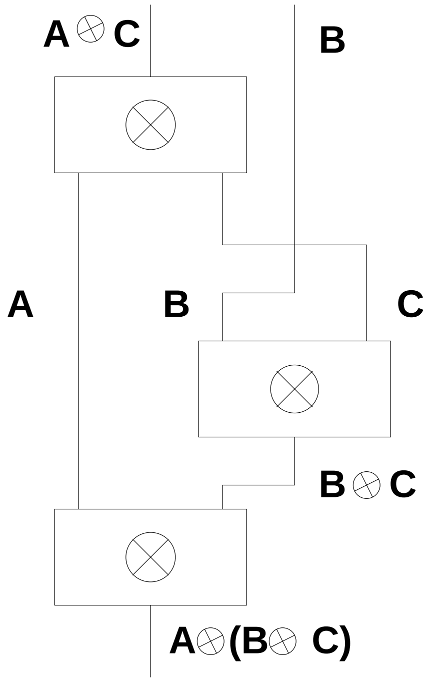


Figure 1.5:

Just as we have that commuting conversion proof equivalences correspond to equalities on the proof nets, we can see that the commuting-conversion equivalent form of this proof (constructed as on page 14), which has the term

$$\_; x'{:}A \otimes C \vdash \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ (\lambda y{:}B.x_1 \otimes (y \otimes x_2)){:}B \multimap (A \otimes (B \otimes C))$$

has a relation

$$\begin{aligned} S'(z', x') = {} & \exists x_1{:}A, x_2{:}C.(x_1 \otimes x_2 = x') \wedge \\ & \quad \forall y{:}B.\exists z{:}(A \otimes (B \otimes C)), y'{:}B \otimes C.(z'y = z) \wedge (y' = y \otimes x_2) \wedge (z = x_1 \otimes y') \end{aligned}$$

which is the same as $S(z', x')$ up to certain obvious syntactic manipulations of the quantifiers.

Using these relations, we can give a syntactic representation of the proofs of our general linear type-theory. We can then define rewrites on them which are confluent and strongly normalising, such that two relations have the same normal form under rewriting if and only if they correspond to terms which are judged equal in an instance of the type-theory over an empty axiom set. We call such an instance *pure*. This strongly-normalising rewrite then provides a decision procedure for any pure instance of our general linear type-theory.

Furthermore, this result immediately provides a complete solution to the coherence problem for the structural fragment of linear logic. Via our completeness result, a diagram commutes in all models of any suitable pure instance if and only if it does in the term-model, and checking this amounts to checking a term equality in the pure instance, which is therefore decidable.

However, it must be said that there are few useful systems which arise as pure instances of our general linear type theory; in order to present systems for action calculi and linear logic we have had to incorporate extra axiomatic equalities, which mean that our decidability result no longer applies. We proceed, therefore, to extend the system of relations and rewrites to the higher-order case. We define *pure higher-order instances* to be instances of our higher-order general linear type theory over an empty axiom set. We then give relations and rewrites on them which again are confluent and normalising, and such that two relations have the same normal form if and only if they correspond to terms which are judged equal in the pure higher-order instance of the type-theory. As before, this provides a decision procedure for such pure higher-order instances, and this time provides a complete solution to the coherence problem for models of our higher-order general linear type-theory, in the same way as before.

This result is more substantial. It has as corollaries the decidability of the equality of all the action calculi fragments studied (by virtue of our conservativity results), the decidability of the equality judgement in the type theory of DILL, and hence the decidability of the equality judgement in the 'Cambridge' type theory [BBdPH93b]. We should also note that Ghani [Gha96] independently decided the $\beta\eta - cc$ equality of DILL($\mathbb{C}$), working within the term syntax and using expansionary η-rewrites.

Finally, we note that the system of relations and rewrites for a pure instance of our general linear type-theory embeds soundly and fully into the system of relations and rewrites for the higher-order extension of the pure instance. In this way we obtain a syntactic conservativity proof for conservativity on pure instances, a subcase of the problem solved semantically in chapter 7.

1.4 Chapter Summary

Having introduced our approach, and the main concepts involved, we summarise the contents of the rest of this thesis.

2) An Alternative Formulation of ILL This chapter introduces our alternative formulation of intuitionistic linear logic built over sequents $\Gamma; \Delta \vdash A$. Having given the alternative logic, we give the derived type theory and some useful results, and then proceed to show that the type theory is equivalent to the familiar Cambridge type theory [BBdPH93b] over ILL, up to provable equality.

3) **The Semantics of DILL** In this chapter, we give a semantics for the type theory based on DILL, and show it to be sound and complete by the construction of a term model. We also consider the relation between the models we have chosen and another influential notion of model, first stated for the Cambridge type theory.

4) **Linear Type Theories** In this chapter, we move away from linear logic. We present a generalisation of Aczel's binding operators, and then introduce a general linear logic over the structural rules of DILL, which incorporates rules derived from these binding operators. We then give the type-theory $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ based on this logic.

5) **The Semantics of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$** In this chapter we develop a semantics of the generalised type-theory presented in the previous chapter. This is based around the structural fragment of the models of ILL.

6) **Action Calculi and Extensions** In this chapter we introduce Milner's action calculi and show how they can also be viewed as instances of our general linear type theory. We also show the same for various higher-order extensions of action calculi, including that given by Milner himself. Further, we consider the semantics induced by our semantics for the general linear type theory.

7) **Higher-Order Extensions** In this chapter we introduce higher-order type theories in general, give the canonical higher-order extension of a general linear type theory, inspired by linear logic, and use a categorical argument to show that the embedding of any general linear type-theory into its higher-order extension is conservative. We then consider various implications of this result for the theory of action calculi. We also introduce an instance of the higher-order type theory which is equivalent to the system $\mathsf{DILL}(\mathbb{C})$.

8) **Normal Forms for $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$** In this chapter, we introduce relations, our syntax for proofs based on proof nets. We show that using a rewrite and an equality on relations, we can give a system which is equivalent to the familiar general linear type-theory under a restricted derivable equality.

9) **Normal Forms and Decidability** This chapter extends the results of the previous chapter by showing that the full provable equality of any pure in-

stance of our general linear type theory is equivalent to a rewrite over and equality on relations. We further show that this rewrite is strongly normalising, and the equality is decidable, thereby showing that the provable equality judgement of the general linear type-theory with empty axiom set is decidable. This result is then extended by giving rewrites for the higher-order type-theories which decide the axiomatic equality of the higher-order theory without arbitrary axioms.

10) **Conclusion** We discuss the implications of this work, future directions and questions raised, in particular discussing classical linear logics, enriching our general linear type-theory with a rewrite, and applications to process calculi.

1.5 Historical Notes

The work in chapters 2 and 3 is based on a talk [Plo93b] and unpublished notes [Plo93a] by Gordon Plotkin, whilst the work on action calculi, higher-order action calculi and conservativity in chapters 6 and 7 is a reformulation of work done in collaboration with Philippa Gardner, Gordon Plotkin and Masahito Hasegawa in [BGHP97].

Chapter 2

An Alternative Formulation of ILL

We introduce an alternative natural deduction formulation of ILL, *dual intuitionistic linear logic* (henceforth DILL), in which we use two kinds of assumptions, linear and intuitionistic. The exponential is introduced with rules allowing a deduction of a formula $!A$ from a formula A when no linear assumptions have been used, and eliminated by allowing a conclusion $!A$ to substitute for an intuitionistic assumption A.

When we express this natural deduction formulation in a sequent style, the general form of a sequent is $\Gamma; \Delta \vdash A$, which is interpreted as meaning that from intuitionistic assumptions in Γ and linear assumptions in Δ we can deduce A. This splitting of the context leads to an extra intuitionistic axiom form, but the remainder of the rules are very similar to their counterparts in the natural deduction formulation of ILL.

Having outlined the form of the logic, we can see that it is easy to give a term assignment calculus. The introduction and elimination rules for ! are reflected in two new term constructs, an introduction construct $!t$ and an elimination construct $\mathsf{let}\ !x\ \mathsf{be}\ u\ \mathsf{in}\ t$, together forming a constructor-destructor pair.

In this chapter, we first present the logic DILL, and then give the type theory based upon it. We then present the type theory ILLand show its relation to the type theory based on DILL.

2.1 The Logic DILL

In order to present the logic, we assume a base set P_L of primitive propositions, ranged over by $l, k \ldots$, and then define *formulae*, ranged over by $A, B, C \ldots$:

$$A ::= l \in \mathsf{P}_L \mid I \mid A \otimes A \mid A \multimap A \mid !A$$

Now we define an *logical context* to be a pair of a set of formulae and a multiset of formulae, where we overload comma for the (set/multiset) union and $\_$ for the empty (set/multiset). We will write such a pair as $\Gamma;\Delta$, where the Γ is the *intuitionistic* part and Δ is the *linear* part.

We now assume a set C of formulae, the primitive assumptions. These are assumptions which we may use in a deduction with no further justification. We give the rules of the logic over P_L and C, which we call $\mathsf{D}(\mathsf{P}_L,\mathsf{C})$.

DEFINITION 2.1.1 (LOGICAL RULES OF $\mathrm{D}(\mathrm{P}_L,\mathrm{C})$)
We say that a sequent $\Gamma;\Delta \vdash A$ can be derived for a formula A in $\mathsf{D}(\mathsf{P}_L,\mathsf{C})$ if it can be shown using the following rules:

$$(Int - Ax)\ \Gamma, A; \_ \vdash A \qquad\qquad (Lin - Ax)\ \Gamma; A \vdash A$$

$$(Ass)\ \Gamma; \_ \vdash A\ (A \in \mathsf{C})$$

$$(I - I)\ \Gamma; \_ \vdash I \qquad\qquad \frac{\Gamma;\Delta_1 \vdash I \quad \Gamma;\Delta_2 \vdash A}{\Gamma;\Delta_1,\Delta_2 \vdash A}\ (I - E)$$

$$\frac{\Gamma;\Delta_1 \vdash A \quad \Gamma;\Delta_2 \vdash B}{\Gamma;\Delta_1,\Delta_2 \vdash A \otimes B}\ (\otimes - I) \qquad \frac{\Gamma;\Delta_1 \vdash A \otimes B \quad \Gamma;\Delta_2, A, B \vdash C}{\Gamma;\Delta_1,\Delta_2 \vdash C}\ (\otimes - E)$$

$$\frac{\Gamma;\Delta, A \vdash B}{\Gamma;\Delta \vdash A \multimap B}\ (\multimap I) \qquad \frac{\Gamma;\Delta_1 \vdash A \multimap B \quad \Gamma;\Delta_2 \vdash A}{\Gamma;\Delta_1,\Delta_2 \vdash B}\ (\multimap E)$$

$$\frac{\Gamma; \_ \vdash A}{\Gamma; \_ \vdash !A}\ (! - I) \qquad \frac{\Gamma;\Delta_1 \vdash !A \quad \Gamma, A;\Delta_2 \vdash B}{\Gamma;\Delta_1,\Delta_2 \vdash B}\ (! - E)$$

As remarked in the introduction, we have replaced the four rules involving the !-connective in the original form of ILL with the introduction-elimination pair seen above. The contraction and weakening rules previously used are now derivable by virtue of the fact that we allow contraction and weakening in the intuitionistic side of the context.

The Structural Rules

We can now give three structural rules for this logic: we have intuitionistic weakening and two flavours of cut, one intuitionistic and one linear.

$$\frac{\Gamma;\Delta \vdash B}{\Gamma, A;\Delta \vdash B}\ Weakening$$

$$\frac{\Gamma;\Delta_1 \vdash A \quad \Gamma;\Delta_2, A \vdash B}{\Gamma;\Delta_1,\Delta_2 \vdash B}\ (L - Cut) \qquad \frac{\Gamma; \_ \vdash A \quad \Gamma, A;\Delta \vdash B}{\Gamma;\Delta \vdash B}\ (I - Cut)$$

These three structural rules can be shown to be admissible, by which we mean that given a derivation of the premise of each rule, we can construct a derivation of its conclusion. Further, using these structural rules we can prove the lemma:

Lemma 2.1.2 (L-Transfer)
The following structural rule is admissible:

$$\frac{\Gamma; A, \Delta \vdash B}{\Gamma, A; \Delta \vdash B}$$

We now go on to state a number of defining equivalences for the connectives, which characterise their behaviour. One of the main motivations for this logic is the equivalence for ! presented here:

Lemma 2.1.3 (!-Equivalence)
In the presence of the structural rules, the two-way proof rule:

$$\begin{array}{c} \Gamma; !A, \Delta \vdash B \\ \hline\hline \Gamma, A; \Delta \vdash B \end{array}$$

is equivalent to the $(! - I), (! - E)$ pair introduced earlier, in the sense that in a system having the structural rules we would be able to derive the same sequents using the two way proof rule as we could with the introduction-elimination pair.

Proof If we have the two-way proof rule and the cut rules, then we can derive $! - I$:

$$\frac{\Gamma; \_ \vdash A \quad \dfrac{\_; !A \vdash !A}{A; \_ \vdash !A}}{\Gamma; \_ \vdash !A} \; (I - Cut)$$

Further, we can derive $! - E$:

$$\frac{\Gamma; \Delta_1 \vdash !A \quad \dfrac{\Gamma, A; \Delta_2 \vdash B}{\Gamma; !A, \Delta_2 \vdash B}}{\Gamma; \Delta_1, \Delta_2 \vdash B} \; (L - Cut)$$

Going the other way, it is easy to derive both directions of the equivalence given the $(! - I), (! - E)$ pair. The forward direction follows from one use of !-E, and the other direction is an instance of linear cut and weakening:

$$\frac{\dfrac{\Gamma, A; \_ \vdash A}{\Gamma, A; \_ \vdash !A} \quad \dfrac{\Gamma; !A, \Delta \vdash B}{\Gamma, A; !A, \Delta \vdash B}}{\Gamma, A; \Delta \vdash B}$$

□

We give similar results for the other connectives:

Lemma 2.1.4 (Equivalences)
In the presence of the structural rules:

- The two-way proof rule $\dfrac{\Gamma; \Delta, I \vdash C}{\Gamma; \Delta \vdash C}$ is equivalent to the $I - I, I - E$ pair.
- The two way proof rule $\dfrac{\Gamma; \Delta, A \otimes B \vdash C}{\Gamma; \Delta, A, B \vdash C}$ is equivalent to the $\otimes - I, \otimes - E$ pair.
- The two way proof rule $\dfrac{\Gamma; \Delta \vdash A \multimap B}{\Gamma; \Delta, A \vdash B}$ is equivalent to the $\multimap -I, \multimap -E$ pair.

These are all proved in a similar way to that for !.

2.2 The Type Theory $\mathsf{DILL}(\mathbb{C})$

We can now annotate the assumptions with variables, and obtain a type-theory from our logic. The types of the type theory (over primitive types P_L) are precisely the formulae of the logic (over primitive propositions P_L). We assume a countably infinite set of variables X ranged over by $x, y \ldots$ (which will be ubiquitous in this thesis), and we now assume a set of constants $\mathcal{C}$ ranged over by $c \ldots$, each of which has a type of DILL. We write $c{:}A$ to indicate that the constant c has the type A. We refer to the pair $(\mathsf{P}_L, \mathcal{C})$ as a *DILL-signature* , and let $\mathbb{C} \ldots$ range over DILL-signatures. We will also assume that $\mathbb{C} = (\mathsf{P}_L, \mathcal{C})$, $\mathbb{C}' = (\mathsf{P}'_L, \mathcal{C}')$ and similarly.

Definition 2.2.1 (Pre-Terms)
We define *pre-terms*, ranged over by $t, u \ldots$, as follows:

$$\begin{array}{rl} t ::= & x \mid c{:}A \mid \ * \ \mid \mathsf{let}\ * \ \mathsf{be}\ t\ \mathsf{in}\ t \mid t \otimes t \mid \mathsf{let}\ x \otimes x{:}A \otimes A\ \mathsf{be}\ t\ \mathsf{in}\ t \\ & \mid \lambda x{:}A.t \mid tt \mid !t \mid \mathsf{let}\ !x{:}A\ \mathsf{be}\ t\ \mathsf{in}\ t \end{array}$$

Having given pre-terms with type annotations, we omit them where possible for brevity. We define the usual capture-avoiding substitution $t\{u/x\}$, and also use a simultaneous form $t\{\vec{u}/\vec{x}\}$, where $\vec{u}$ is a sequence of pre-terms and $\vec{x}$ is a sequence of variables. In pre-terms, $\mathsf{let}\ x \otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u$ binds x and y in u, $\lambda x.t$ binds x in t, and $\mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ u$ binds x in u. We will identify pre-terms up to α-equivalence on bound variables. The multiset of free variables of a pre-term t, written $\mathsf{FV}(t)$, is defined inductively, where the mixed complement $M - S$ for M

a multiset and S a set removes all copies of anything in S from M:

$$\begin{aligned}
\mathsf{FV}(x) &= \{x\} \\
\mathsf{FV}(c) &= \emptyset \\
\mathsf{FV}(*) &= \emptyset \\
\mathsf{FV}(\mathsf{let}\ *\ \mathsf{be}\ u\ \mathsf{in}\ t) &= \mathsf{FV}(u) \cup \mathsf{FV}(t) \\
\mathsf{FV}(t \otimes u) &= \mathsf{FV}(t) \cup \mathsf{FV}(u) \\
\mathsf{FV}(\mathsf{let}\ x \otimes y\ \mathsf{be}\ u\ \mathsf{in}\ t) &= \mathsf{FV}(u) \cup (\mathsf{FV}(t) - \{x, y\}) \\
\mathsf{FV}(\lambda x.t) &= \mathsf{FV}(t) - \{x\} \\
\mathsf{FV}(tu) &= \mathsf{FV}(t) \cup \mathsf{FV}(u) \\
\mathsf{FV}(\mathsf{let}\ !x\ \mathsf{be}\ u\ \mathsf{in}\ t) &= \mathsf{FV}(u) \cup (\mathsf{FV}(t) - \{x\}) \\
\mathsf{FV}(!t) &= \mathsf{FV}(t)
\end{aligned}$$

Now we recall from appendix A the definitions of *typing* and *(dual) typing context* along with their auxiliary definitions, for the variable set X and the set of types of DILL.

We will now give a type theory with judgements of the form $\Gamma; \Delta \vdash t : A$, where $\Gamma; \Delta$ is a typing context, t is a pre-term and A is a type (or formula of the logic). We call this type-theory DILL($\mathbb{C}$).

In order to give the typing rules of DILL($\mathbb{C}$), we recall the merge relation $\Delta = \Delta_1 \# \Delta_2$, read as Δ is a merge of Δ_1 and Δ_2, from appendix A.

Definition 2.2.2 (The Typing Judgement of DILL($\mathbb{C}$))
The rules for deriving typing judgements $\Gamma; \Delta \vdash t : A$ are as follows, where $\Delta' = \Delta_1 \# \Delta_2$:

$(Int - Ax)\ \Gamma_1, x{:}A, \Gamma_2; \_ \vdash x{:}A$ $\qquad$ $(Lin - Ax)\ \Gamma; x{:}A \vdash x{:}A$

$(Ass)\ \Gamma; \_ \vdash c{:}A$

$(I - I)\ \Gamma; \_ \vdash *{:}I$ $\qquad$ $(I - E)\ \dfrac{\Gamma; \Delta_1 \vdash t{:}I \quad \Gamma; \Delta_2 \vdash u{:}A}{\Gamma; \Delta' \vdash \mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ u{:}A}$

$(\otimes - I)\ \dfrac{\Gamma; \Delta_1 \vdash t{:}A \quad \Gamma; \Delta_2 \vdash u{:}B}{\Gamma; \Delta' \vdash t \otimes u{:}A \otimes B}$ $\qquad$ $(\otimes - E)\ \dfrac{\Gamma; \Delta_1 \vdash u{:}A \otimes B \quad \Gamma; \Delta_2, x{:}A, y{:}B \vdash t{:}C}{\Gamma; \Delta' \vdash \mathsf{let}\ x \otimes y{:}A \otimes B\ \mathsf{be}\ u\ \mathsf{in}\ t{:}C}$

$(\multimap I)\ \dfrac{\Gamma; \Delta, x{:}A \vdash t{:}B}{\Gamma; \Delta \vdash (\lambda x{:}A.t){:}(A \multimap B)}$ $\qquad$ $(\multimap E)\ \dfrac{\Gamma; \Delta_1 \vdash u{:}A \multimap B \quad \Gamma; \Delta_2 \vdash t{:}A}{\Gamma; \Delta' \vdash (ut){:}B}$

$(! - I)\ \dfrac{\Gamma; \_ \vdash t{:}A}{\Gamma; \_ \vdash !t\ {:}!A}$ $\qquad$ $(! - E)\ \dfrac{\Gamma; \Delta_1 \vdash u\ {:}!A \quad \Gamma, x{:}A; \Delta_2 \vdash t{:}B}{\Gamma; \Delta' \vdash \mathsf{let}\ !x\ {:}!A\ \mathsf{be}\ u\ \mathsf{in}\ t{:}B}$

Definition 2.2.3 (Term)
A $\Gamma;\Delta$ term t of type A in $\mathsf{DILL}(\mathbb{C})$ for some typing context $\Gamma;\Delta$ and type A is a pre-term such that $\Gamma;\Delta \vdash t\!:\!A$ is a valid typing judgement. We will often indicate that a pre-term t is a $\Gamma;\Delta$ term of type A by writing $\Gamma;\Delta \vdash t : A$, or omit the context and typing where it is clear.

We now say that for a term $\Gamma;\Delta \vdash t\!:\!A$, a variable $x \in \mathsf{FV}(t)$ is an *intuitionistic* free variable of t if it occurs in occurs in $\mathsf{dom}(\Gamma)$, and $x \in \mathsf{FV}(t)$ is a *linear* free variable of t if it occurs in $\mathsf{dom}(\Delta)$.

Lemma 2.2.4 (Typing Properties)
We have the following in the type system $\mathsf{DILL}(\mathbb{C})$:

Free Variables I If $\Gamma;\Delta \vdash t\!:\!A$, then the underlying set of the multiset $\mathsf{FV}(t)$ is a subset of $\mathsf{dom}(\Gamma) \cup \mathsf{dom}(\Delta)$.

Free Variables II If $\Gamma;\Delta, x\!:\!A \vdash t\!:\!B$, then x occurs precisely once in $\mathsf{FV}(t)$.

Strengthening If $\Gamma, x\!:\!A;\Delta \vdash t\!:\!B$ and $x \notin \mathsf{FV}(t)$, then $\Gamma;\Delta \vdash t\!:\!B$.

I-Transfer If $\Gamma;\Delta, x\!:\!A \vdash t\!:\!B$, then $\Gamma, x\!:\!A;\Delta \vdash t\!:\!B$.

Unique Derivation Given a $\Gamma;\Delta$ term t of type A, there is a unique derivation of the typing judgement $\Gamma;\Delta \vdash t\!:\!A$.

The proofs of these properties are straightforward. Given the unique derivation lemma, we will interchangeably refer to terms and derivations of typing judgements.

We now present some admissible annotated structural rules, where $\Delta' = \Delta_1 \# \Delta_2$.

$$\frac{\Gamma, y\!:\!B, x\!:\!A, \Gamma' \vdash t\!:\!C}{\Gamma, x\!:\!A, y\!:\!B, \Gamma' \vdash t\!:\!C}\,(I - Exch) \qquad \frac{\Gamma;\Delta_1, y\!:\!B, x\!:\!A, \Delta_2 \vdash t\!:\!C}{\Gamma;\Delta_1, x\!:\!A, y\!:\!B, \Delta_2 \vdash t\!:\!C}\,(L - Exch)$$

$$\frac{\Gamma;\Delta \vdash t\!:\!B}{\Gamma, x\!:\!A;\Delta \vdash t\!:\!B}\,(Weak) \qquad \frac{\Gamma, x\!:\!A, y\!:\!A;\Delta \vdash t\!:\!B}{\Gamma, x\!:\!A;\Delta \vdash t\{x/y\}\!:\!B}\,(Cont)$$

$$\frac{\Gamma;\_ \vdash u\!:\!A \quad \Gamma, x\!:\!A;\Delta \vdash t\!:\!B}{\Gamma;\Delta \vdash t\{u/x\}\!:\!B}\,(I - Cut) \qquad \frac{\Gamma;\Delta_1 \vdash u\!:\!A \quad \Gamma;\Delta_2, x\!:\!A \vdash t\!:\!B}{\Gamma;\Delta' \vdash t\{u/x\}\!:\!B}\,(L - Cut)$$

These are shown to be admissible easily- notably, the linear exchange is shown by virtue of the merging we have incorporated into the typing rules. We outline the proof of the linear cut lemma. The proof proceeds by induction on the structure of t, and we consider only a few example cases in the proof; the rest are similar.

Proof: Note that since $x:A$ is a linear typing, x must occur precisely once in $\mathsf{FV}(t)$, by our lemma.

- If t is y then y is x, since x is free in y, and hence the required sequent is the second premise.
- If t is $\lambda y:B.t'$ then by considering the unique derivation, it must end:

  $$\frac{\Gamma; \Delta_1, x:A, y:B \vdash t':C}{\Gamma; \Delta_1, x:A \vdash (\lambda y:B.t'):B \multimap C}$$

  By the induction hypothesis on the premise, we now have:

  $$\Gamma; \Delta_1, \Delta_2, y':B \vdash t'\{y'/y, u/x\}:C$$

  where y' is a fresh variable not occuring in Δ_2, and the required sequent follows by abstraction (up to α-conversion).
- If t is $\mathsf{let}\ !y:C\ \mathsf{be}\ t_1\ \mathsf{in}\ t_2$ then we know that we have a derivation:

  $$\frac{\Gamma; \Delta_1' \vdash t_1 :!C \quad \Gamma, y:C; \Delta_1'' \vdash t_2:B}{\Gamma; \Delta_1 \vdash \mathsf{let}\ !y:C\ \mathsf{be}\ t_1\ \mathsf{in}\ t_2:B}$$

  where $\Delta_1 = \Delta_1' \# \Delta_1''$ and $x:A$ occurs in either Δ_1' or Δ_1''. In either case we can use the induction hypothesis to obtain the result required, possibly using α-conversion as before.

We now need to define a notion of (term-)contexts, where these may be linear (ie, use their 'argument' linearly) or intuitionistic (ie, use their argument inside a !-construct).

We define a general *pre-context*, written $C[\_]$, as follows:

$$\begin{aligned} C[\_] ::= \_ \mid & \mathsf{let} * \mathsf{be}\ C[\_]\ \mathsf{in}\ t \mid \mathsf{let} * \mathsf{be}\ t\ \mathsf{in}\ C[\_] \mid t \otimes C[\_] \mid C[\_] \otimes t \mid \\ & \mathsf{let}\ x \otimes x\ \mathsf{be}\ C[\_]\ \mathsf{in}\ t \mid \mathsf{let}\ x \otimes x\ \mathsf{be}\ t\ \mathsf{in}\ C[\_] \mid \lambda x.C[\_] \mid C[\_]t \mid tC[\_] \mid \\ & !C[\_] \mid \mathsf{let}\ !x\ \mathsf{be}\ C[\_]\ \mathsf{in}\ t \mid \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ C[\_] \end{aligned}$$

It is easily shown there will be precisely one occurrence of the symbol $\_$ in every pre-context. We extend the definition of free variable multiset to pre-contexts $\mathsf{FV}(C[\_])$ in the obvious way by saying $\mathsf{FV}(\_) = \emptyset$.

Definition 2.2.5 (Instantiation of Pre-contexts)
Define $C[t]$ for a given pre-context $C[\_]$ and pre-term t to be the pre-context $C[\_]$ with the unique occurrence of the symbol $\_$ replaced by t. We can easily show by induction over contexts that $C[t]$ is a pre-term.

Definition 2.2.6 (Context)
A $\Gamma;\Delta,A$-$\Gamma';\Delta',B$ context is a pre-context $C[\_]$ such that for any term $\Gamma;\Delta\vdash t$: A, we have a derivation of the judgement $\Gamma';\Delta'\vdash C[t]\colon B$.

Although this definition is not constructive, it can be formulated in an equivalent but much more lengthy constructive form, using a system of judgements of the form $\Gamma;\Delta\vdash C:\Gamma;\Delta,A-\Gamma';\Delta',B$. For the pre-contexts we will claim are contexts, it will be easy to see that the definition above is satisfied.

Definition 2.2.7 (Linear and Binding Contexts)
We say that a context is *linear* if it does not contain the clause $!C[\_]$. Further, a context *binds* a variable x if the context is constructed with the use of a clause instance $\mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ C[\_]$, $\mathsf{let}\ y\otimes x\ \mathsf{be}\ t\ \mathsf{in}\ C[\_]$, $\lambda x.C[\_]$ or $\mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ C[\_]$.

Now we can define our contextual equality judgement.

Definition 2.2.8 (The Equality Judgement)
We say that two terms $\Gamma;\Delta\vdash t\colon A$ and $\Gamma;\Delta\vdash u\colon A$ are provably equal of type A in an environment $\Gamma;\Delta$, written $\Gamma;\Delta\vdash t=u\colon A$, if their equality is provable using the following rules:

$$\frac{\Gamma;\Delta\vdash t\colon A}{\Gamma;\Delta\vdash t=t\colon A}\qquad\frac{\Gamma;\Delta\vdash t=t'\colon A\quad\Gamma;\Delta\vdash t'=t''\colon A}{\Gamma;\Delta\vdash t=t''\colon A}\qquad\frac{\Gamma;\Delta\vdash t=u\colon A}{\Gamma;\Delta\vdash u=t\colon A}$$

$$\frac{\Gamma;\Delta\vdash t=u\colon A}{\Gamma';\Delta'\vdash C[t]=C[u]\colon B}$$

where $C[\_]$ is a $\Gamma;\Delta,A$-$\Gamma';\Delta',B$-context.

(β)	(η)
$\Gamma;\Delta\vdash \mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ t=t\colon A$	$\Gamma;\Delta\vdash \mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ *=t\colon I$
$\Gamma;\Delta\vdash \mathsf{let}\ x\otimes y\ \mathsf{be}\ u_1\otimes u_2\ \mathsf{in}\ t=t\{u_1/x,u_2/y\}\colon A$	$\Gamma;\Delta\vdash \mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ x\otimes y=t\colon A\otimes B$
$\Gamma;\Delta\vdash(\lambda x\colon A.t)u=t\{u/x\}\colon A$	$\Gamma;\Delta\vdash\lambda x\colon A.(tx)=t\colon A\multimap B$ where x is not free in t
$\Gamma;\Delta\vdash \mathsf{let}\ !x\colon A\ \mathsf{be}\ !u\ \mathsf{in}\ t=t\{u/x\}\colon A$	$\Gamma;\Delta\vdash \mathsf{let}\ !x\colon A\ \mathsf{be}\ t\ \mathsf{in}\ !x=t\ \colon !A$

In these, C is supposed to be a linear context.

$$\Gamma;\Delta\vdash \mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ C[u]=C[\mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ u]\colon A$$

$$\Gamma;\Delta\vdash \mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ C[u]=C[\mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u]\colon A$$

$$\Gamma;\Delta\vdash \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ C[u]=C[\mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ u]\colon A$$

where $C[\_]$ does not bind x or contain it free

These commuting conversions are the equality judgements which correspond to the trivial proof permutations which exist in linear logics. In other presentations of term calculi for ILL, the proof permutations are expressed as a large number of primitive typed equality judgements such as:

$$\Gamma;\Delta \vdash \mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u_1\otimes u_2 = u_1\otimes(\mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u_2){:}A\otimes B$$

where x, y are not free in u.

These many individual typed equalities all occur as examples of one of our commuting conversion schemas. For example, the typed equality above would be an instance of our second commuting conversion axiom, where the context C is $v\otimes \_$.

Note 2.2.9 (Commuting Conversions of $! - E$)
Notice that the restriction on the context in the !-commuting conversion, means that it does not have as an instance the following equality judgement:

$$y:!A; \_ \vdash !(\mathsf{let}\ !x\ \mathsf{be}\ y\ \mathsf{in}\ x) = \mathsf{let}\ !x\ \mathsf{be}\ y\ \mathsf{in}\ !x{:}A$$

This cannot be motivated from the proof structure of linear logic, and it corresponds to imposing an extra requirement on models (idempotency of the comonad !).

From now on, we will assume that whenever we write $t = u$, there exists an appropriate equality judgement $\Gamma;\Delta\vdash t = u{:}A$.

Equivalences

We can now prove a lemma extending our results on equivalences to the typed case.

Lemma 2.2.10 (Equivalences)

- If $\Gamma;\Delta, x:I \vdash t:A$, then $\Gamma;\Delta \vdash t\{*/x\}:A$, and if $\Gamma;\Delta\vdash u:A$, then $\Gamma; x:I,\Delta \vdash \mathsf{let}\ *\ \mathsf{be}\ x\ \mathsf{in}\ u:A$, where x is fresh. Further, these maps are inverse up to provable term equality.

- If $\Gamma;\Delta, x_1:A, x_2:B \vdash t{:}C$, then $\Gamma;\Delta, y:A\otimes B \vdash \mathsf{let}\ x_1\otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ t{:}C$ where y is fresh, and if $\Gamma;\Delta, y:A\otimes B\vdash u{:}C$, then $\Gamma;\Delta, x_1:A, x_2:B\vdash u\{x_1\otimes x_2/y\}{:}C$ where x_1 and x_2 are fresh. Further, these maps are inverse up to provable term equality.

- If $\Gamma;\Delta \vdash t:A\multimap B$, then $\Gamma;\Delta, x:A\vdash tx:B$ where x is fresh, and if $\Gamma;\Delta, x{:}A\vdash u{:}B$, then $\Gamma;\Delta\vdash \lambda x{:}A.u{:}A\multimap B$. Further, these maps are inverse up to provable term equality.

- If $\Gamma; x :!A, \Delta \vdash t : B$, then $\Gamma, y : A; \Delta \vdash t\{!y/x\} : B$ where y is fresh, and if $\Gamma, y : A; \Delta \vdash u : B$, then $\Gamma; x :!A, \Delta \vdash \mathsf{let}\ !y\ \mathsf{be}\ x\ \mathsf{in}\ u : B$ where x is fresh. Further, these maps are inverse up to provable term equality.

It is interesting to note that we could use these equivalences to define the term equality. If we consider the typing system without any notion of equality judgement, and read the equivalences above as isomorphisms on proofs, they say the following:

$$\text{Proofs } \Gamma; \Delta, x{:}I \vdash t{:}A \text{ are isomorphic to proofs } \Gamma; \Delta, x{:}I \vdash \mathsf{let} * \mathsf{be}\ x\ \mathsf{in}\ t\{*/x\}{:}A$$
$$\text{Proofs } \Gamma; \Delta \vdash u{:}A \text{ are isomorphic to proofs } \Gamma; \Delta \vdash \mathsf{let} * \mathsf{be} * \mathsf{in}\ u{:}A$$

$$\text{Proofs } \Gamma; \Delta, x_1{:}A, x_2{:}B \vdash t{:}C \text{ are isomorphic to proofs}$$
$$\Gamma; \Delta, x_1{:}A, x_2{:}B \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ t{:}C$$
$$\text{Proofs } \Gamma; \Delta, y{:}A \otimes B \vdash u{:}C \text{ are isomorphic to proofs}$$
$$\Gamma; \Delta, y{:}A \otimes B \vdash \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ u\{x_1 \otimes x_2/y\}{:}C$$

$$\text{Proofs } \Gamma; \Delta, x{:}A \vdash t{:}B \text{ are isomorphic to proofs } \Gamma; \Delta, x{:}A \vdash (\lambda x.t)x{:}B$$
$$\text{Proofs } \Gamma; \Delta \vdash u{:}A \multimap B \text{ are isomorphic to proofs } \Gamma; \Delta \vdash \lambda x.(ux){:}A \multimap B$$

$$\text{Proofs } \Gamma, x{:}A; \Delta \vdash t{:}B \text{ are isomorphic to proofs } \Gamma, x{:}A; \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ !x\ \mathsf{in}\ t{:}B$$
$$\text{Proofs } \Gamma; y :!A, \Delta \vdash u{:}B \text{ are isomorphic to proofs}$$
$$\Gamma; y :!A, \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ y\ \mathsf{in}\ u\{!x/y\}{:}B$$

Now the alternative equality judgement with the symmetric, transitivity, reflexivity and congruence rules already defined, with the addition of two cut rules:

$$\frac{\Gamma; \Delta_1 \vdash u{:}A \quad \Gamma; x{:}A, \Delta_2 \vdash t = t'{:}B}{\Gamma; \Delta' \vdash t\{u/x\} = t'\{u/x\}{:}B} \qquad \frac{\Gamma; \_ \vdash u{:}A \quad \Gamma, x{:}A; \Delta \vdash t = t'{:}B}{\Gamma; \Delta \vdash t\{u/x\} = t'\{u/x\}{:}B}$$

(where $\Delta' = \Delta_1 \# \Delta_2$) and with the axiomatic equalities induced by the proof isomorphisms listed above is equivalent in strength to the equality judgement first defined.

A Definable Intuitionistic Function Space

We now briefly present a definable extension to the logic and term calculus given above. In order to make the syntax more usable, we show how we can define the types and terms associated with an intuitionistic arrow type purely in terms of the structures we already have.

Definition 2.2.11 (The Intuitionistic Arrow)

Types Define the type $A \to B$ in our new system as the type $!A \multimap B$.

Terms Define the intuitionistic abstraction and application term constructs as follows (where we use γ for the abstraction, and y is fresh):

$$\gamma x{:}A.t \quad = \quad \lambda y :!A\ \mathsf{let}\ !x :!A\ \mathsf{be}\ y\ \mathsf{in}\ t$$

$$tu \quad = \quad t(!u)$$

Now we have the following lemma:

Lemma 2.2.12 (Typing and Equality Rules for $\to$)
The following introduction and elimination rules are admissible in $\mathsf{DILL}(\mathbb{C})$:

$$\frac{\Gamma, x{:}A; \Delta \vdash t{:}B}{\Gamma; \Delta \vdash \gamma x{:}A.t{:}A \to B} \to -I \qquad \frac{\Gamma; \Delta \vdash t{:}A \to B \quad \Gamma; \_ \vdash u{:}A}{\Gamma; \Delta \vdash tu{:}B} \to -E$$

The following equality judgements are admissible in $\mathsf{DILL}(\mathbb{C})$:

$$\Gamma; \Delta \vdash (\gamma x{:}A.t)u = t[u/x]{:}A \qquad \Gamma; \Delta \vdash \gamma x{:}A.(tx) = t{:}A \to B$$

Clearly we can erase the typings from the introduction and elimination rules to obtain admissible introduction and elimination rules for $\to$ in the logic $\mathsf{DILL}(\mathbb{C})$.

This construct and its associated terms and equalities can by virtue of this lemma be used without changing the development to follow. Other definitions of this function space are possible via more complex embeddings of intuitionistic logic into linear logic; for example see Benton [BW96] or Schellinx [Sch94].

2.3 The Type Theory $\mathsf{ILL}(\mathbb{C})$

We present the type-theory based on ILL due to Benton *et al.* [BBdPH93b], with two amendments. Firstly, we build the type theory over a dill-signature, in the analogous way to DILL, and secondly we have adapted the presentation of the equality in *op. cit.* using contexts to express the commuting conversions, and giving an equality judgement. For brevity we do not present the logic; this can be found in [BBdPH93b].

We assume the same set X of variables as used in $\mathsf{DILL}(\mathbb{C})$. The types of $\mathsf{ILL}(\mathbb{C})$ are exactly those of $\mathsf{DILL}(\mathbb{C})$. Given this, we define pre-terms, ranged over by $M, N \ldots$ as follows:

$$\begin{aligned} M ::= {} & x \mid c{:}A \mid * \mid \mathsf{let}\ * \ \mathsf{be}\ M\ \mathsf{in}\ M \mid M \otimes M \mid \mathsf{let}\ x \otimes x{:}A \otimes A\ \mathsf{be}\ M\ \mathsf{in}\ M \mid \\ & \lambda x{:}A.M \mid MM \mid \mathsf{promote}\ \vec{M}\ \mathsf{for}\ x :!A \ldots x :!A\ \mathsf{in}\ M \mid \\ & \mathsf{derelict}(M) \mid \mathsf{copy}\ M\ \mathsf{for}\ x, x :!A\ \mathsf{in}\ M \mid \mathsf{discard}\ M\ \mathsf{in}\ M \end{aligned}$$

As before, we will omit type information in pre-terms where convenient. Following Benton *et al.*, we use the form ($\mathsf{discard}\ \vec{M}\ \mathsf{in}\ N$) to abbreviate

$$(\mathsf{discard}\ M_1\ \mathsf{in}\ \ldots\ \mathsf{in}\ \mathsf{discard}\ M_r\ \mathsf{in}\ N)$$

We also use ($\mathsf{copy}\ \vec{M}\ \mathsf{for}\ \vec{x}, \vec{y}\ \mathsf{in}\ N$) to abbreviate

$$(\mathsf{copy}\ M_1\ \mathsf{for}\ x_1, y_1\ \mathsf{in}\ \ldots\ \mathsf{copy}\ M_r\ \mathsf{for}\ x_r, y_r\ \mathsf{in}\ N)$$

We now summarise the binding behaviour of the pre-terms: the pre-terms $\mathsf{let}\ x \otimes y\ \mathsf{be}\ M\ \mathsf{in}\ N$ and $\mathsf{copy}\ M\ \mathsf{for}\ x, y\ \mathsf{in}\ N$ bind x and y in N; the pre-term $\lambda x.M$ binds x in M and the pre-term $\mathsf{promote}\ \vec{M}\ \mathsf{for}\ \vec{X}\ \mathsf{in}\ N$ bind each of the $\vec{x}$ in N. Again, we will require a standard capture-avoiding notion of substitution $M\{N/x\}$, and its simultaneous counterpart $t\{\vec{u}/\vec{x}\}$.

Now we define an ILL *typing context* as a single typing context over the variable set X and the types of $\mathsf{DILL}(\mathbb{C})$. We give the contextual typing judgement $\Delta \vdash M{:}A$ as follows:

Definition 2.3.1 (The Typing Judgement)
The rules for deriving judgements $\Delta \vdash M{:}A$ are as follows, where $\Delta' = \Delta_1\#\Delta_2$ and $\Delta'' = \Delta_1\#\ldots\#\Delta_r$:

$$(Lin - Ax)\ x{:}A \vdash x{:}A \qquad\qquad (Ass)\ \_ \vdash c{:}A$$

$$(I - I)\ \_ \vdash *{:}I \qquad\qquad (I - E)\ \frac{\Delta_1 \vdash M{:}I \quad \Delta_2 \vdash N{:}A}{\Delta' \vdash \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ N{:}A}$$

$$(\otimes - I)\ \frac{\Delta_1 \vdash M{:}A \quad \Delta_2 \vdash N{:}B}{\Delta' \vdash M \otimes N{:}A \otimes B} \qquad (\otimes - E)\ \frac{\Delta_1 \vdash N{:}A \otimes B \quad \Delta_2, x{:}A, y{:}B \vdash M{:}C}{\Delta' \vdash \mathsf{let}\ x \otimes y{:}A \otimes B\ \mathsf{be}\ N\ \mathsf{in}\ M{:}C}$$

$$(\multimap I)\ \frac{\Delta, x{:}A \vdash M{:}B}{\Delta \vdash (\lambda x{:}A.M){:}(A \multimap B)} \qquad (\multimap E)\ \frac{\Delta_1 \vdash M{:}A \multimap B \quad \Delta_2 \vdash N{:}A}{\Delta' \vdash (MN){:}B}$$

$$(\mathrm{Weak})\ \frac{\Delta_1 \vdash M\ {:}!A \quad \Delta_2 \vdash N{:}B}{\Delta' \vdash \mathsf{discard}\ M\ \mathsf{in}\ N{:}B} \qquad (\mathrm{Der})\ \frac{\Delta \vdash M\ {:}!A}{\Delta \vdash \mathsf{derelict}(M){:}A}$$

$$\mathrm{Contraction}\ \frac{\Delta_1 \vdash M\ {:}!A \quad \Delta_2, x\ {:}!A, y\ {:}!A \vdash N{:}B}{\Delta' \vdash \mathsf{copy}\ M\ \mathsf{as}\ x, y\ \mathsf{in}\ N{:}B}$$

$$\mathrm{Promotion}\ \frac{i = 1..r \quad \Delta_i \vdash M_i\ {:}!A_i \quad x_1\ {:}!A_1, \ldots, x_r\ {:}!A_r \vdash N{:}B}{\Delta'' \vdash \mathsf{promote}\ \vec{M}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ N\ {:}!B}$$

Definition 2.3.2 (Term)
We define a Δ term t of type A in $\mathsf{ILL}(\mathbb{C})$ to be a pre-term t of $\mathsf{ILL}(\mathbb{C})$ such that we can derive the typing judgement $\Delta \vdash t{:}A$.

We can now give admissible exchange and linear cut rules.

Definition 2.3.3 (Pre-contexts)
We define pre-contexts for $\mathsf{ILL}(\mathbb{C})$ inductively as follows:

$$\begin{aligned} C[\_] ::= \_ \mid & \ \mathsf{let} * \mathsf{be}\ C[\_]\ \mathsf{in}\ t \mid \mathsf{let} * \mathsf{be}\ t\ \mathsf{in}\ C[\_] \mid t \otimes C[\_] \mid C[\_] \otimes t \mid \\ & \mathsf{let}\ x \otimes x\ \mathsf{be}\ C[\_]\ \mathsf{in}\ t \mid \mathsf{let}\ x \otimes x\ \mathsf{be}\ t\ \mathsf{in}\ C[\_] \mid \lambda x.C[\_] \mid C[\_]t \mid tC[\_] \mid \\ & \mathsf{promote}\ \vec{M}_1, C[\_], \vec{M}_2\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ N \mid \mathsf{promote}\ \vec{M}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ C[\_] \\ & \mathsf{copy}\ C[\_]\ \mathsf{for}\ x, y\ \mathsf{in}\ N \mid \mathsf{copy}\ M\ \mathsf{for}\ x, y\ \mathsf{in}\ C[\_] \mathsf{derelict}(C[\_]) \\ & \mathsf{discard}\ C[\_]\ \mathsf{in}\ N \mid \mathsf{discard}\ M\ \mathsf{in}\ C[\_] \end{aligned}$$

We define the *instantiation* of a pre-context $C[\_]$ with a pre-term M again to be the pre-context with the unique occurrence of the symbol $\_$ replaced by the pre-term M. We say that any pre-context constructed with no instance of the schema $\mathsf{promote}\ \vec{M}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ C[\_]$ is *linear*.

Definition 2.3.4 (Context)
We define a Δ, A-Δ', B' context to be a pre-context $C[\_]$ such that for any term $\Delta \vdash t{:}A$, we have $\Delta' \vdash C[t]{:}B$.

Again, this definition of contexts is not inductive, but as in the case of contexts for $\mathsf{DILL}(\mathbb{C})$, it could easily be equivalently given inductively.

Definition 2.3.5 (The Equality Judgement)
We define a contextual equality judgement $\Delta \vdash M = N{:}A$, where M and N are Δ terms of type A, by the following rules (where for brevity we write $M = N$ for $\Delta \vdash M = N{:}A$):

The Congruence Rules

$$\frac{\Delta \vdash M{:}A}{\Delta \vdash M = M{:}A} \qquad \frac{\Delta \vdash M = M'{:}A \quad \Delta \vdash M' = M''{:}A}{\Delta \vdash M = M''{:}A} \qquad \frac{\Delta \vdash M = N{:}A}{\Delta \vdash N = M{:}A}$$

$$\frac{\Delta \vdash t = u{:}A}{\Delta' \vdash C[t] = C[u]{:}B}$$

where $C[\_]$ is a Δ, A-Δ', B-context.

The Basic Equalities

$$\mathsf{let} * \mathsf{be} * \mathsf{in}\ M = M \tag{2.1}$$

$$\mathsf{let} * \mathsf{be}\ M\ \mathsf{in} * = M \tag{2.2}$$

$$\mathsf{let}\ x \otimes y\ \mathsf{be}\ M_1 \otimes M_2\ \mathsf{in}\ N = N\{M_1/x, M_2/y\} \tag{2.3}$$

$$\mathsf{let}\ x \otimes y\ \mathsf{be}\ M\ \mathsf{in}\ x \otimes y = M \tag{2.4}$$

$$(\lambda x.M)N = M\{N/x\} \tag{2.5}$$

$$\lambda x.(Mx) = M \tag{2.6}$$

The Commuting Conversions For linear $C[\_]$,

$$\text{let } * \text{ be } M \text{ in } C[N] = C[\text{let } * \text{ be } M \text{ in } N] \tag{2.7}$$
$$\text{let } x \otimes y \text{ be } M \text{ in } C[N] = C[\text{let } x \otimes y \text{ be } M \text{ in } N] \tag{2.8}$$
$$\text{discard } M \text{ in } C[N] = C[\text{discard } M \text{ in } N] \tag{2.9}$$
$$\text{copy } M \text{ for } x, y \text{ in } C[N] = C[\text{copy } M \text{ for } x, y \text{ in } N] \tag{2.10}$$

The Exponential Equalities

$$\text{derelict (promote } \vec{M} \text{ for } \vec{x} \text{ in } N) = N\{\vec{M}/\vec{x}\} \tag{1}$$
$$\text{discard (promote } \vec{M} \text{ for } \vec{x} \text{ in } N) \text{ in } N' = \text{discard } \vec{M} \text{ in } N' \tag{2}$$
$$\text{promote } M \text{ for } x \text{ in derelict } (x) = M \tag{3}$$
$$\text{copy } M \text{ for } x, y \text{ in } N = \text{copy } M \text{ for } y, x \text{ in } N \tag{4}$$
$$\text{copy } M \text{ for } x, y \text{ in (discard } x \text{ in } N) = N\{M/y\} \tag{5}$$
$$\text{copy } M \text{ for } x, y \text{ in copy } y \text{ for } x', y' \text{ in } N = \text{copy } M \text{ for } y, y' \text{ in copy } y \text{ for } x, x' \text{ in } N \tag{6}$$
$$\begin{aligned}\text{promote } M', \vec{M} \text{ for } x', \vec{x} &= \text{discard } M' \text{ in (promote } \vec{M} \text{ for } \vec{x} \text{ in } N) \\ \text{in (discard } x' \text{ in } N)&\end{aligned} \tag{7}$$
$$\begin{aligned}\text{copy (promote } \vec{M} \text{ for } \vec{x} \text{ in } N) &= \text{copy } \vec{M} \text{ for } \vec{x}_1, \vec{x}_2 \text{ in } N'\{p_1, p_2/y_1, y_2\} \\ \text{for } y_1, y_2 \text{ in } N'&\end{aligned} \tag{8}$$

(where $p_1 = \text{promote } \vec{x}_1 \text{ for } \vec{x} \text{ in } N$ **and** $p_2 = \text{promote } \vec{x}_2 \text{ for } \vec{x} \text{ in } N$**)**

$$\begin{aligned}\text{promote } M', \vec{M} \text{ for } x', \vec{x} &= \text{copy } M' \text{ for } y_1', y_2' \text{ in } p_3 \\ \text{in (copy } x' \text{ for } y_1, y_2 \text{ in } N)&\end{aligned} \tag{9}$$

(where $p_3 = \text{promote } \vec{M}, y_1', y_2' \text{ for } \vec{x}, y_1, y_2 \text{ in } N$**)**

$$\begin{aligned}\text{promote (promote } \vec{y_1} \text{ for } \vec{x}_1 \text{ in } M), \vec{y_2} &= \text{promote } \vec{y_1}, \vec{y_2} \text{ for } \vec{y_3}, \vec{x}_2 \text{ in } N\{p_4/x'\} \\ \text{for } x', \vec{x}_2 \text{ in } N&\end{aligned} \tag{10}$$

(where $p_4 = \text{promote } \vec{y_3} \text{ for } \vec{x}_1 \text{ in } M$**)**

2.4 Relating DILL and ILL

We now show that DILL is essentially equivalent to ILL at the level of the respective type-theories, by which we mean firstly that we can give a translation from DILL($\mathbb{C}$) to the type-theory ILL($\mathbb{C}$), and one in the reverse direction, such that two terms are equal in ILL($\mathbb{C}$) if and only if their images are equal in DILL($\mathbb{C}$)and secondly that we can give a translation from the terms of DILL($\mathbb{C}$) derivable with no intuitionistic assumptions, $\_;\Delta \vdash t : A$, to the terms of ILL($\mathbb{C}$). Wherever numbers are used to refer to particular equalities of ILL($\mathbb{C}$), those numbers are as given in section 2.3. Of course these translations easily yield translations on the logics DILL and ILL.

In the following, we will use the abbreviation ($\mathsf{let}\ !\vec{x}\ \mathsf{be}\ \vec{u}\ \mathsf{in}\ t$) to indicate the term

$$\mathsf{let}\ !x_1\ \mathsf{be}\ u_1\ \mathsf{in} \ldots \mathsf{let}\ !x_r\ \mathsf{be}\ u_r\ \mathsf{in}\ t$$

in $\mathsf{DILL}(\mathbb{C})$. We use the turnstile $\vdash_{\mathsf{ILL}(\mathbb{C})}$ to indicate the typing and equality judgements of $\mathsf{ILL}(\mathbb{C})$ and similarly the turnstile $\vdash_{\mathsf{DILL}(\mathbb{C})}$ to indicate those of $\mathsf{DILL}(\mathbb{C})$ where the distinction is unclear.

We will define two translations, Φ taking $\mathsf{ILL}(\mathbb{C})$ to $\mathsf{DILL}(\mathbb{C})$, and Ψ taking $\mathsf{DILL}(\mathbb{C})$ to $\mathsf{ILL}(\mathbb{C})$. These translations will be the identity on types.

Lemma 2.4.1 (Properties of the Translations)
The following are properties of Φ and Ψ:

1. If $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M{:}A$, then $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(M){:}A$
2. If $x_1{:}A_1 \ldots x_r{:}A_r;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} u{:}A$, then $x_1 :!A_1 \ldots x_r :!A_r, \Delta \vdash_{\mathsf{ILL}(\mathbb{C})} \Psi_{\vec{x}}(u){:}A$
3. If $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M = N{:}A$, then $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(M) = \Phi(N){:}A$
4. If $x_1 : A_1 \ldots x_r : A_r;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} t = u : A$, then $x_1 :!A_1 \ldots x_r : A_r, \Delta \vdash_{\mathsf{ILL}(\mathbb{C})} \Psi_{\vec{x}}(t) = \Psi_{\vec{x}}(u){:}A$
5. If $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M{:}A$, then $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} \Psi_{\varepsilon}(\Phi(M)) = M{:}A$
6. If $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} t{:}A$, then $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(\Psi_{\varepsilon}(t)) = t{:}A$

Given these lemmas, we can prove:

Theorem 1 (Equivalence)

1. $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M = N{:}A$ iff $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(M) = \Phi(N){:}A$
2. $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} t = u{:}A$ iff $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} \Psi_{\varepsilon}(t) = \Psi_{\varepsilon}(u){:}A$

Proof The two proofs are almost identical. Consider the first case. We already have the implication

$$\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M = N{:}A \text{ implies } \_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(M) = \Phi(N){:}A$$

Now assume $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(M) = \Phi(N) : A$. By lemma 2.4.1 (4), we have that $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} \Psi_{\varepsilon}(\Phi(M)) = \Psi_{\varepsilon}(\Phi(N)){:}A$, but we also have by lemma 2.4.1 (5) that

$\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} \Psi_\varepsilon(\Phi(M)) = M : A$, and hence the other direction of the implication holds.

The proof of the second case uses the analogous results in lemma 2.4.1 (3) and 2.4.1 (6) □

We now proceed to give the details of the translations, and prove lemma 2.4.1, thereby showing that the results of theorem 1 hold.

From Intuitionistic Linear Logic to DILL

Now we can define the translation Φ from the intuitionistic linear type theory $\mathsf{ILL}(\mathbb{C})$ to $\mathsf{DILL}(\mathbb{C})$.

DEFINITION 2.4.2 (THE TRANSLATION Φ)
On Types we define Φ to be the identity.
On Pre-Terms we define Φ as follows:

$$\begin{array}{ccc}
\Phi(x) & = & x \\
\Phi(c) & = & c \\
\Phi(*) & = & * \\
\Phi(\mathsf{let}\ * \ \mathsf{be}\ M\ \mathsf{in}\ N) & = & \mathsf{let}\ *\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N) \\
\Phi(M \otimes N) & = & \Phi(M) \otimes \Phi(N) \\
\Phi(\mathsf{let}\ x \otimes y\ \mathsf{be}\ M\ \mathsf{in}\ N) & = & \mathsf{let}\ x \otimes y\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N) \\
\Phi(\lambda x.M) & = & \lambda x.\Phi(M) \\
\Phi(MN) & = & \Phi(M)\Phi(N) \\
\Phi(\mathsf{discard}\ M\ \mathsf{in}\ N) & = & \mathsf{let}\ !x_1\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N) \\
\Phi(\mathsf{copy}\ M\ \mathsf{for}\ x, y\ \mathsf{in}\ N) & = & \mathsf{let}\ !x_1\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N)[!x_1/x, y] \\
\Phi(\mathsf{derelict}(M)) & = & \mathsf{let}\ !x_1\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ x_1 \\
\Phi(\mathsf{promote}\ \vec{M}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ N) & = & \mathsf{let}\ !\vec{x}_1\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ !(\Phi(N)[!\vec{x}_1/\vec{x}])
\end{array}$$

where in the last four rules x_1 is a fresh variable from X.

Now we need to prove the lemma.

LEMMA 2.4.1 (1) *If $\Delta \vdash_{\mathit{ILL}(\mathbb{C})} M : A$, then $\_; \Delta \vdash_{\mathit{DILL}(\mathbb{C})} \Phi(M) : A$.*

Proof This proof is by induction over the structure of the term t. We give a summary proof only.

Axiom Instance: In this case, we have $x : A \vdash x : A$, so that the translation is $\_; x : A \vdash x : A$, which is derivable.

Assumption and Unit-I In these cases there is almost nothing to show, as the corresponding typing rules in $\mathsf{DILL}(\mathbb{C})$ are analogous. We note the requirement that we have the same constants in each type theory.

⊗-Introduction: We present this case as an example of the simple cases not involving the exponential. We have the derivation

$$\frac{\Delta_1 \vdash M\!:\!A \quad \Delta_2 \vdash N\!:\!B}{\Delta \vdash M \otimes N\!:\!A \otimes B}\,(\otimes - I)$$

where $\Delta = \Delta_1 \# \Delta_2$.

By the inductive hypothesis, we have $\_; \Delta_1 \vdash \Phi(M)\!:\!A$, and $\_; \Delta_2 \vdash \Phi(N)\!:\!A$, so we have $\_; \Delta \vdash \Phi(M) \otimes \Phi(N)\!:\!A \otimes B$ via the ⊗-introduction rule of $\mathsf{DILL}(\mathbb{C})$. But $\Phi(M \otimes N) = \Phi(M) \otimes \Phi(N)$, so we are done.

⊗-E,⊸-I and ⊸-E: Again these rules in $\mathsf{ILL}(\mathbb{C})$ are exactly parallelled in $\mathsf{DILL}(\mathbb{C})$.

Weakening Rule: In this case, we have the derivation

$$\frac{\Delta_2 \vdash M :!A \quad \Delta_1 \vdash N\!:\!B}{\Delta \vdash \mathsf{discard}\ M\ \mathsf{in}\ N\!:\!B}$$

where $\Delta = \Delta_1 \# \Delta_2$.

By intuitionistic weakening and our inductive hypothesis, in $\mathsf{DILL}(\mathbb{C})$ we have $x_1 : A; \Delta_1 \vdash \Phi(N) : B$. Now one application of our !-E rule gives us $\_; \Delta \vdash \mathsf{let}\ !x_1\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N) : B$, but $\Phi(\mathsf{discard}\ M\ \mathsf{in}\ N)$ is precisely $\mathsf{let}\ !x_1\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N)$.

Contraction: In this case, we have the derivation in $\mathsf{ILL}(\mathbb{C})$:

$$\frac{\Delta_1 \vdash M :!A \quad \Delta_2, x :!A, y :!A \vdash N\!:\!B}{\Delta \vdash \mathsf{copy}\ M\ \mathsf{for}\ x, y\ \mathsf{in}\ N\!:\!B}$$

where $\Delta = \Delta_1 \# \Delta_2$.

Now by the inductive hypothesis, in $\mathsf{DILL}(\mathbb{C})$ we have the derivations:

$$\_; \Delta_1 \vdash \Phi(M) :!A$$

and

$$\_; \Delta_2, x :!A, y :!A \vdash \Phi(N)\!:\!B$$

But now, using the substitution lemmas of $\mathsf{DILL}(\mathbb{C})$ and the !-I,E pair, we have the following derivation:

$$\_; \Delta \vdash \mathsf{let}\ !x_1\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N)\{!x_1/x, y\}\!:\!B$$

which proves the case.

Dereliction: In this case we have the derivation

$$\frac{\Delta \vdash M : !A}{\Delta \vdash \mathsf{derelict}\ M : A}$$

in $\mathsf{ILL}(\mathbb{C})$. Using the inductive hypothesis, we have in $\mathsf{DILL}(\mathbb{C})$ that

$$\_; \Delta \vdash \Phi(M) : !A$$

so using one instance of !-E and an intuitionistic axiom, we have

$$\_; \Delta \vdash \mathsf{let}\ !x_1\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ x_1 : A$$

as required.

Promotion Rule: The derivation here is

$$\frac{i = 1..r \quad \Delta_i \vdash M_i : !A_i \quad x_1 : !A_1 \ldots x_r : !A_r \vdash N : B}{\Delta \vdash \mathsf{promote}\ \vec{M}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ N : B}$$

where $\Delta = \Delta_1 \# \ldots \# \Delta_r$.

By the substitution lemma and the inductive hypothesis, we have $y_1 : A_1 \ldots y_r : A_r; \_ \vdash \Phi(N)\{!\vec{y}/\vec{x}\} : B$. Hence, by the promotion rule of $\mathsf{DILL}(\mathbb{C})$, we have

$$y_1 : A_1 \ldots y_r : A_r; \_ \vdash !\Phi(N)\{!\vec{y}/\vec{x}\} : !B$$

We also have by induction $\_; \Delta_i \vdash \Phi(M_i) : !A_i$ for each $i = 1..r$. Hence, by r applications of the !-E rule, we have

$$\_; \Delta_i \vdash \mathsf{let}\ !\vec{y}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ !\Phi(N)\{!\vec{y}/\vec{x}\} : B$$

This is precisely what is given in the translations. □

We now give one auxiliary lemma:

Lemma 2.4.3
We have that for terms $\Delta_1, x : A \vdash M : B$ and $\Delta_2 \vdash N : A$ of $\mathsf{ILL}(\mathbb{C})$,

$$\_; \Delta \vdash \Phi(M\{N/x\}) = \Phi(M)\{\Phi(N)/x\} : B$$

where $\Delta = \Delta_1 \# \Delta_2$.

This is easily proved by induction over the first term, M, and we leave it to the reader.

We now prove the lemma on derivable equality judgements.

Lemma 2.4.1 (3) *If* $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M = N : A$, *then* $\_; \Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(M) = \Phi(N) : A$.

Proof This is proved by induction over the derivation of the derivable equality judgement $\Delta \vdash M = N : A$ in $\mathsf{ILL}(\mathbb{C})$. First we consider all the one-step derivations, ie axioms. However, since there are a large number of these, most of which are identical to those already given in $\mathsf{DILL}(\mathbb{C})$, we give the proof only for the substantially different ones, that is the exponential axioms. It is routine for the other equality axioms and the reflexivity, symmetry and transitivity rules. In the following, we show it for the exponential axioms based on the numbering given in the definition of the equality judgement of $\mathsf{ILL}(\mathbb{C})$. We omit contexts and types for equality judgements in this proof for brevity, as they will always be reconstructible from those present in the relevant axiomatic equality.

1): In this case, the image of the left-hand side is

$$\mathsf{let}\ !y\ \mathsf{be}\ (\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ !\Phi(N)\{!\vec{z}/\vec{x}\})\ \mathsf{in}\ y$$

This is equivalent by a commuting conversion (since w and z_i are fresh) to

$$\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ (\mathsf{let}\ !y\ \mathsf{be}\ !\Phi(N)\{!\vec{z}/\vec{x}\}\ \mathsf{in}\ y)$$

However, $(\mathsf{let}\ !y\ \mathsf{be}\ !\Phi(N)\{!\vec{z}/\vec{x}\}\ \mathsf{in}\ y) = \Phi(N)\{!\vec{z}/\vec{x}\}$. Hence the image of the left-hand side of the equality is

$$\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ (\Phi(N)\{!\vec{z}/\vec{x}\})$$

But now by commuting conversions and η-equality, this is equal to $\Phi(N)\{\Phi(\vec{M})/\vec{x}\}$, which is the image of the right-hand side of the equality.

2): In this case, the image of the left-hand side is:

$$\mathsf{let}\ !y\ \mathsf{be}\ (\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ !\Phi(N)\{!\vec{z}/\vec{x}\})\ \mathsf{in}\ \Phi(N')$$

This is equivalent by a commuting conversion (since y and z_i are fresh) to

$$\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ (\mathsf{let}\ !y\ \mathsf{be}\ !\Phi(N)\{!\vec{z}/\vec{x}\}\ \mathsf{in}\ \Phi(N'))$$

We know, however, that y does not occur in $\Phi(N')$, as it is fresh, so this is equal to $\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ \Phi(N')$, which is precisely the image of the right-hand side.

3): In this case, the image of the left-hand side is:

$$\mathsf{let}\ !y\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ !(\mathsf{let}\ !z\ \mathsf{be}\ !y\ \mathsf{in}\ z)$$

which is β-equal to

$$\mathsf{let}\ !y\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ !y$$

which is η-equal to the image of the right-hand side.

4): In this case, the image of the left-hand side is:

$$\mathsf{let}\ !z\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N)\{!z/x, y\}$$

which is easily seen to be the image of the right-hand side by analogous reasoning.

5): In this case the image of the left-hand side is

$$\mathsf{let}\ !z\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ (\mathsf{let}\ !z'\ \mathsf{be}\ x\ \mathsf{in}\ \Phi(N))\{!z/x, y\}$$

which is equal to

$$\mathsf{let}\ !z\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ (\mathsf{let}\ !z'\ \mathsf{be}\ !z\ \mathsf{in}\ \Phi(N))\{!z/y\}$$

but by one β-equality this is

$$\mathsf{let}\ !z\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N)\{!z/y\}$$

so we can now see that via commuting conversions and an η-equality this is $\Phi(N)\{\Phi(M)/y\}$, which is the image of the right-hand side.

6): In this case the left hand side of the equality has image

$$\mathsf{let}\ !z\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ (\mathsf{let}\ !z'\ \mathsf{be}\ y\ \mathsf{in}\ \Phi(N)\{!z'/x', y'\})\{!z/x, y\}$$

but this is equal to

$$\mathsf{let}\ !z\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ (\mathsf{let}\ !z'\ \mathsf{be}\ !z\ \mathsf{in}\ \Phi(N)\{!z'/x', y'\})\{!z/x\}$$

However, by a β-equality this is equal to

$$\mathsf{let}\ !z\ \mathsf{be}\ \Phi(M)\ \mathsf{in}\ \Phi(N)\{!z/x, y', x'\}$$

and by a similar process we can see that the image of the right-hand side is also equal to this term.

7): In this case, the image of the left-hand side is:

$$\mathsf{let}\ !z', !\vec{z}\ \mathsf{be}\ \Phi(M'), \Phi(\vec{M})\ \mathsf{in}\ !(\mathsf{let}\ !z''\ \mathsf{be}\ x'\ \mathsf{in}\ \Phi(N))\{!z', !\vec{z}/x', \vec{x}\}$$

which is

$$\mathsf{let}\ !z', !\vec{z}\ \mathsf{be}\ \Phi(M'), \Phi(\vec{M})\ \mathsf{in}\ !(\mathsf{let}\ !z''\ \mathsf{be}\ !z'\ \mathsf{in}\ \Phi(N))\{!\vec{z}/\vec{x}\}$$

which is β-equal to

$$\mathsf{let}\ !z', !\vec{z}\ \mathsf{be}\ \Phi(M'), \Phi(\vec{M})\ \mathsf{in}\ !(\Phi(N))\{z', !\vec{z}/z'', \vec{x}\}$$

This is

$$\mathsf{let}\ !z'\ \mathsf{be}\ \Phi(M')\ \mathsf{in}\ (\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ !\Phi(N)\{z', !\vec{z}/z'', \vec{x}\})$$

but we know that z'' does not occur in $\Phi(N)$, so this is

$$\mathsf{let}\ !z'\ \mathsf{be}\ \Phi(M')\ \mathsf{in}\ (\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ !\Phi(N)\{!\vec{z}/\vec{x}\})$$

which is the image of the right-hand side.

8): In this case, the image of the left-hand side is:

$$\mathsf{let}\ !z'\ \mathsf{be}\ (\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ !\Phi(N)\{!\vec{z}/\vec{x}\})\ \mathsf{in}\ \Phi(N')\{!z'/y_1, y_2\}$$

This is equivalent, again by a commuting conversion, to

$$\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ (\mathsf{let}\ !z'\ \mathsf{be}\ !\Phi(N)\{!\vec{z}/\vec{x}\}\ \mathsf{in}\ \Phi(N')\{!z'/y_1, y_2\})$$

Now this is β-equal to

$$\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ (!\Phi(N)\{!\Phi(N')\{!\vec{z}/\vec{x}\}/y_1, y_2\})$$

But $!\Phi(N)\{!\Phi(N')\{!\vec{z}/\vec{x}\}/y_1, y_2\}$ is β-equal to

$$!\Phi(N)\{\mathsf{let}\ !\vec{z''}\ \mathsf{be}\ !\vec{z}\ \mathsf{in}\ !\Phi(N')\{!\vec{z''}/\vec{x}\}/y_1, y_2\}$$

And this in turn is the same as

$$!\Phi(N)\{(\mathsf{let}\ !\vec{z''}\ \mathsf{be}\ \vec{x'}\ \mathsf{in}\ !\Phi(N')\{!\vec{z''}/\vec{x}\}), (\mathsf{let}\ !\vec{z''}\ \mathsf{be}\ \vec{x''}\ \mathsf{in}\ !\Phi(N')\{!\vec{z''}/\vec{x}\})/y_1, y_2\}\{!\vec{z}/\vec{x'}, \vec{x''}\}$$

which is

$$\Phi(N\{(\mathsf{promote}\ \vec{x'}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ N'), (\mathsf{promote}\ \vec{x''}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ N')/y_1, y_2\}\{!\vec{z}/\vec{x'}, \vec{x''}\})$$

Therefore, the image of the left-hand side is

$$\mathsf{let}\ !\vec{z''}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}$$

$$(\Phi(N\{(\mathsf{promote}\ \vec{x'}\ \mathsf{for}\ x\ \mathsf{in}\ N'), (\mathsf{promote}\ \vec{x''}\ \mathsf{for}\ x\ \mathsf{in}\ N')/y_1, y_2\}\{!\vec{z''}/\vec{x'}, \vec{x''}\}))$$

But this is precisely the image of the right-hand side, so we are done.

9): In this case, the image of the left-hand side is:

$$\mathsf{let}\ !z', !\vec{z}\ \mathsf{be}\ \Phi(M'), \Phi(\vec{M})\ \mathsf{in}\ !(\mathsf{let}\ !z''\ \mathsf{be}\ !z'\ \mathsf{in}\ \Phi(N)\{!z''/y_1, y_2\})\{!\vec{z}/\vec{x}\}$$

This is

$$\mathsf{let}\ !z', !\vec{z}\ \mathsf{be}\ \Phi(M'), \Phi(\vec{M})\ \mathsf{in}\ !(\mathsf{let}\ !z''\ \mathsf{be}\ !z'\ \mathsf{in}\ \Phi(N)\{!z''/y_1, y_2\}\{!\vec{z}/\vec{x}\})$$

which is β-equal to

$$\mathsf{let}\ !z', !\vec{z}\ \mathsf{be}\ \Phi(M'), \Phi(\vec{M})\ \mathsf{in}\ !(\Phi(N)\{!z'/y_1, y_2\}\{!\vec{z}/\vec{x}\})$$

and this is equal to

$$\mathsf{let}\ !z'\ \mathsf{be}\ \Phi(M')\ \mathsf{in}\ (\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{M})\ \mathsf{in}\ !(\Phi(N)\{!z'/y_1, y_2\}\{!\vec{z}/\vec{x}\}))$$

But this is η-equal to the translation of the right-hand side.

10): The image of the left-hand side of this equality is

$$\mathsf{let}\ !z', !\vec{z''}\ \mathsf{be}\ (\mathsf{let}\ !\vec{z'''}\ \mathsf{be}\ \vec{y_1}\ \mathsf{in}\ !\Phi(M)[!\vec{z'''}/\vec{x}_1]), \vec{y_2}\ \mathsf{in}\ !\Phi(N)\{!z', !\vec{z''}/x', \vec{x}_2\}$$

By commuting conversions this is equal to

$$\mathsf{let}\ !\vec{z'''}\ \mathsf{be}\ \vec{y_1}\ \mathsf{in}\ (\mathsf{let}\ !z', !\vec{z''}\ \mathsf{be}\ !\Phi(M)\{!\vec{z'''}/\vec{x}_1\}, \vec{w}\ \mathsf{in}\ !\Phi(N)\{!z'1, !\vec{z''}/x', \vec{x}_2\})$$

This then is η-equal to

$$\mathsf{let}\ !\vec{z'''}\ \mathsf{be}\ \vec{y_1}\ \mathsf{in}\ (\mathsf{let}\ !\vec{z''}\ \mathsf{be}\ \vec{y_2}\ \mathsf{in}\ !\Phi(N)\{!\Phi(M)\{!\vec{z'''}/\vec{x}_1\}, !\vec{z''}/x', \vec{x}_2\})$$

This is abbreviated to

$$\mathsf{let}\ !\vec{z'''}, !\vec{z''}\ \mathsf{be}\ \vec{y_1},, \vec{y_2}\ \mathsf{in}\ (!\Phi(N)\{!\Phi(M)\{!\vec{z'''}/\vec{x}_1\}, !\vec{z''}/x', \vec{x}_2\})$$

But by an η-equality this is equal to

$$\mathsf{let}\ !\vec{z'''}, !\vec{z''}\ \mathsf{be}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ (!\Phi(N)\{(\mathsf{let}\ !\vec{z''''}\ \mathsf{be}\ !\vec{z'''}\ \mathsf{in}\ !\Phi(M)\{!\vec{z''''}/\vec{x}_1\}), !\vec{z''}/x', \vec{x}_2\})$$

and this can be written as

$$\mathsf{let}\ !\vec{z'''}, !\vec{z''}\ \mathsf{be}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ (!\Phi(N)\{(\mathsf{let}\ !\vec{z''''}\ \mathsf{be}\ !\vec{z}\ \mathsf{in}\ !\Phi(M)\{!\vec{z''''}/\vec{x}_1\})/x'\})\{!\vec{z'''}, !\vec{z''}/\vec{z}, \vec{x}_2\}$$

But now this is the image of the right-hand side.

$\square$

From **DILL**($\mathbb{C}$) to **ILL**($\mathbb{C}$)

For each term of $\mathsf{DILL}(\mathbb{C})$ $x_1 : A_1 \ldots x_r : A_r; \Delta \vdash t : A$, we give a term $x_1 : !A_1 \ldots x_r :!A_r, \Delta \vdash \Psi_{\vec{x}}(t){:}A$ in $\mathsf{ILL}(\mathbb{C})$.

DEFINITION 2.4.4 (THE TRANSLATION Ψ)
On Types define Ψ as the identity.
On Terms define Ψ as follows:

$$
\begin{aligned}
\Psi_{\vec{y}}(x{:}A) &= \begin{cases} \mathsf{discard}\ \vec{y_1}\vec{y_2}\ \mathsf{in}\ \mathsf{derelict}(x) & \text{if } \vec{y} = \vec{y_1}x\vec{y_2} \\ \mathsf{discard}\ \vec{y}\ \mathsf{in}\ x & \text{otherwise} \end{cases} \\
\Psi_{\vec{y}}(c) &= \mathsf{discard}\ \vec{y}\ \mathsf{in}\ c \\
\Psi_{\vec{y}}(*) &= \mathsf{discard}\ \vec{y}\ \mathsf{in}\ * \\
\Psi_{\vec{y}}(\mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ u) &= \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ (\mathsf{let}\ *\ \mathsf{be}\ \Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\})\ \mathsf{in}\ \Psi_{\vec{y_2}}(u\{\vec{y_2}/\vec{y}\})) \\
\Psi_{\vec{y}}(t \otimes u) &= \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\}) \otimes \Psi_{\vec{y_2}}(u\{\vec{y_2}/\vec{y}\}) \\
\Psi_{\vec{y}}(\mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ t\ \mathsf{in}\ u) &= \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ p_1 \\
&\qquad \text{where } p_1 = \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ \Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\})\ \mathsf{in}\ \Psi_{\vec{y_2}}(u\{\vec{y_2}/\vec{y}\}) \\
\Psi_{\vec{y}}(\lambda x.t) &= \lambda x.\Psi_{\vec{y}}(t) \\
\Psi_{\vec{y}}(tu) &= \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\})\Psi_{\vec{y_2}}(u\{\vec{y_2}/\vec{y}\}) \\
\Psi_{\vec{y}}(!t) &= \mathsf{promote}\ \vec{y}\ \mathsf{for}\ \vec{y_1}\ \mathsf{in}\ \Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\}) \\
\Psi_{\vec{y}}(\mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ u) &= \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_2}x}(u\{\vec{y_2}/\vec{y}\})\{\Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\})/x\}
\end{aligned}
$$

where $\vec{y_1}$ and $\vec{y_2}$ are vectors of fresh variables.

We first make an abbreviation; given a sequence of types $\vec{A} = A_1 \ldots A_r$, we will write $!\vec{A}$ for $!A_1 \ldots !A_r$. Similarly, given a sequence of typings $\Gamma = x_1{:}A_1 \ldots x_r{:} A_r$, we will write $!\Gamma$ for $x_1!A_1 \ldots x_r!A_r$.

LEMMA 2.4.1 (2) *If* $\Gamma; \Delta \vdash_{\mathit{DILL}(\mathbb{C})} t{:}A$, *then* $!\Gamma, \Delta \vdash_{\mathit{ILL}(\mathbb{C})} \Psi_{\vec{y}}(t){:}A$.

Proof This is proved by induction over the first derivation. We leave most of this proof, as it is routine, but we consider the tensor introduction as a sample case, and also the rules for ! as they are significantly different.

Tensor Introduction We have in this case that there is a deduction in $\mathsf{DILL}(\mathbb{C})$

$$\frac{\Gamma; \Delta_1 \vdash u{:}A \quad \Gamma; \Delta_2 \vdash v{:}B}{\Gamma; \Delta \vdash u \otimes v{:}A \otimes B}$$

where $\Delta = \Delta_1 \# \Delta_2$. By the inductive hypothesis we have that there exist derivations in $\mathsf{ILL}(\mathbb{C})$ (using some α-conversion, and assuming that $\Gamma = \vec{y}\colon \vec{C}$):

$$\vec{y_1} :!\vec{C}, \Delta_1 \vdash \Psi_{\vec{y_1}}(u\{\vec{y_1}/\vec{y}\}) : A \otimes B$$

and

$$\vec{y_2} :!\vec{C}, \Delta_2 \vdash \Psi_{\vec{y_2}}(v\{\vec{y_2}/\vec{y}\}) : A \otimes B$$

Now we have by the tensor introduction in $\mathsf{ILL}(\mathbb{C})$

$$\vec{y_1} :!\vec{C}, \vec{y_2} :!\vec{C}, \Delta \vdash \Psi_{\vec{y_1}}(u\{\vec{y_1}/\vec{y}\}) \otimes \Psi_{\vec{y_2}}(v\{\vec{y_2}/\vec{y}\}) : A \otimes B$$

But by a sequence of copies, we can now obtain:

$$!\Gamma, \Delta \vdash \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_1}}(u\{\vec{y_1}/\vec{y}\}) \otimes \Psi_{\vec{y_2}}(v\{\vec{y_2}/\vec{y}\}) : A \otimes B$$

which is precisely the image of the tensor. In fact, the technique of modelling the shared intuitionistic context with repeated contractions accounts for all of the copy constructs in the definition of Ψ.

!-Introduction In this case, we have the following deduction in $\mathsf{DILL}(\mathbb{C})$:

$$\frac{\Gamma; \_ \vdash t : A}{\Gamma; \_ \vdash !t :!A}$$

By the inductive hypothesis, we have a derivation in $\mathsf{ILL}(\mathbb{C})$(using some α-conversion)

$$\vec{y'} :!\vec{B} \vdash \Psi_{\vec{y'}}(t\{\vec{y'}/\vec{y}\}) : A$$

where $\Gamma = \vec{y}\colon \vec{B}$. Now by one use of promotion, we have

$$!\Gamma \vdash \mathsf{promote}\ \vec{y}\ \mathsf{for}\ \vec{y'}\ \mathsf{in}\ \Psi_{\vec{y'}}(t\{\vec{y'}/\vec{y}\}) :!A$$

which is precisely the image of $!t$.

!-Elimination In this case, we have the following derivation in $\mathsf{DILL}(\mathbb{C})$.

$$\frac{\Gamma; \Delta_1 \vdash t :!A \quad \Gamma, x\colon A; \Delta_2 \vdash u : B}{\Gamma; \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ u : B}$$

where $\Delta = \Delta_1 \# \Delta_2$. Hence again by the inductive hypothesis we have the following derivations in $\mathsf{ILL}(\mathbb{C})$ (using some α-conversion:

$$\vec{y_1} :!\vec{C}, \Delta_1 \vdash \Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\}) :!A$$

where $\Gamma = \vec{y}\colon \vec{C}$, and

$$\vec{y_2} :!\vec{C}, x :!A, \Delta \vdash \Psi_{\vec{y_2}}(u\{\vec{y_2}/\vec{y}\}) : B$$

Now by the admissible cut rule in $\mathsf{ILL}(\mathbb{C})$ we have

$$\vec{y_1} :!\vec{C}, \vec{y_2} :!\vec{C}, \Delta \vdash \Psi\vec{y_2}(u\{\vec{y_2}/\vec{y}\})\{\Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\})/x\}:B$$

Now by the familiar series of contractions, we have

$$!\Gamma, \Delta \vdash \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_2}}(u\{\vec{y_2}/\vec{y}\})\{\Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\})/x\}:B$$

□

We now give auxiliary lemmas relating intuitionistic and linear substitutions in $\mathsf{DILL}(\mathbb{C})$ to substitution in $\mathsf{ILL}(\mathbb{C})$.

LEMMA 2.4.5 (LINEAR SUBSTITUTION)
If we consider the substitution:

$$\frac{\Gamma; \Delta_1, x:A \vdash_{\mathsf{DILL}(\mathbb{C})} t:B \quad \Gamma; \Delta_2 \vdash_{\mathsf{DILL}(\mathbb{C})} u:A}{\Gamma; \Delta_1, \Delta_2 \vdash_{\mathsf{DILL}(\mathbb{C})} t[u/x]:B}$$

where $\Gamma = \vec{y}:\vec{C}$, then

$$\Psi_{\vec{y}}(t[u/x]) = \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{x}\})\{\Psi(u)_{\vec{y_2}}\{\vec{y_2}/\vec{x}\}/x\}$$

LEMMA 2.4.6 (INTUITIONISTIC SUBSTITUTION)
If we consider the substitution:

$$\frac{\Gamma, x:A; \Delta \vdash_{\mathsf{DILL}(\mathbb{C})} t:B \quad \Gamma; \_ \vdash_{\mathsf{DILL}(\mathbb{C})} u:A}{\Gamma; \Delta \vdash_{\mathsf{DILL}(\mathbb{C})} t[u/x]:B}$$

where $\Gamma = \vec{y}:\vec{C}$, then we have that

$$\Psi_{\vec{y}}(t[u/x]) = \mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_1},x}(t\{\vec{y_1}/\vec{y}\})\{\mathsf{promote}\ \vec{y_2}\ \mathsf{for}\ \vec{y_3}\ \mathsf{in}\ \Psi_{\vec{y_3}}(u\{\vec{y_3}/\vec{y}\})/x\}$$

These are both routine inductions over the structure of the first term, and are left to the reader.

Now we can prove the equality lemma:

LEMMA 2.4.1.4 *If* $\Gamma; \Delta \vdash_{\mathit{DILL}(\mathbb{C})} t = u:A$, *then* $!\Gamma, \Delta \vdash_{\mathit{ILL}(\mathbb{C})} \Psi_{\vec{y}}(t) = \Psi_{\vec{y}}(u):A$ *where* $\Gamma = \vec{y}:\vec{A}$.

Proof This is proved again by induction over the length of the derivation of equality in $\mathsf{DILL}(\mathbb{C})$. We consider only the exponential equalities, as it is easy but time-consuming to show that the other components of the equality rule system over $\mathsf{DILL}(\mathbb{C})$ correspond to equalities on $\mathsf{ILL}(\mathbb{C})$.

!-β This equality is:

$$\Gamma; \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ !u\ \mathsf{in}\ v = v\{u/x\} : A$$

The image of the left-hand side of this is:

$$\mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_1}}(v[\vec{y_1}/\vec{y}])[\mathsf{promote}\ \vec{y_2}\ \mathsf{for}\ \vec{y_3}\ \mathsf{in}\ \Psi_{\vec{y_3}}(u[\vec{y_3}/\vec{y}]/x])$$

assuming that $\Gamma = \vec{y}{:}\vec{C}$. But this is just the image of intuitionistic substitution in $\mathsf{ILL}(\mathbb{C})$.

!-η This equality is

$$\Gamma; \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ !x = t !A$$

The image of the left-hand side is

$$\mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ (\mathsf{discard}\ \vec{y_2}\ \mathsf{in}\ \mathsf{promote}\ x\ \mathsf{for}\ x'\ \mathsf{in}\ \mathsf{derelict}\ x'\ )\{\Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\})/x\}$$

assuming that $\Gamma = \vec{y}{:}\vec{B}$. By equality (5) of $\mathsf{ILL}(\mathbb{C})$ we have that this is precisely

$$\mathsf{copy}\ \vec{y}\ \mathsf{for}\ \vec{y_1}, \vec{y_2}\ \mathsf{in}\ (\mathsf{discard}\ \vec{y_2}\ \mathsf{in}\ \Psi_{\vec{y_1}}(t\{\vec{y_1}/\vec{y}\}))$$

However, using equality (7) this is just $\Psi_{\vec{y}}(t)$

Commuting Conversions are dealt with easily, as they translate to the commuting conversions in $\mathsf{ILL}(\mathbb{C})$. $\square$

LEMMA 2.4.1.5 *For any term $\Delta \vdash_{ILL(\mathbb{C})} M : A$ of* ILL*, $\Delta \vdash_{ILL(\mathbb{C})} \Psi_{\varepsilon}(\Phi(M)) =_A M$.*

Proof We note first that the translation Φ is effectively the identity on terms other than those containing the exponential constructors. Moreover, since Φ translates sequents to sequents derivable from no intuitionistic assumptions, applying Ψ to these sequents gives the identity (as we need no copy or $\mathsf{discard}$ constructs). Hence we know that $\Psi(\Phi(t))$ is the identity except perhaps on terms involving the exponential constructors.

We prove that the translation satisfies the property above by consideration of the structure of t. We consider only the exponential cases.

derelict: In this case, we have that M has the form $\mathsf{derelict}(N)$, and hence that $\Phi(M)$ has the form $\mathsf{let}\ !z\ \mathsf{be}\ \Phi(N)\ \mathsf{in}\ z$. This must have the following derivation in $\mathsf{DILL}(\mathbb{C})$:

$$\frac{z{:}A;\ \_ \vdash z{:}A \qquad \_;\Delta \vdash \Phi(N) :!A}{\_;\Delta \vdash \mathsf{let}\ !z\ \mathsf{be}\ \Phi(N)\ \mathsf{in}\ z{:}A}$$

The translation Ψ takes this derivation to

$$\frac{z :!A \vdash \mathsf{derelict}\ (z){:}A \qquad \Delta \vdash \Psi_{\langle\rangle}(\Phi(N)) :!A}{\Psi(\Delta) \vdash \mathsf{derelict}\ (z)\{\Psi_\varepsilon(\Phi(N))/z\}{:}A}$$

which is $\mathsf{derelict}(\Psi_\varepsilon(\Phi(u)))$, but this is $\beta\eta$-equal to $\mathsf{derelict}\ (u)$ by the inductive hypothesis.

discard: In this case, M has the form $\mathsf{discard}\ N\ \mathsf{in}\ N'$, so that $\Phi(M)$ is $\mathsf{let}\ !z\ \mathsf{be}\ \Phi(N)\ \mathsf{in}\ \Phi(N')$. This must have the following derivation in $\mathsf{DILL}(\mathbb{C})$:

$$\frac{\_;\Delta_1 \vdash \Phi(N) :!A \qquad z{:}A;\Delta_2 \vdash \Phi(N'){:}B}{\_;\Delta \vdash \mathsf{let}\ !z\ \mathsf{be}\ \Phi(N)\ \mathsf{in}\ \Phi(N'){:}B}$$

where $\Delta = \Delta_1 \# \Delta_2$. Ψ applied to this derivation gives the following:

$$\frac{z :!A, \Delta_1 \vdash \mathsf{discard}\ z\ \mathsf{in}\ \Psi_\varepsilon(\Phi(N')){:}B \qquad \Delta_2 \vdash \Psi_\varepsilon(\Phi(N)) :!A}{\Delta \vdash \mathsf{discard}\ z\ \mathsf{in}\ (\Psi_\varepsilon(\Phi(N')))\{\Psi_\varepsilon(\Phi(N))/z\}{:}B}$$

but since z does not occur in $\Phi(N')$ and hence in $\Psi_\varepsilon(\Phi(N'))$, this is precisely $\mathsf{discard}\ \Psi_\varepsilon(\Phi(N))\ \mathsf{in}\ (\Psi_\varepsilon(\Phi(N')))$ which is $\beta\eta$-equal to $\mathsf{discard}\ N\ \mathsf{in}\ N'$ by hypothesis.

copy: Here, $\Phi(M)$ has the form $\mathsf{let}\ !z\ \mathsf{be}\ \Phi(N)\ \mathsf{in}\ \Phi(N')\{!z/x, y\}$, and therefore has the derivation:

$$\frac{z{:}A;\Delta_1 \vdash \Phi(N')\{!z/x,y\}{:}B \qquad \_;\Delta_2 \vdash \Phi(N) :!A}{\_;\Delta \vdash \mathsf{let}\ !z\ \mathsf{be}\ \Phi(N)\ \mathsf{in}\ \Phi(N')\{!z/x,y\}{:}B}$$

Under Ψ, this derivation becomes

$$\frac{z :!A, \Delta_1 \vdash \mathsf{copy}\ z\ \mathsf{for}\ x,y\ \mathsf{in}\ \Psi_\varepsilon(\Phi(N')){:}B \qquad \Delta_2 \vdash \Psi_\varepsilon(\Phi(N)) :!A}{\Delta \vdash \mathsf{copy}\ z\ \mathsf{for}\ x,y\ \mathsf{in}\ \Psi_\varepsilon(\Phi(N'))\{\Psi_\varepsilon(\Phi(N))/z\}{:}B}$$

which is $\mathsf{copy}\ \Psi_\varepsilon(\Phi(N))\ \mathsf{for}\ x,y\ \mathsf{in}\ \Psi_\varepsilon(\Phi(N'))$, which is by hypothesis equal to $\mathsf{copy}\ N\ \mathsf{for}\ x,y\ \mathsf{in}\ N'$, or M.

promote: In this case, M has the form $\mathsf{promote}\ \vec{N}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ N'$. Hence $\Phi(M)$ is $\mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{N})\ \mathsf{in}\ !\Phi(N')\{!\vec{z}/\vec{x}\}$.

This has the derivation

$$\frac{\vec{z}{:}\vec{A};\ \_ \vdash !\Phi(N')[!\vec{z}/\vec{x}]{:}B \qquad \_;\Delta_i \vdash \Phi(N_i) :!A_i \qquad (i = 1 \ldots r)}{\_;\Delta \vdash \mathsf{let}\ !\vec{z}\ \mathsf{be}\ \Phi(\vec{N})\ \mathsf{in}\ !\Phi(N')\{!\vec{z}/\vec{x}\}{:}B}$$

where $\Delta = \Delta_1\# \ldots \#\Delta_r$. When Ψ is applied, this becomes

$$\frac{\vec{z} : !\vec{A} \vdash \mathsf{promote}\ \vec{z}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ \Psi_\varepsilon(\Phi(N')){:}B \qquad \Delta_i \vdash \Psi_\varepsilon(\Phi(N_i)) :!A_i \qquad (i = 1 \ldots r)}{\Delta \vdash (\mathsf{promote}\ \vec{z}\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ \Psi_\varepsilon(\Phi(N')))\{\Psi_\varepsilon(\Phi(N_1)) \ldots \Psi_\varepsilon(\Phi(N_r))/z_1 \ldots z_r\}{:}B}$$

but this final term is just $\mathsf{promote}\ \Psi_\varepsilon(\vec{\Phi}(N))\ \mathsf{for}\ \vec{x}\ \mathsf{in}\ \Psi_\varepsilon(\Phi(N'))$, which is equal to the original term by induction.

We can prove an analogous lemma for the alternative direction:

LEMMA 2.4.1)(6) *For any term* $\_;\Delta \vdash t{:}A$ *of* $\mathit{DILL}(\mathbb{C})$, $\_;\Delta \vdash \Phi(\Psi_\varepsilon(t)) = t{:}A$.

This is easily proved in the same manner as the previous lemma.

Now by virtue of the proof at the beginning of this section, we have the results:

THEOREM 1.1 $\Delta \vdash_{\mathit{ILL}(\mathbb{C})} M = N{:}A$ *iff* $\_;\Delta \vdash_{\mathit{DILL}(\mathbb{C})} \Phi(M) = \Phi(N){:}A$

THEOREM 1.2 $\_;\Delta \vdash_{\mathit{DILL}(\mathbb{C})} t = u{:}A$ *iff* $\Delta \vdash_{\mathit{ILL}(\mathbb{C})} \Psi_\varepsilon(t) = \Psi_\varepsilon(u){:}A$

Chapter 3

The Semantics of DILL

We now show that $\mathsf{DILL}(\mathbb{C})$ can be soundly and completely mapped into a class of models for linear logic. By this we mean that we can map terms $\Gamma; \Delta \vdash t : A$ to morphisms in the model in such a way that the $\Gamma; \Delta$ terms t and u of type A are judged to be equal $\Gamma; \Delta \vdash t = u : A$ if and only if the interpretations of the derivations are equal in every model of the class.

The models we will consider are based on linear-non-linear models, which were introduced by Benton [Ben95a] and studied by Bierman [Bie95]. These are pairs of categories $(\mathcal{C}, \mathcal{S})$ such that $\mathcal{C}$ is a CCC and $\mathcal{S}$ is a SMCC and there is a monoidal adjunction between them. The intention behind the construction is that the normal power of intuitionistic logic, arising from the exponential, should be modelled in the CCC, with the intuitionistic linear logic being as usual modelled in the SMCC. This idea emerged in 1993 from discussions between a number of people, including Plotkin, Benton and Hyland. However, it was only during further work by Benton [Ben95a] that the details became clear, and in particular that it was necessary to impose the requirement that the adjunction be monoidal. In [Ben94] there is an extensive comparison between these models and the previously proposed models [BBdPH93b]. In fact, although Benton required that the cartesian category be closed, this is not essential for our purposes, and we take the more general definition.

3.1 The Interpretation

We will make extensive use of the primitive categorical definitions in appendix A.2, and in particular will refer to equations there using their numbers with no further comment.

The *carrier* of a $\mathsf{DILL}(\mathbb{C})$-model is a quadruple $(\mathcal{C}, \mathcal{S}, F, G)$, such that $\mathcal{C}$ is a CC, $\mathcal{S}$ is a SMCC and $F \dashv G : \mathcal{C} \to \mathcal{S}$ is a strong monoidal functor. We will write un for the unit of the monoidal adjunction, and nu for its counit.

DEFINITION 3.1.1 (DILL($\mathbb{C}$)-MODEL)
A $\mathsf{DILL}(\mathbb{C})$-model, which we will write $\mathcal{L}$, is a carrier $(\mathcal{C}, \mathcal{S}, F, G)$ together with a primitive interpretation function $[\![\_]\!]^{\mathcal{L}}_{\mathsf{P}_L} : \mathsf{P}_L \to \mathsf{obj}(\mathcal{S})$ and a morphism $[\![c]\!]^{\mathcal{L}}_{\mathsf{C}} \in \mathcal{S}(I, [\![A]\!])$ for each constant $c{:}A$ in C, where the interpretation function is extended to arbitrary types in the obvious way, as defined shortly.

Where the particular model we are referring to is clear, we may omit the superscript $\mathcal{L}$ on the interpretation $[\![\_]\!]^{\mathcal{L}}$.

We now make some definitions to simplify the interpretation. For a sequence of objects in the category $\mathcal{S}$, $\vec{X} = X_1 \ldots X_r$, we define $\bigotimes \vec{X}$ to be the left-bracketed tensor of this sequence, $\otimes\vec{X} = (..(X_1 \otimes X_2) \otimes \ldots X_r)$ Also, for a sequence of typings $\vec{x}{:}\vec{A}$, we define $!(\vec{x}{:}\vec{A}) = \vec{x} :!\vec{A}$.

Given a $\mathsf{DILL}(\mathbb{C})$-model $\mathcal{L}$ over signature $(\mathsf{P}_L, \mathsf{C})$, we now define an interpretation $[\![\_]\!]^{\mathcal{L}}$ which takes the types and typing contexts of $\mathsf{DILL}(\mathbb{C})$ to objects in $\mathcal{S}$, and takes the terms of $\mathsf{DILL}(\mathbb{C})$ to morphisms in $\mathcal{S}$.

DEFINITION 3.1.2 (THE INTERPRETATION ON TYPES AND CONTEXTS)
On Types We define $[\![\_]\!]^{\mathcal{L}}$ on types as follows:

$$\begin{array}{rcl} [\![l]\!] & = & [\![l]\!]_{\mathsf{P}_L} \text{ for } l \in \mathsf{P}_L \\ [\![I]\!] & = & I \\ [\![A \otimes B]\!] & = & [\![A]\!] \otimes [\![B]\!] \\ [\![A \multimap B]\!] & = & [\![A]\!] \multimap [\![B]\!] \\ [\![!A]\!] & = & FG([\![A]\!]) \end{array}$$

On Typing Contexts We extend this firstly to sequences of types by saying that $[\![A_1 \ldots A_r]\!] = \bigotimes([\![A_1]\!] \ldots [\![A_r]\!])$. We then say that for a sequence of typings Δ, $[\![\Delta]\!] = [\![|\Delta|]\!]$. We then extend the definition to typing contexts by saying that for a typing context $\Gamma; \Delta$, $[\![\Gamma; \Delta]\!] = [\![!|\Gamma|, |\Delta|]\!]$.

We can now define some context-manipulation arrows, using the structure we have in the model.

Define

$$\begin{array}{rl} \mathsf{perm}_{A,B,A',B'} & :(A \otimes B) \otimes (A' \otimes B') \to (A \otimes A') \otimes (B \otimes B') \\ \mathsf{mge}_{\Delta,\Delta_1,\Delta_2} & :[\![\Delta]\!] \to [\![\Delta_1, \Delta_2]\!] \text{ where } \Delta = \Delta_1 \# \Delta_2 \\ \mathsf{sep}_{\Delta_1,\Delta_2} & :[\![\Delta_1, \Delta_2]\!] \to [\![\Delta_1]\!] \otimes [\![\Delta_2]\!] \\ \mathsf{dup}_{\Gamma} & :[\![!|\Gamma|]\!] \to [\![!|\Gamma|]\!] \otimes [\![!|\Gamma|]\!] \\ \mathsf{split}_{\Gamma,\Delta,\Delta_1,\Delta_2} & :[\![\Gamma; \Delta]\!] \to [\![\Gamma; \Delta_1]\!] \otimes [\![\Gamma; \Delta_2]\!] \text{ where } \Delta = \Delta_1 \# \Delta_2 \\ \mathsf{disc}_{\Gamma} & :[\![!|\Gamma|]\!] \to I \\ \mathsf{prom}_{\Gamma} & :[\![\Gamma; \_]\!] \to ![\![\Gamma; \_]\!] \end{array}$$

as follows:

$$\mathsf{perm}_{A,B,A',B'} = \mathsf{a}_{A,B,A'\otimes B'}; (\mathsf{id}_A \otimes \mathsf{a}^{-1}_{B,A',B'}); (\mathsf{id}_A \otimes (\mathsf{s}_{B,A'} \otimes \mathsf{id}_{[\![B']\!]})); \\ (\mathsf{id}_A \otimes \mathsf{a}_{A',B,B'}); \mathsf{a}^{-1}_{A,A',B\otimes B')}$$

$$\mathsf{mge}_{\Delta,\Delta_1,\Delta_2} = \begin{cases} \mathsf{sep}_{xA,\Delta'}; & (\mathsf{id}_{[\![A]\!]} \otimes \mathsf{mge}_{\Delta',\Delta'_1,\Delta_2}); \mathsf{sep}^{-1}_{xA,(\Delta'_1,\Delta_2)} \\ & \text{if } \Delta = x{:}A, \Delta' \text{ and } \Delta_1 = x{:}A, \Delta'_1 \\ \mathsf{sep}_{xA,\Delta'}; & (\mathsf{id}_{[\![A]\!]} \otimes \mathsf{mge}_{\Delta',\Delta_1,\Delta'_2}); (\mathsf{id}_{[\![A]\!]} \otimes \mathsf{sep}_{\Delta_1,\Delta'_2}); \\ & \mathsf{a}_{[\![A]\!],[\![\Delta_1]\!],[\![\Delta'_2]\!]}; (\mathsf{s}_{[\![A]\!],[\![\Delta_1]\!]} \otimes \mathsf{id}_{\Delta'_2})(\mathsf{sep}^{-1}_{\Delta_1,A} \otimes \mathsf{id}_{\Delta'_2}); \mathsf{sep}^{-1}_{(\Delta_1,A),\Delta'_2} \\ & \text{if } \Delta = x{:}A, \Delta' \text{ and } \Delta_2 = x{:}A, \Delta'_2 \\ \mathsf{id}_I & \text{if } \Delta = \Delta_1 = \Delta_2 = \_ \end{cases}$$

$$\mathsf{sep}_{\Delta_1,\Delta_2} = \begin{cases} (\mathsf{sep}_{\Delta_1,\Delta'_2} \otimes \mathsf{id}_{[\![A]\!]}); \mathsf{a}_{[\![\Delta_1]\!],[\![\Delta_2]\!],[\![A]\!]}; & \\ \mathsf{id}_{[\![\Delta_1]\!]} \otimes \mathsf{sep}^{-1}_{\Delta'_2,A}) & \text{where } \Delta_1 \neq \_ \text{ and } \Delta_2 = x{:}A, \Delta'_2 \\ \mathsf{ri}^{-1}_{[\![\Delta_2]\!]} & \text{if } \Delta_1 = \_ \\ \mathsf{li}^{-1}_{[\![\Delta_1]\!]} & \text{if } \Delta_2 = \_ \end{cases}$$

$$\mathsf{dup}_\Gamma = \begin{cases} \mathsf{dup}_{\Gamma',x:A} \otimes (F(\mathsf{c}_{G[\![B]\!]}); \mathsf{m}^{-1}_{G[\![B]\!],G[\![B]\!]}); & \\ \qquad \mathsf{perm}_{!(\Gamma',x:A),!(\Gamma',x:A),y:!B,y:!B} & \text{if } \Gamma = \Gamma', x : A, y : B \\ F(\mathsf{c}_{G[\![A]\!]}); \mathsf{m}^{-1}_{G[\![A]\!],G[\![A]\!]} & \text{if } \Gamma = x{:}A \\ \mathsf{ri}_I & \text{if } \Gamma = \_ \end{cases}$$

$$\mathsf{split}_{\Gamma,\Delta,\Delta_1,\Delta_2} = \mathsf{sep}_{!\Gamma,\Delta}; (\mathsf{dup}_\Gamma \otimes \mathsf{mge}_{\Delta,\Delta_1,\Delta_2}); (\mathsf{id}_{[\![\Gamma]\!]\otimes[\![\Gamma]\!]} \otimes \mathsf{sep}_{\Delta_1,\Delta_2}) \\ ; \mathsf{perm}_{[\![!\Gamma]\!],[\![!\Gamma]\!],[\![\Delta_1]\!],[\![\Delta_2]\!]}; (\mathsf{sep}^{-1}_{!\Gamma,\Delta_1} \otimes \mathsf{sep}^{-1}_{!\Gamma,\Delta_2})$$

$$\mathsf{disc}_\Gamma = \begin{cases} \mathsf{disc}_{\Gamma',y:B} \otimes (F(\mathsf{d}_{G[\![A]\!]}); \mathsf{mi}^{-1}); \mathsf{ri}^{-1}_I & \text{if } \Gamma = \Gamma', y{:}B, x{:}A \\ F(\mathsf{d}_{G[\![A]\!]}); \mathsf{mi}^{-1} & \text{if } \Gamma = x{:}A \\ \mathsf{id}_I & \text{if } \Gamma = \_ \end{cases}$$

$$\mathsf{prom}_\Gamma = \begin{cases} \mathsf{prom}_{\Gamma',y:B} \otimes F(\mathsf{un}_{G[\![A]\!]}) & \text{if } \Gamma = \Gamma', y{:}B, x{:}A \\ F(\mathsf{un}_{G[\![A]\!]}) & \text{if } \Gamma = x{:}A \\ \mathsf{mi} & \text{if } \Gamma = \_ \end{cases}$$

where $\mathsf{m}_{X,Y}$ and mi are the monoidality natural transformations for the functor FG.

The interpretation will take a term $\Gamma; \Delta \vdash t{:}A$ to an arrow

$$[\![\Gamma; \Delta \vdash t{:}A]\!] : [\![\Gamma; \Delta]\!] \to [\![A]\!]$$

in the SMCC part of $\mathcal{L}$.

Definition 3.1.3 (The Interpretation on Sequents)
First we recall that given a $\Gamma; \Delta$ term t of type A, there is a unique derivation of the typing judgement $\Gamma; \Delta \vdash t : A$. We now define $[\![\_]\!]^{\mathcal{L}}$ inductively over the structure of this unique derivation as follows:

Linear Axiom Here we have:

$$\begin{aligned}&[\![\Gamma; x\!:\!A \vdash x\!:\!A]\!] : [\![\Gamma; A]\!] \to [\![A]\!]\\ &= \mathsf{sep}_{,!\Gamma,A}; (\mathsf{disc}_\Gamma \otimes \mathsf{id}_{[\![A]\!]}); \mathsf{ri}_{[\![A]\!]}\end{aligned}$$

Intuitionistic Axiom Here, we have

$$\begin{aligned}&[\![\Gamma, x\!:\!A, \Gamma'; \_ \vdash x\!:\!A]\!] : [\![\Gamma, A, \Gamma'; \_]\!] \to [\![A]\!]\\ &= \mathsf{sep}_{(!\Gamma,x!A),!\Gamma'}; (\mathsf{sep}_{(!\Gamma),!xA} \otimes \mathsf{id}_{[\![!\Gamma']\!]});((\mathsf{disc}_\Gamma \otimes \mathsf{der}_A) \otimes \mathsf{disc}_{\Gamma'}); \mathsf{li}_{I\otimes[\![A]\!]}; \mathsf{ri}_{[\![A]\!]}\end{aligned}$$

Constant Axiom Here. we have

$$\begin{aligned}&[\![\Gamma; \_ \vdash c\!:\!A]\!] : [\![\Gamma; \_]\!] \to [\![A]\!]\\ &= \mathsf{disc}_\Gamma; [\![c]\!]_{\mathsf{C}}\end{aligned}$$

I-introduction Here we have:

$$\begin{aligned}&[\![\Gamma; \_ \vdash *\!:\!I]\!] : [\![\Gamma; \_]\!] \to [\![I]\!]\\ &= \mathsf{disc}_\Gamma\end{aligned}$$

I-elimination We need to interpret the derivation

$$\frac{\Gamma; \Delta_1 \vdash t\!:\!I \quad \Gamma; \Delta_2 \vdash u\!:\!A}{\Gamma; \Delta \vdash \mathsf{let}\ * \ \mathsf{be}\ t\ \mathsf{in}\ u\!:\!A}$$

where $\Delta = \Delta_1 \# \Delta_2$. In this case, we have arrows

$$\begin{aligned}f &: [\![\Gamma; \Delta_1]\!] \to [\![I]\!]\\ g &: [\![\Gamma; \Delta_2]\!] \to [\![A]\!]\end{aligned}$$

Hence we have

$$\begin{aligned}&[\![\Gamma; \Delta \vdash \mathsf{let}\ * \ \mathsf{be}\ t\ \mathsf{in}\ u\!:\!A]\!] : [\![\Gamma; \Delta]\!] \to [\![A]\!]\\ &= \mathsf{split}_{\Gamma,\Delta,\Delta_1,\Delta_2}; (f \otimes g); \mathsf{ri}_{[\![A]\!]}\end{aligned}$$

$\otimes$-introduction The rule is:

$$\frac{\Gamma; \Delta_1 \vdash t\!:\!A \quad \Gamma; \Delta_2 \vdash u\!:\!B}{\Gamma; \Delta \vdash t \otimes u\!:\!A \otimes B}$$

where $\Delta = \Delta_1 \# \Delta_2$. Using the premises, we already have arrows

$$\begin{aligned}f &: [\![\Gamma; \Delta_1]\!] \to [\![A]\!]\\ g &: [\![\Gamma; \Delta_2]\!] \to [\![B]\!]\end{aligned}$$

Hence we have

$$\begin{aligned}[\![\Gamma;\Delta \vdash t \otimes u{:}A \otimes B]\!] : [\![\Gamma;\Delta]\!] &\to [\![A \otimes B]\!] \\ &= \mathsf{str}_{\Gamma,\Delta,\Delta_1,\Delta_2};(f \otimes g)\end{aligned}$$

⊗-Elimination The rule is

$$\frac{\Gamma;\Delta_1 \vdash u{:}(A \otimes B) \quad \Gamma;\Delta_2, x{:}A, y{:}B \vdash t{:}C}{\Gamma;\Delta \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ u\ \mathsf{in}\ t{:}C}$$

where $\Delta = \Delta_1 \# \Delta_2$. We have arrows:

$$f : [\![\Gamma;\Delta_1]\!] \to [\![A \otimes B]\!]$$
$$g : [\![\Gamma;\Delta_2, A, B]\!] \to [\![C]\!]$$

Hence we have

$$\begin{aligned}[\![\Gamma;\Delta \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ u\ \mathsf{in}\ t{:}C]\!] : &[\![\Gamma;\Delta]\!] \to [\![C]\!] \\ &= \mathsf{split}_{\Gamma,\Delta,\Delta_2,\Delta_1};(\mathsf{id}_{[\![\Gamma;\Delta_2]\!]} \otimes f);\mathsf{sep}^{-1}_{(!\Gamma,\Delta_2),(A \otimes B)};g\end{aligned}$$

⊸-introduction The rule is as follows:

$$\frac{\Gamma;\Delta, x{:}A \vdash t{:}B}{\Gamma;\Delta \vdash \lambda x.t{:}A \multimap B}$$

so we have an arrow:

$$f : [\![\Gamma;\Delta, A]\!] \to [\![B]\!]$$

Hence we have

$$\begin{aligned}[\![\Gamma;\Delta \vdash \lambda x.t{:}A \multimap B]\!] : &[\![\Gamma;\Delta]\!] \to [\![A \multimap B]\!] \\ &= \lambda(\mathsf{sep}^{-1}_{(!\Gamma,\Delta),A};f)\end{aligned}$$

⊸-elimination The rule is:

$$\frac{\Gamma;\Delta_1 \vdash u{:}(A \multimap B) \quad \Gamma;\Delta_2 \vdash t{:}A}{\Gamma;\Delta \vdash (ut){:}B}$$

where $\Delta = \Delta_1 \# \Delta_2$, so we have arrows:

$$f : [\![\Gamma;\Delta_1]\!] \to [\![A \multimap B]\!]$$
$$g : [\![\Gamma;\Delta_2]\!] \to [\![A]\!]$$

Hence we have

$$\begin{aligned}[\![\Gamma;\Delta \vdash (ut){:}B]\!] : [\![\Gamma;\Delta]\!] &\to [\![B]\!] \\ &= \mathsf{split}_{\Gamma,\Delta,\Delta_1,\Delta_2};(f \otimes g);\mathsf{ap}_{[\![A]\!],[\![B]\!]}\end{aligned}$$

!-introduction The rule is:
$$\frac{\Gamma;\_\vdash t\colon A}{\Gamma;\_\vdash !t :!A}$$

so we have an arrow:

$$f : [\![\Gamma;\_]\!] \to [\![A]\!]$$

Hence we have

$$\begin{aligned}[\![\Gamma;\_\vdash !t :!A]\!] :&[\![\Gamma;\_]\!] \to [\![!A]\!] \\ =&\mathsf{prom}_\Gamma; FG(f)\end{aligned}$$

!-elimination The rule is:

$$\frac{\Gamma;\Delta_1 \vdash u :!A \quad \Gamma, x\colon A;\Delta_2 \vdash t\colon B}{\Gamma;\Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ u\ \mathsf{in}\ t\colon B}$$

where $\Delta = \Delta_1 \# \Delta_2$. Hence we have arrows:

$$\begin{aligned}f &: [\![\Gamma;\Delta_1]\!] \to [\![!A]\!] \\ g &: [\![\Gamma, A;\Delta_2]\!] \to [\![B]\!]\end{aligned}$$

Hence we have

$$\begin{aligned}&[\![\Gamma;\Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ u\ \mathsf{in}\ t\colon B]\!] : [\![\Gamma;\Delta]\!] \to [\![B]\!] \\ &\quad = \mathsf{str}_{\Gamma,\Delta,\Delta_1,\Delta_2};(f \otimes \mathsf{id}_{[\![\Gamma;\Delta_2]\!]}); \mathsf{sep}^{-1}_{!A,(!\Gamma,\Delta_2)}; \mathsf{mge}_{(!A,!\Gamma,\Delta_2),!\Gamma,(!A,\Delta_2)}; g\end{aligned}$$

3.2 Soundness

Having given the interpretation function $[\![\_]\!]^{\mathcal{L}}$, we now need to show that the interpretation is sound. That is, we need to demonstrate that the equality judgement we have given for $\mathsf{DILL}(\mathbb{C})$ is respected by the interpretation $[\![\_]\!]^{\mathcal{L}}$ for any model $\mathcal{L}$.

First we need to prove two technical lemmas, which give the interpretations of the two substitutions in the model:

LEMMA 3.2.1
If $[\![\Gamma;\Delta_1 \vdash t\colon A]\!] = f$ and $[\![\Gamma;\Delta_2, x\colon A \vdash u\colon B]\!] = g$, and $\Delta = \Delta_1 \# \Delta_2$, then

$$[\![\Gamma;\Delta \vdash u[t/x]\colon B]\!] = \mathsf{split}_{\Gamma,\Delta,\Delta_2,\Delta_1};(\mathsf{id}_{[\![\Gamma;\Delta']\!]} \otimes f); \mathsf{sep}^{-1}_{(!\Gamma,\Delta),A}; g$$

LEMMA 3.2.2
If $[\![\Gamma;\_\vdash t\colon A]\!] = f$ and $[\![\Gamma, x\colon A;\Delta \vdash u\colon B]\!] = g$, then

$$[\![\Gamma;\Delta \vdash u[t/x]\colon B]\!] = \mathsf{split}_{\Gamma,\Delta,\_,\Delta};((\mathsf{prom}_\Gamma; FG(f)) \otimes \mathsf{id}_{[\![\Gamma;\Delta]\!]}); \mathsf{admin}_{\Gamma,\Delta,A}; g$$

where $\mathsf{admin}_{\Gamma,\Delta,A} : [\![!A]\!] \otimes [\![\Gamma;\Delta]\!] \rightarrow [\![\Gamma, A;\Delta]\!]$ is defined:

$$\mathsf{admin}_{\Gamma,\Delta,A} = (\mathsf{id}_{[\![!A]\!]} \otimes \mathsf{sep}_{!\Gamma,\Delta}); (\mathsf{s}_{[\![!A]\!],[\![!\Gamma]\!]} \otimes \mathsf{id}_{[\![\Delta]\!]}); (\mathsf{sep}^{-1}_{!\Gamma,!A} \otimes \mathsf{id}_{[\![\Delta]\!]}); \mathsf{sep}^{-1}_{!(\Gamma,A),\Delta}$$

These lemmas are proved by induction over the structure of the first term. The proofs are again left to the reader, but as an example we prove the intuitionistic lemma in the case where u is the term $\Gamma, x{:}A; \_ \vdash x{:}A$. Clearly, in this case the result of the substitution $\{t/x\}$ is just the term $\Gamma; \_ \vdash t{:}A$, so we need to show:

$$\begin{aligned} f = {} & \mathsf{split}_{\Gamma,\_,\_,\_}; ((\mathsf{prom}_\Gamma; FG(f)) \otimes \mathsf{id}_{[\![\Gamma;\_]\!]}); \mathsf{admin}_{\Gamma,\_,A}; \mathsf{sep}_{(!\Gamma,x{:}!A),\_}; \\ & (\mathsf{sep}_{(!\Gamma),!x{:}A} \otimes \mathsf{id}_{[\![\_]\!]}); ((\mathsf{disc}_\Gamma \otimes \mathsf{der}_A) \otimes \mathsf{disc}_\_); \mathsf{li}_{I \otimes [\![A]\!]}; \mathsf{ri}_{[\![A]\!]} \end{aligned}$$

We can see that $\mathsf{admin}_{\Gamma,\_,A} = \mathsf{s}_{[\![!A]\!],[\![!\Gamma]\!]}; \mathsf{sep}^{-1}_{!\Gamma,!x{:}A}$, and hence the right-hand side of this simplifies to

$$\mathsf{split}_{\Gamma,\_,\_,\_}; ((\mathsf{prom}_\Gamma; FG(f)) \otimes \mathsf{id}_{[\![\Gamma;\_]\!]}); \mathsf{s}_{[\![!A]\!],[\![!\Gamma]\!]}; \mathsf{sep}^{-1}_{!\Gamma,!x{:}A}; \mathsf{sep}_{(!\Gamma),!x{:}A}; (\mathsf{disc}_\Gamma \otimes \mathsf{der}_A); \mathsf{ri}_{[\![A]\!]}$$

But this is precisely

$$\mathsf{split}_{\Gamma,\_,\_,\_}; ((\mathsf{prom}_\Gamma; FG(f)) \otimes \mathsf{id}_{[\![\Gamma;\_]\!]}); (\mathsf{der}_A \otimes \mathsf{disc}_\Gamma); \mathsf{s}_{[\![A]\!],I}; \mathsf{ri}_{[\![A]\!]}$$

By the adjunction and the tensor rules, this is

$$\mathsf{split}_{\Gamma,\_,\_,\_}; (f \otimes \mathsf{disc}_\Gamma); \mathsf{li}_{[\![A]\!]}$$

which is

$$\mathsf{split}_{\Gamma,\_,\_,\_}; (\mathsf{id}_{[\![\Gamma;\_]\!]} \otimes \mathsf{disc}_\Gamma); \mathsf{ri}_{[\![\Gamma;\_]\!]}; f$$

But we can prove that $\mathsf{split}_{\Gamma,\_,\_,\_}; (\mathsf{id}_{[\![\Gamma;\_]\!]} \otimes \mathsf{disc}_\Gamma); \mathsf{ri}_{[\![\Gamma;\_]\!]} = \mathsf{id}_{[\![!\Gamma]\!]}$, so that the result holds.

Now we are able to prove soundness by considering the derivation of equality judgements in $\mathsf{DILL}(\mathbb{C})$.

Theorem 2 (Soundness)
If $\Gamma; \Delta \vdash tu : A$ then

$$[\![\Gamma; \Delta \vdash t{:}A]\!] = [\![\Gamma; \Delta \vdash u{:}A]\!]$$

Proof

$I - \beta$ In this case, we have

$$\Gamma; \Delta \vdash \mathsf{let} * \mathsf{be} * \mathsf{in}\ t = t : A$$

The interpretation of the left hand side is the arrow

$$\mathsf{str}_{\Gamma,\_,\Delta};(\mathsf{disc}_\Gamma \otimes g);\mathsf{ri}_{[\![A]\!]}$$

but

$$\mathsf{str}_{\Gamma,\_,\Delta};(\mathsf{disc}_\Gamma \otimes \mathsf{id}_{[\![\Gamma;\Delta]\!]}) = \mathsf{ri}^{-1}_{[\![\Gamma;\Delta]\!]}$$

so this is just g by naturality of ri.

$I-\eta$ In this case we have

$$\Gamma;\Delta \vdash \mathsf{let} * \mathsf{be}\ t\ \mathsf{in} * = t : I$$

The interpretation of the left-hand side of this is the arrow

$$\mathsf{str}_{\Gamma,\Delta,\_};(f \otimes \mathsf{disc}_\Gamma);\mathsf{ri}_I$$

but by a symmetric equality to that used above we have that this is

$$\mathsf{li}^{-1}_{[\![\Gamma;\Delta]\!]};(f \otimes \mathsf{id}_I);\mathsf{ri}_I$$

but now since $\mathsf{ri}_I = \mathsf{li}_I$ this is just f.

$\otimes-\eta$ First, note that

$$\begin{aligned}[\![\Gamma;x{:}A,y{:}B \vdash x\otimes y{:}A\otimes B]\!] &= \mathsf{str}_{\Gamma,A,B};((\mathsf{lcon}_{!\Gamma,A};(\mathsf{disc}_\Gamma \otimes \mathsf{id}_{[\![A]\!]});\mathsf{ri}_{[\![A]\!]})\\ &\qquad \otimes(\mathsf{lcon}_{!\Gamma,B};(\mathsf{disc}_\Gamma \otimes \mathsf{id}_{[\![B]\!]});\mathsf{ri}_{[\![B]\!]}))\\ &= \mathsf{lcon}_{!\Gamma,(A,B)};(\mathsf{disc}_\Gamma \otimes \mathsf{id}_{[\![A\otimes B]\!]});\mathsf{ri}_{[\![A\otimes B]\!]}\end{aligned}$$

This means that

$$\begin{aligned}[\![\Gamma;\Delta \vdash \mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ x\otimes y]\!] &= \mathsf{str}_{\Gamma,\_,\Delta};(\mathsf{id}_{[\![\Gamma;\_]\!]}\otimes f);\mathsf{a}^{-1}_{[\![\Gamma;\_]\!],A,B};\mathsf{lcon}_{!\Gamma,(AB)};\\ &\qquad (\mathsf{disc}_\Gamma \otimes \mathsf{id}_{A\otimes B});\mathsf{ri}_{[\![A]\!]})\\ &= \mathsf{str}_{\Gamma,\_,\Delta};(\mathsf{id}_{[\![\Gamma;\_]\!]}\otimes f);(\mathsf{disc}_\Gamma \otimes \mathsf{id}_{A\otimes B});\mathsf{ri}_{A\otimes B}\\ &= \mathsf{str}_{\Gamma,\_,\Delta};(\mathsf{disc}_\Gamma \otimes f);\mathsf{ri}_{A\otimes B}\\ &= f\end{aligned}$$

$!-\beta$ In this case,

$$\begin{aligned}[\![\Gamma;\Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ !t\ \mathsf{in}\ u{:}B]\!] =& \mathsf{str}_{\Gamma,\_,\Delta};((\mathsf{prom}_\Gamma;FG(f))\otimes \mathsf{id}_{[\![\Gamma;\Delta]\!]});(\mathsf{id}_{[\![!A]\!]}\otimes \mathsf{lcon}_{!\Gamma,\Delta_2});\\ &(\mathsf{s}_{[\![!A]\!],[\![!\Gamma]\!]}\otimes \mathsf{id}_{[\![\Delta_2]\!]});(\mathsf{lcon}^{-1}_{!\Gamma,!A}\otimes \mathsf{id}_{[\![\Delta_2]\!]});\mathsf{lcon}^{-1}_{!(\Gamma,A),\Delta_2};g\end{aligned}$$

which is just the interpretation of $u[t/x]$.

$\lambda - \beta$ In this case, assuming that $[\![\Gamma; \Delta_1, A \vdash t{:}B]\!] = f$ and $[\![\Gamma; \Delta_2 \vdash u{:}A]\!] = g$, we have

$$\begin{aligned}
[\![\Gamma; \Delta_1, \Delta_2 \vdash (\lambda x.t)u{:}B]\!] =& \mathsf{str}_{\Gamma,(\Delta_1,A),\Delta_2}; (\lambda(\mathsf{lcon}^{-1}_{(!\Gamma,\Delta_1),A}; f) \otimes g); \mathsf{ap}_{[\![A]\!],[\![B]\!]} \\
=& \mathsf{str}_{\Gamma,\Delta_1,\Delta_2}; (\mathsf{id}_{[\![\Gamma;\Delta_1]\!]} \otimes g); \mathsf{lcon}^{-1}_{!\Gamma,\Delta_1}; f \\
=& [\![\Gamma; \Delta_1, \Delta_2 \vdash t[u/x]{:}B]\!]
\end{aligned}$$

Congruence These cases are easily shown, firstly because categorical equality is an equivalence, and secondly since given a context $C[\_]$ and a term t, the interpretation of a term having the form $C[t]$ is a term of the internal language of the category containing a sub-term the interpretation of t.

This shows soundness. □

3.3 The Term Model

The first stage in establishing completeness is to define the term model of $\mathsf{DILL}(\mathbb{C})$. It is clear that we will as normal construct the term category to form the SMCC part of the model, but we will need to use a somewhat more complex construction to provide the CCC part.

Definition 3.3.1 (The Term Category)
We define the term category $\mathcal{S}_T$ for the signature $(\mathsf{P}_L, \mathsf{C})$ as follows:

- The objects of $\mathcal{S}_T$ are the types of $\mathsf{DILL}(\mathbb{C})$.
- $\mathcal{S}_T(A, B) = \{[(x,t)^{A,B}] | \_; x{:}A \vdash t{:}B\}$, where we write $[(x,t)^{A,B}]$ to denote the equivalence class of $(x,t)^{A,B}$ under the equivalence $\equiv$ defined by:

$$(x,t)^{A,B} \equiv (y,u)^{A,B} \quad \text{if } \_; x{:}A \vdash t = u\{x/y\} : B$$

For now on, we will write $[(x,t)]^{A,B}$ as $[x, t^{A,B}]$ for clarity, and omit the type information where possible. Also, we will assume where necessary that in referring to $[x,t]$ and $[y,u]$ etc., the pairs (x,t) and (y,u) are chosen from their equivalence classes in such a way that the variables x and y are distinct, unless they are explicitly identified.

Now define identities and substitution:

- $\mathsf{id}_A = [x, x]$
- $[x,t]; [y,u] = [x, u\{t/y\}]$

Now it is easily demonstrated that these definitions give a category, given primitive results on substitution. Further, we certainly have appropriate primitive interpretation functions $[\![\_]\!]_{\mathsf{P}_L} : \mathsf{P}_L \to \mathsf{obj}(\mathcal{S})$, given by the identity, and $[\![c]\!]_{\mathsf{C}}$ given by $[x, \mathsf{let}\ * \ \mathsf{be}\ x\ \mathsf{in}\ c]$ for each constant c in C. The next step is to show that $\mathcal{S}_T$ is in fact a SMCC. First we define the tensor of two objects A and B as the object $A \otimes B$, and on arrows we define the tensor and its associated natural isomorphisms as follows:

$$[x_1, t] \otimes [x_2, u] = [y, \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ t \otimes u]$$

$$\begin{aligned}
\mathsf{ri}_A : I \otimes A \to A &= [y, \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ x_2] \\
\mathsf{li}_A : A \otimes I \to A &= [y, \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x_2\ \mathsf{in}\ x_1] \\
\mathsf{a}_{A,B,C} &= [y_1, \mathsf{let}\ x_1 \otimes y_2\ \mathsf{be}\ y_1\ \mathsf{in}\ \mathsf{let}\ x_2 \otimes x_3\ \mathsf{be}\ y_2\ \mathsf{in}\ (x_1 \otimes x_2) \otimes x_3] \\
&: A \otimes (B \otimes C) \to (A \otimes B) \otimes C \\
\mathsf{s}_{A,B} : A \otimes B \to B \otimes A &= [y, \mathsf{let}\ x_1 \otimes x_1\ \mathsf{be}\ y\ \mathsf{in}\ x_2 \otimes x_1]
\end{aligned}$$

We can easily now show:

$$\begin{aligned}
\mathsf{ri}_A^{-1} &= [x, * \otimes x] \\
\mathsf{li}_A^{-1} &= [x, x \otimes *] \\
\mathsf{a}_{A,B,C}^{-1} &= [y_1, \mathsf{let}\ y_2 \otimes x_3\ \mathsf{be}\ y_1\ \mathsf{in}\ \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y_2\ \mathsf{in}\ x_1 \otimes (x_2 \otimes x_3)] \\
\mathsf{s}_{A,B}^{-1} &= \mathsf{s}_{B,A}
\end{aligned}$$

We give the proof of the fact that ri_A has the inverse given above as an example. In order to show that the given arrow ri has the inverse $[x, * \otimes x]$, there are two cases to consider. Firstly we consider the composition $\mathsf{ri}_A; [x, * \otimes x]$, which is

$$[y, * \otimes (\mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ x_2)]$$

Now in $\mathsf{DILL}(\mathbb{C})$ we have by two commuting conversions the equality judgement

$$\begin{aligned}
_; y{:}I \otimes A \vdash * \otimes (&\mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ x_2) \\
&= \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ (\mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ *) \otimes x_2 {:} I \otimes A
\end{aligned}$$

and by η equalities for $\otimes$ and I the equality judgement

$$\_; y{:}I \otimes A \vdash \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ (\mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ *) \otimes x_2 = y{:}A \otimes A$$

and hence

$$[y, * \otimes (\mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ x_2)] = [y, y]$$

as required. Secondly, we consider the composition $[x, * \otimes x]; \mathsf{ri}_A$, which is

$$[x, \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ * \otimes x\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ x_2]$$

But in $\mathsf{DILL}(\mathbb{C})$ we have by $\otimes$ and $I-\beta$-equalities the equality judgement

$$\_; x{:}A \vdash \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ * \otimes x\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ x_2 = x{:}A$$

so that we have

$$[x, \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ * \otimes x\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x_1\ \mathsf{in}\ x_2] = [x, x]$$

as required. □

In future we omit the explicit statements of the equality judgements which justify the equalities between equivalence classes, and merely annotate some of these equalities with the axiomatic equalities of $\mathsf{DILL}(\mathbb{C})$ used to derive them.

$\otimes$ is a Functor

We need to show that the definition given above of the tensor is functorial. This amounts to showing that identities are preserved:

$$\begin{array}{rcl} [x,x]\otimes[x',x'] & = & [y, \mathsf{let}\ x\otimes x'\ \mathsf{be}\ y\ \mathsf{in}\ x\otimes x'] \\ (\text{by } \otimes-\eta) & = & [y,y] \end{array}$$

and that composition is preserved. We show this in the second place of the functor; the proof for the first place is analogous.

$$\begin{array}{rcl} \multicolumn{3}{l}{[x,x]\otimes([y,t];[y',u])} \\ & = & [x', \mathsf{let}\ x\otimes y\ \mathsf{be}\ x'\ \mathsf{in}\ x\otimes(u\{t/y'\})] \\ (\text{by } \otimes\text{-cc and } \beta) & = & [x', \mathsf{let}\ x''\otimes y'\ \mathsf{be}\ (\mathsf{let}\ x\otimes y\ \mathsf{be}\ x'\ \mathsf{in}\ x\otimes t)\ \mathsf{in}\ x''\otimes u] \\ & = & ([x,x]\otimes[y,t]);([x'',x'']\otimes[y',u]) \end{array}$$

We now need to check that the specified natural transformations ri, li, a and s have the correct properties:

ri_A There is one diagram to check here for naturality, and two equalities for isomorphism.

$$\begin{array}{rcl} \multicolumn{3}{l}{\mathsf{ri}_A;[x,t]} \\ & = & [y, t\{\mathsf{let}\ x'\otimes x\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x'\ \mathsf{in}\ x/x\}] \\ (\text{by cc}) & = & [y, \mathsf{let}\ x'\otimes x\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ x'\ \mathsf{in}\ t] \\ (\text{by cc and } \otimes-\beta) & = & [y, \mathsf{let}\ y'\otimes y''\ \mathsf{be}\ \mathsf{let}\ x'\otimes x\ \mathsf{be}\ y\ \mathsf{in}\ x'\otimes t] \\ & & \quad \mathsf{in}\ \mathsf{let}\ *\ \mathsf{be}\ y'\ \mathsf{in}\ y'' \\ & = & (\mathsf{id}_I\otimes[x,t]);\mathsf{ri}_B \end{array}$$

This shows the naturality of ri. We previously demonstrated that ri_A was an isomorphism.

li_A The diagrams in this case are exactly analogous to the above ones, and hence are omitted.

$\mathsf{a}_{A,B,C}$ We first check naturality for this arrow.

$$
\begin{aligned}
& \mathsf{a}_{A_1,A_2,A_3};([x_1,t_1)]\otimes[x_2,t_2])\otimes[x_3,t_3] \\
= \ & [y',\ \mathsf{let}\ x_1\otimes x' \\
& \quad \mathsf{be\ let}\ y_1\otimes y''\ \mathsf{be}\ y'\ \mathsf{in\ let}\ y_2\otimes y_3\ \mathsf{be}\ y''\ \mathsf{in}\ (y_1\otimes y_2)\otimes y_3 \\
& \quad \mathsf{in\ let}\ x_2\otimes x_3\ \mathsf{be}\ x'\ \mathsf{in}\ (t_1\otimes t_2)\otimes t_3] \\
& (\text{by cc, } \alpha\text{-conv. and } \otimes-\beta) \\
= \ & [y',\mathsf{let}\ x_1\otimes x'\ \mathsf{be}\ y'\ \mathsf{in\ let}\ x_2\otimes x_3\ \mathsf{be}\ x'\ \mathsf{in}\ (t_1\otimes t_2)\otimes t_3] \\
& (\text{by cc and } \otimes-\beta) \\
= \ & [y',\ \mathsf{let}\ y_1\otimes y'' \\
& \quad \mathsf{be\ let}\ x_1\otimes x'\ \mathsf{be}\ y'\ \mathsf{in\ let}\ x_2\otimes x_3\ \mathsf{be}\ x'\ \mathsf{in}\ t_1\otimes(t_2\otimes t_3) \\
& \quad \mathsf{in\ let}\ y_2\otimes y_3\ \mathsf{be}\ y'\ \mathsf{in}\ (y_1\otimes y_2)\otimes y_3] \\
= \ & [x_1,t_1]\otimes([x_2,t_2]\otimes[x_3,t_3]);\mathsf{a}_{B_1,B_2,B_3}
\end{aligned}
$$

The isomorphism is easily seen.

$\mathsf{s}_{A,B}$ We need to show naturality:

$$
\begin{aligned}
[x,t]\otimes[y,u];\mathsf{s}_{B_1,B_2} & \\
= \ & [y'',\mathsf{let}\ x'\otimes y'\ \mathsf{be\ let}\ x\otimes y\ \mathsf{be}\ y''\ \mathsf{in}\ t\otimes u] \\
& \quad \mathsf{in}\ y'\otimes x' \\
(\text{by cc and } \otimes-\beta) = \ & [y'',\mathsf{let}\ x\otimes y\ \mathsf{be}\ y''\ \mathsf{in}\ u\otimes t] \\
(\text{by cc, } \alpha\text{-conv. and } \otimes-\beta) = \ & [y'',\mathsf{let}\ y\otimes x\ \mathsf{be\ let}\ x'\otimes y'\ \mathsf{be}\ y''\ \mathsf{in}\ y'\otimes x'] \\
& \quad \mathsf{in}\ u\otimes t \\
= \ & \mathsf{s}_{A_1,A_2};[y,u]\otimes[x,t]
\end{aligned}
$$

Again, the isomorphism is easily seen.

From now on we will omit the justifications of equality steps for brevity.

Now we need to show that the coherence equalities given earlier hold in $\mathcal{S}_T$. We check these by number based on the numbering given in appendix A.2. Because the demonstration for larger equalities consists of equalities between huge terms of $\mathsf{DILL}(\mathbb{C})$, we check here only equalities A.2.2, A.2.3, A.2.5 and A.2.6.

A.2.2

$$
\begin{aligned}
LHS = & [y',\ \mathsf{let}\ x_1\otimes x_2 \\
& \quad \mathsf{be\ let}\ y''\otimes x'''\ \mathsf{be}\ y'\ \mathsf{in\ let}\ x'\otimes x''\ \mathsf{be}\ y''\ \mathsf{in}\ x'\otimes(x''\otimes x''') \\
& \quad \mathsf{in}\ x_1\otimes(\mathsf{let}\ y_1\otimes y_2\ \mathsf{be}\ x_2\ \mathsf{in\ let}\ *\ \mathsf{be}\ y_1\ \mathsf{in}\ y_2)] \\
= & [y',\mathsf{let}\ y''\otimes x'''\ \mathsf{be}\ y'\ \mathsf{in\ let}\ x'\otimes x''\ \mathsf{be}\ y''\ \mathsf{in}\ x'\otimes(\mathsf{let}\ *\ \mathsf{be}\ x''\ \mathsf{in}\ x''')] \\
= & [y',\mathsf{let}\ y''\otimes x'''\ \mathsf{be}\ y'\ \mathsf{in}\ (\mathsf{let}\ x'\otimes x''\ \mathsf{be}\ y''\ \mathsf{in\ let}\ *\ \mathsf{be}\ x''\ \mathsf{in}\ x')\otimes x''']
\end{aligned}
$$

A.2.3

$$\begin{aligned} LHS =& [y', \mathsf{let}\ x \otimes y\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ y\ \mathsf{in}\ x] \\ =& [y', \mathsf{let}\ x \otimes y\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ x\ \mathsf{in}\ *] \\ =& [y', \mathsf{let}\ x \otimes y\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ x\ \mathsf{in}\ y] \\ =& RHS \end{aligned}$$

A.2.5 We show this by demonstrating $\mathsf{s}_{A,B}; \mathsf{s}_{B,A} = \mathsf{id}$.

$$\begin{aligned} LHS =& [y'', \mathsf{let}\ x \otimes y\ \mathsf{be}\ \mathsf{let}\ x' \otimes y'\ \mathsf{be}\ y''\ \mathsf{in}\ y' \otimes x'\ \mathsf{in}\ y \otimes x] \\ =& [y'', y''] \end{aligned}$$

A.2.6

$$\begin{aligned} LHS =& [y'', \mathsf{let}\ x \otimes y\ \mathsf{be}\ \mathsf{let}\ x' \otimes y'\ \mathsf{be}\ y''\ \mathsf{in}\ y' \otimes x'\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ y\ \mathsf{in}\ x] \\ =& [y'', \mathsf{let}\ x' \otimes y'\ \mathsf{be}\ y''\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ x'\ \mathsf{in}\ y'] \\ =& RHS \end{aligned}$$

At this point we have shown that the term category is an s.m. category. It remains to demonstrate that a suitable candidate exists for the right adjoint of the tensor.

Closedness of the SMC

We will define the functor $\multimap$ on types in the obvious way, and on morphisms $[x, t] : A \to B$, $[y, u] : A' \to B'$ and $[x', t'] : A \otimes B \to C$ as follows:

$[x, t] \multimap [y, u] : (B \multimap A') \to (A \multimap B') = [x', \lambda x{:}A.(u\{(x't)/y\})]$

$$\lambda[x', t'] : A \to (B \multimap C) = [x'', \lambda y'{:}B.(t'\{(x'' \otimes y')/x'\})]$$
$$\mathsf{ap}_{A,B} : A \otimes (A \multimap B) \to B = [y, \mathsf{let}\ x_1 \otimes x_2{:}(A \multimap B) \otimes A\ \mathsf{be}\ y\ \mathsf{in}\ x_1 x_2]$$

It is easy to show that these definitions give a functor, and that it is right adjoint to the tensor, using lemma 2.2.10. Hence the term category $\mathcal{S}_T$ is an SMCC.

Having shown that $\mathcal{S}_T$ is a SMCC, and hence forms part of a $\mathsf{DILL}(\mathbb{C})$-model, we now need to find a suitable candidate for the CCC part of this model.

The Intuitionistic Term category

We now give an explicit construction of a strict cartesian category which we will show to be adjoint to the term category.

Definition 3.3.2 (The Intuitionistic Term Category)
We define the intuitionistic term category $\mathcal{C}_T$ as follows:

- The objects of $\mathcal{C}_T$ are sequences of types of $\mathsf{DILL}(\mathbb{C})$.
- $\mathcal{C}_T(\vec{A}, \vec{B}) = \{[(\vec{x}, \vec{t})^{\vec{A},\vec{B}}] | \vec{x} : \vec{A}; \_ \vdash t_i : B_i \quad \text{for } t_i \in \vec{t}\}$, where we write $[(\vec{x}, \vec{t})^{\vec{A},\vec{B}}]$ to denote the equivalence class of $(\vec{x}, \vec{t})^{\vec{A},\vec{B}}$ under the equivalence $\equiv_C$ defined by:

$$(\vec{x}, \vec{t})^{\vec{A},\vec{B}} \equiv_C (\vec{y}, \vec{u})^{\vec{A},\vec{B}} \text{ if for all } t_i \in \vec{t},\ \vec{x}{:}\vec{A}; \_ \vdash t_i = u_i\{\vec{x}/\vec{y}\} : B_i$$

As before, we write $[\vec{x}, \vec{t}]^{\vec{A},\vec{B}}$ for $[(\vec{x}, \vec{t})]^{\vec{A},\vec{B}}$ for clarity, omit type information where possible and assume that the variables $\vec{x}$ and $\vec{y}$ in the arrows $[\vec{x}, \vec{t}]$ $[\vec{y}, \vec{u}]$ are always chosen from the equivalence classes so as to make all the variables in the concatenation $\vec{x}\vec{y}$ distinct, except where explicitly identified.

Now define identities and substitution:

- $\mathsf{id}_{\vec{A}} = [\vec{x}, \vec{x}]$
- $[\vec{x}, \vec{t}]; [\vec{y}, \vec{u}] = [\vec{x}, \vec{u}\{\vec{t}/\vec{y}\}]$

It is again easy to show that these definitions make $\mathcal{C}_T$ into a category. We now specify the strict cartesian structure on sequences of types to be concatenation, and on morphisms where $[\vec{x}, \vec{t}] : \vec{A}_1 \to \vec{B}_1$ and $[\vec{y}, \vec{u}] : \vec{A}_2 \to \vec{B}_2$:

$$[\vec{x}, \vec{t}] \times [\vec{y}, \vec{u}] : \vec{A}_1\vec{A}_2 \to \vec{B}_1\vec{B}_2 = [\vec{x}\vec{y}, \vec{t}\vec{u}]$$

$$\begin{aligned} \langle [\vec{x}, \vec{t}], [\vec{x}, \vec{u}] \rangle : \vec{A}_1 \to \vec{B}_1\vec{B}_2 &= [\vec{x}, \vec{t}\vec{u}] \\ \mathsf{p}_{i,r} : A_1 \ldots A_r \to A_i &= [\vec{x}, x_i] \end{aligned}$$

These definitions make $\mathcal{C}_T$ into a strict cartesian category; as an example, we show the defining equality of the projections:

$$\begin{aligned} \langle [\vec{x}, t_1] \ldots [\vec{x}, t_r] \rangle; \mathsf{p}_{i,r} &= [\vec{x}, t_1 \ldots t_r]; [\vec{y}, y_i] \\ &= [\vec{x}, t_i] \end{aligned}$$

It is possible to make this category into a CCC by defining an arrow type $A \to B = !A \multimap B$, but we do not do this here.

$\mathcal{S}_T$ is monoidally adjoint to $\mathcal{C}_T$

Now, in order to construct a term model for $\mathsf{DILL}(\mathbb{C})$, we need to show that the categories $\mathcal{S}_T$ and $\mathcal{C}_T$ which we have constructed are monoidally adjoint.

Note 3.3.3 (Strict Tensor)
Firstly, since we will be mapping a strict cartesian structure into the category $\mathcal{S}_T$, we will need to define some new term constructors. Define the tensor of $\otimes t_1 \ldots t_r$

$$\otimes t_1, \ldots t_r = \begin{cases} * & \text{if } r = 0 \\ t_1 & \text{if } r = 1 \\ (\otimes t_1 \ldots t_{r-1}) \otimes t_r & \text{otherwise} \end{cases}$$

Further, we need to define a let term construct to eliminate sequences of variables, which we again do as follows:

$$\mathsf{let}\ \otimes x_1 \ldots x_r\ \mathsf{be}\ t\ \mathsf{in}\ u = \begin{cases} \mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ u & \text{if } r = 0 \\ u[t/x_1] & \text{if } r = 1 \\ \mathsf{let}\ z \otimes x_r\ \mathsf{be}\ t & \\ \quad \mathsf{in}\ (\mathsf{let}\ \otimes x_1 \ldots x_{r-1}\ \mathsf{be}\ z\ \mathsf{in}\ u) & \text{otherwise} \end{cases}$$

Now, assume that $\Gamma; \Delta'', \vec{x}{:}\vec{A} \vdash u{:}B$, that $\Gamma; \Delta''' \vdash u'{:}\otimes\vec{A}$, and that $\Gamma; \Delta'_i \vdash t{:}A_i$ for $i = 1 \ldots r$, where $\vec{A} = A_1 \ldots A_r$. Then, using the β and η equalities of $\mathsf{DILL}(\mathbb{C})$, we can derive equality judgements:

$$\Gamma; \Delta \vdash \mathsf{let}\ \otimes\vec{x}\ \mathsf{be}\ \otimes\vec{t}\ \mathsf{in}\ u = u\{\vec{t}/\vec{x}\}{:}B \qquad \Gamma; \Delta''' \vdash \mathsf{let}\ \otimes\vec{x}\ \mathsf{be}\ u'\ \mathsf{in}\ \otimes\vec{x} = u'{:}\otimes\vec{A}$$

where $\Delta = \Delta''\#\Delta'_1\#\ldots\#\Delta'_r$. Now assume we have a $\Gamma_1; \Delta_1, A_1 - \Gamma_2; \Delta_2, A_2$ linear term context $C[\_]$. Given $\Gamma_1; \Delta_1, \vec{x}{:}\vec{B} \vdash u{:}A_1$, $\Gamma_2; \Delta' \vdash t'{:}\otimes\vec{B}$ and $\Gamma; \Delta''_i \vdash t_i : !B_i$ for $i = 1 \ldots r$, where $\vec{B} = B_1 \ldots B_r$, we have:

$$\Gamma; \Delta \vdash \mathsf{let}\ !\vec{x}\ \mathsf{be}\ \vec{t}\ \mathsf{in}\ \otimes!\vec{x} = \otimes\vec{t}\otimes!\vec{A}$$
$$\Gamma_2; \Delta''' \vdash C[\mathsf{let}\ \otimes\vec{x}\ \mathsf{be}\ t'\ \mathsf{in}\ u] = \mathsf{let}\ \otimes\vec{x}\ \mathsf{be}\ t'\ \mathsf{in}\ C[u]{:}A_2$$

where $\Delta''' = \Delta'\#\Delta_2$ and $\Delta = \Delta'_1\#\ldots\#\Delta_r$.

Definition 3.3.4 (The Functors F_T and G_T)
We define the functors $F_T : \mathcal{C}_T \to \mathcal{S}_T$ and $G_T : \mathcal{S}_T \to \mathcal{C}_T$:

$$F_T(\vec{A}) = \otimes!\vec{A}$$
$$F_T([\vec{x}, \vec{t}]) = [y, \mathsf{let}\ \otimes\vec{x}_1\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ !\vec{x}\ \mathsf{be}\ \vec{x}_1\ \mathsf{in}\ \otimes!\vec{t}]$$

$$G_T(A) = A$$
$$G_T([x, t]) = [x, t]$$

We show that these definitions are functorial. It is easy to see that G preserves identities and composition, but more tricky for F. To show that it preserves identities:

$$\begin{aligned}
F_T(\mathsf{id}_{\vec{A}}) &= F_T([\vec{x},\vec{x}]) \\
&= [y, \mathsf{let}\ \otimes\vec{x}_1\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ !\vec{x}\ \mathsf{be}\ \vec{x}_1\ \mathsf{in}\ \otimes!\vec{x}] \\
&= [y, \mathsf{let}\ \otimes\vec{x}_1\ \mathsf{be}\ y\ \mathsf{in}\ \otimes\vec{x}_1] \\
&= [y, y]
\end{aligned}$$

To show that it preserves composition:

$$\begin{aligned}
F_T([\vec{x},\vec{t}];[\vec{y},\vec{u}]) &= F_T([\vec{x},\vec{u}\{\vec{t}/\vec{y}\}]) \\
&= [y', \mathsf{let}\ \otimes\vec{x}_1\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ !\vec{x}\ \mathsf{be}\ \vec{x}_1\ \mathsf{in}\ \otimes!(\vec{u}\{\vec{t}/\vec{y}\})] \\
&= [y', \mathsf{let}\ \otimes\vec{x}_1\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ !\vec{x}\ \mathsf{be}\ \vec{x}_1\ \mathsf{in}\ \mathsf{let}\ !\vec{y}\ \mathsf{be}\ !\vec{t}\ \mathsf{in}\ \otimes!\vec{u}] \\
&= [y',\ \mathsf{let}\ \otimes\vec{x}_3 \\
&\quad\ \mathsf{be}\ \mathsf{let}\ \otimes\vec{x}_1\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ !\vec{x}\ \mathsf{be}\ \vec{x}_1\ \mathsf{in}\ \otimes!\vec{t} \\
&\quad\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{x}_2\ \mathsf{be}\ \otimes\vec{x}_3\ \mathsf{in}\ \mathsf{let}\ !\vec{y}\ \mathsf{be}\ \vec{x}_2\ \mathsf{in}\ \otimes!\vec{u}] \\
&= F_T([\vec{x},\vec{t}]); F_T([\vec{y},\vec{u}])
\end{aligned}$$

We now need to show that both F_T and G_T are monoidal functors. This means we must give natural transformations $\mathsf{m}^G_{A,B} : G(A)G(B) \to G(A\otimes B)$, $\mathsf{mi}^G : \_ \to G(I)$, $\mathsf{m}^F_{A,B} : F(\vec{A}) \otimes F(\vec{B}) \to F(\vec{A}\vec{B})$ and $\mathsf{mi}^F : I \to F(\_)$.

$$\begin{aligned}
\mathsf{m}^G_{A,B} &= [xy, x\otimes y] \\
\mathsf{mi}^G &= [\varepsilon, *]
\end{aligned}$$

$$\begin{aligned}
\mathsf{m}^F_{\vec{A},\vec{B}} &= [y', \mathsf{let}\ x_1\otimes x_2\ \mathsf{be}\ y' \\
&\qquad\qquad \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_1\ \mathsf{be}\ x_1\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_2\ \mathsf{be}\ x_2\ \mathsf{in}\ \otimes\vec{y}_1\vec{y}_2] \\
\mathsf{mi}^F &= [x, x]
\end{aligned}$$

We must now check naturality and certain coherence conditions.

Naturality of $\mathbf{m}^G$ Assume that $f = [x,t]$ and that $g = [y,u]$. Then

$$\begin{aligned}
LHS &= (G(f)\times G(g)); \mathsf{m}^G \\
&= [xy, tu]; [x'y', x'\otimes y'] \\
&= [xy, t\otimes u] \\
&= [xy, x\otimes y]; [y', \mathsf{let}\ x\otimes y\ \mathsf{be}\ y'\ \mathsf{in}\ t\otimes u] \\
&= \mathsf{m}^G; G(f\otimes g) \\
&= RHS
\end{aligned}$$

A.2.8) for $\mathbf{m}^F$

$$\begin{aligned}LHS &= [y', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ x_1\ \mathsf{in}\ x_2]\\ &= [y', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y'\ \mathsf{in}\ x_1 \otimes x_2];\\ &\qquad [x', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ x_1\ \mathsf{in}\ x_2]; [x', x']\end{aligned}$$

A.2.9) for $\mathbf{m}^F$

$$\begin{aligned}LHS &= [y', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y'\ \mathsf{in}\ x_2 \otimes x_1];\\ &\quad [x', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_1\ \mathsf{be}\ x_1\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_2\ \mathsf{be}\ x_2\ \mathsf{in}\ \otimes\vec{y}_1\vec{y}_2]\\ &= [y', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_1\ \mathsf{be}\ x_2\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_2\ \mathsf{be}\ x_1\ \mathsf{in}\ \otimes\vec{y}_1\vec{y}_2\\ &= [y', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_1\ \mathsf{be}\ x_1\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_2\ \mathsf{be}\ x_2\ \mathsf{in}\ \otimes\vec{y}_1\vec{y}_2];\\ &\quad [x', \mathsf{let}\ \otimes\vec{x}_1\vec{x}_2\ \mathsf{be}\ x'\ \mathsf{in}\ \otimes\vec{x}_2\vec{x}_1]\end{aligned}$$

A.2.8) for $\mathbf{m}^G$

$$\begin{aligned}LHS &= [x, x]\\ &= [x, * \otimes x]; [y', \mathsf{let}\ x \otimes y\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ x\ \mathsf{in}\ y]\\ &= [x, *x]; [xy, x \otimes y]; G([y', \mathsf{let}\ x \otimes y\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ * \ \mathsf{be}\ x\ \mathsf{in}\ y])\\ &= RHS\end{aligned}$$

A.2.9) for $\mathbf{m}^G$

$$\begin{aligned}LHS &= [xy, yx]; [xy, x \otimes y]\\ &= [xy, y \otimes x]\\ &= [xy, x \otimes y]; [y', \mathsf{let}\ x \otimes y\ \mathsf{be}\ y'\ \mathsf{in}\ y \otimes x]\\ &= [xy, x \otimes y]; G([z, \mathsf{let}\ x \otimes y\ \mathsf{be}\ z\ \mathsf{in}\ y \otimes x])\\ &= RHS\end{aligned}$$

We need to establish that there is an adjunction between $\mathcal{S}_T$ and $\mathcal{C}_T$. First we give the counit and unit:

$$\begin{aligned}[x, \mathsf{let}\ !y\ \mathsf{be}\ x\ \mathsf{in}\ y] &= \mathsf{nu}_A : FGA \to A\\ [\vec{x}, \otimes!\vec{x}] &= \mathsf{un}_{\vec{A}} : \vec{A} \to GF(\vec{A})\end{aligned}$$

Now, consider an arbitrary morphism $[x, t] : F\vec{A} \to B$ in $\mathcal{S}_T$. Define $[x, t]^*$ to be the morphism $[\vec{y}, t\{\otimes!\vec{y}/x\}] : \vec{A} \to GB$ in $\mathcal{C}_T$. But now:

$$\begin{aligned}F([x,t]^*); \mathsf{nu}_B &= [y', \mathsf{let}\ \otimes\vec{x'}\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ !\vec{y}\ \mathsf{be}\ \vec{x'}\ \mathsf{in}\ !t\{\otimes!\vec{y}/x\}]; [x, \mathsf{let}\ !y\ \mathsf{be}\ x\ \mathsf{in}\ y]\\ &= [y', \mathsf{let}\ \otimes\vec{x'}\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ !\vec{y}\ \mathsf{be}\ \vec{x'}\ \mathsf{in}\ t\{\otimes!\vec{y}/x\}]\\ &= [y', t\{y'/x\}]\end{aligned}$$

Therefore the adjunction triangle commutes. Now assume that it does so for $[\vec{y}, u] : \vec{A} \to GB$, that is to say

$$[y', \mathsf{let}\ \otimes\vec{x'}\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ !\vec{y}\ \mathsf{be}\ \vec{x'}\ \mathsf{in}\ !u] = [y', t\{y'/x\}]$$

This equality implies that

$$\_; y' {:} \otimes!\vec{A} \vdash \mathsf{let}\ \otimes\vec{x'}\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ !\vec{y}\ \mathsf{be}\ \vec{x'}\ \mathsf{in}\ !u = t\{y'/x\} {:} B$$

and by substituting $\otimes!\vec{y}$ for y' this implies that

$$\vec{y}{:}\vec{A}; \_ \vdash !u = t\{\otimes!\vec{y}/x\} {:} B$$

This implies that our choice of $[x,t]^*$ is the unique such making the adjunction triangle commute. Hence we have the required adjunction. We now need to check that the adjunction is monoidal, ie that the unit and counit of the adjunction are monoidal natural transformations. In order to check this, we need the maps $\mathsf{m}^{FG}, \mathsf{mi}^{FG}, \mathsf{m}^{GF}$ and mi^{GF}:

$$\begin{aligned}
[x, \mathsf{let}\ *\ \mathsf{be}\ x\ \mathsf{in}\ !*] &= \mathsf{m}^{FG}_I \\
I &\to FGI \\
[x', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ \mathsf{let}\ !x'_1\ \mathsf{be}\ x_1\ \mathsf{in}\ \mathsf{let}\ !x'_2\ \mathsf{be}\ x_2\ \mathsf{in}\ !(x'_1 \otimes x'_2)] &= \mathsf{m}^{FG}_{A,B} \\
FGA \otimes FGB &\to FG(A \otimes B)
\end{aligned}$$

$$\begin{aligned}
[\varepsilon, *] &= \mathsf{m}^{GF}_{_} \\
_ &\to GF(_) \\
[xy, \mathsf{let}\ \otimes\vec{x}_1\ \mathsf{be}\ x\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{x}_2\ \mathsf{be}\ y\ \mathsf{in}\ \otimes\vec{x}_1\vec{x}_2] &= \mathsf{m}^{GF}_{\vec{A},\vec{B}} \\
(GF\vec{A})(GF\vec{B}) &\to GF(\vec{A}\vec{B})
\end{aligned}$$

We now need to check certain coherence conditions:

A.2.10) for nu

$$\begin{aligned}
LHS = {} & [x', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ \mathsf{let}\ !x'_1\ \mathsf{be}\ x_1\ \mathsf{in}\ \mathsf{let}\ !x'_2\ \mathsf{be}\ x_2\ \mathsf{in}\ !(x'_1 \otimes x'_2)]; \\
& [x, \mathsf{let}\ !y\ \mathsf{be}\ x\ \mathsf{in}\ y] \\
= {} & [y', \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ !x'_1\ \mathsf{be}\ x_1\ \mathsf{in}\ \mathsf{let}\ !x'_2\ \mathsf{be}\ x_2\ \mathsf{in}\ x'_1 \otimes x'_2]
\end{aligned}$$

A.2.11) for nu

$$\begin{aligned}
LHS &= [x, \mathsf{let}\ *\ \mathsf{be}\ x\ \mathsf{in}\ !*]; [x, \mathsf{let}\ !y\ \mathsf{be}\ x\ \mathsf{in}\ y] \\
&= [x, \mathsf{let}\ *\ \mathsf{be}\ x\ \mathsf{in}\ *] \\
&= [x, x]
\end{aligned}$$

A.2.10) for un

$$\begin{aligned} LHS &= [\vec{x}_1\vec{x}_2, \otimes!\vec{x}_1\vec{x}_2] \\ &= [\vec{x}_1\vec{x}_2, (\otimes!\vec{x}_1)(\otimes!\vec{x}_2)]; [xy, \mathsf{let}\ \otimes\vec{x}_1\ \mathsf{be}\ x\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{x}_2\ \mathsf{be}\ y\ \mathsf{in}\ \otimes\vec{x}_1\vec{x}_2] \end{aligned}$$

A.2.11) for un

$$\begin{aligned} LHS &= [\varepsilon, *] \\ &= RHS \end{aligned}$$

We have now demonstrated that $\mathcal{S}_T$ is monoidally adjoint to $\mathcal{C}_T$. We now make the definition:

Definition 3.3.5 (DILL($\mathbb{C}$) Term Model)
Define the term model of $\mathsf{DILL}(\mathbb{C})$ which we will call $\mathcal{L}_T$, to be the carrier $(\mathcal{S}_T, \mathcal{C}_T, F_T, G_T)$ equipped with the primitive interpretation functions $[\![\_]\!]_{\mathsf{P}_L} : \mathsf{P}_L \to \mathsf{obj}(\mathcal{S})$ given by the identity and $[\![c]\!]_{\mathsf{C}} = [x, \mathsf{let}\ *\ \mathsf{be}\ x\ \mathsf{in}\ c]$.

We have by our previous calculations that this is a $\mathsf{DILL}(\mathbb{C})$-model over signature $(\mathsf{P}_L, \mathsf{C})$.

3.4 Completeness

In order to prove completeness, it now suffices to prove a lemma:

Lemma 3.4.1
$[\![\vec{x}'\!:\!\Gamma; \vec{x}''\!:\!\Delta \vdash t\!:\!A]\!]_{\mathcal{L}_T} = [x, \mathsf{let}\ \otimes\vec{y}\vec{x}''\ \mathsf{be}\ x\ \mathsf{in}\ \mathsf{let}\ !\vec{x}'\ \mathsf{be}\ \vec{y}\ \mathsf{in}\ t]$ in the SMCC part of the term model $\mathcal{L}_T$.

We prove this lemma by induction over the structure of the term. Now completeness is easy:

Theorem 3 (Soundness and Completeness)
Suppose we have terms $\Gamma; \Delta \vdash t\!:\!A$ and $\Gamma;\Delta \vdash u\!:\!A$, where $\Gamma = \vec{x}' : \vec{B}'$ and $\Delta = \vec{x}''\!:\!\vec{B}''$. Then $\Gamma; \Delta \vdash t = u : A$ if and only if $[\![\Gamma; \Delta \vdash t\!:\!A]\!]^{\mathcal{L}} = [\![\Gamma; \Delta \vdash u\!:\!A]\!]^{\mathcal{L}}$ in every $\mathsf{DILL}(\mathbb{C})$-model,$\mathcal{L}$.

Proof We have the forward direction of the implication, soundness, proved earlier. As for the other direction, assume that $[\![\Gamma; \Delta \vdash t\!:\!A]\!]^{\mathcal{L}} = [\![\Gamma; \Delta \vdash u\!:\!A]\!]^{\mathcal{L}}$ for all $\mathsf{DILL}(\mathbb{C})$-models $\mathcal{L}$. Then since $\mathcal{L}_T$ is an $\mathsf{DILL}(\mathbb{C})$-model we have by lemma 3.4.1 that

$$[x, \mathsf{let}\ \vec{y}\vec{x}''\ \mathsf{be}\ x\ \mathsf{in}\ \mathsf{let}\ !\vec{x}'\ \mathsf{be}\ \vec{y}\ \mathsf{in}\ t] = [x, \mathsf{let}\ \vec{y}\vec{x}''\ \mathsf{be}\ x\ \mathsf{in}\ \mathsf{let}\ !\vec{x}'\ \mathsf{be}\ \vec{y}\ \mathsf{in}\ u]$$

and hence we have

$$\_; x : \otimes!\vec{B}', \vec{B}'' \vdash \mathsf{let}\ \vec{y}\vec{x}''\ \mathsf{be}\ x\ \mathsf{in}\ \mathsf{let}\ !\vec{x}'\ \mathsf{be}\ \vec{y}\ \mathsf{in}\ t = \mathsf{let}\ \vec{y}\vec{x}''\ \mathsf{be}\ x\ \mathsf{in}\ \mathsf{let}\ !\vec{x}'\ \mathsf{be}\ \vec{y}\ \mathsf{in}\ u : A$$

But now it follows that if we substitute $\bigotimes!\vec{x}', \vec{x}''$ for x in both terms we still have an equality. However, under this substitution, we have:

$$\Gamma; \Delta \vdash \mathsf{let}\ \vec{y}\vec{x}''\ \mathsf{be}\ (\bigotimes!\vec{x}', \vec{x}'')\ \mathsf{in}\ \mathsf{let}\ !\vec{x}'\ \mathsf{be}\ \vec{y}\ \mathsf{in}\ t = t{:}A$$

and hence

$$\Gamma; \Delta \vdash t = u : A$$

as required. □

3.5 ILL and Other Models

Our proof of completeness above has certain easy corollaries. Firstly, since we have shown that the type theory $\mathsf{ILL}(\mathbb{C})$ together with its $\beta\eta$-cc equality is isomorphic to a subsystem of $\mathsf{DILL}(\mathbb{C})$, with its $\beta\eta$-cc equality, we know that the models of $\mathsf{DILL}(\mathbb{C})$ will be very closely related to models of $\mathsf{ILL}(\mathbb{C})$. We give some results which are easily proved.

Lemma 3.5.1 (Interpretation for ILL($\mathbb{C}$))
If $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M{:}A$, then we have an arrow $[\![\_; \Delta \vdash \Phi(M){:}A]\!] : [\![\Delta]\!] \to [\![A]\!]$ in the SMCC part of any $\mathsf{DILL}(\mathbb{C})$-model.

This is obvious from the form of the maps $[\![\_]\!]$ and Φ.

Corollary 3.1 (Soundness for ILL($\mathbb{C}$))
If $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M = N : A$, then the arrows $[\![\_; \Delta \vdash \Phi(M){:}A]\!]$ and $[\![\_; \Delta \vdash \Phi(N){:}A]\!]$ are equal in the SMCC part of any $\mathsf{DILL}(\mathbb{C})$-model.

This follows from the fact that Φ preserves equalities, and from the fact that $[\![\_]\!]$ is sound.

Corollary 3.2 (Soundness and Completeness for ILL($\mathbb{C}$))
For terms $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M{:}A$ and $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} N{:}A$, $\Delta \vdash M = N : A$ if and only if

$$[\![\_; \Delta \vdash \Phi(M){:}A]\!] = [\![\_; \Delta \vdash \Phi(N){:}A]\!]$$

in the SMCC part of every $\mathsf{DILL}(\mathbb{C})$-model.

Proof Clearly by completeness, $[\![\_;\Delta \vdash \Phi(M):A]\!] = [\![\_;\Delta \vdash \Phi(N):A]\!]$ iff $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(M) = \Phi(N) : A$, and this is true iff $\Delta \vdash M = N : A$ by our translation results.

Hence we have shown that $\mathsf{ILL}(\mathbb{C})$ is complete for $\mathsf{DILL}(\mathbb{C})$-models.

Relating $\mathsf{DILL}(\mathbb{C})$-models and Cambridge Models

We now recall that together with the original presentation of $\mathsf{ILL}(\mathbb{C})$, in [BBdPH93b], a categorical model was given based on a SMCC with a comonoidal comonad. These models were referred to as linear categories, or sometimes as Cambridge categories. It is natural to ask how this model of linear logic relates to the $\mathsf{DILL}(\mathbb{C})$-models we have used, and indeed this question has been considered in detail by Benton in [Ben95a], with respect to his LNL-models. We summarise two of his results.

Lemma 3.5.2 ([Ben95a], Corollary 8)
Any LNL-model has as its SMCC part a linear category.

Lemma 3.5.3 ([Ben95a], Corollary 17)
Any linear category is the SMCC part of at least one LNL-model.

This last lemma is of particular interest because the construction of a suitable CC part to make the LNL-model can be accomplished in a variety of ways. We can now prove the following lemma, which neatly confirms the claim made in [BBdPH93a]:

Theorem 4 (ILL($\mathbb{C}$) and linear categories)
For two terms $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M\!:\!A$ and $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} N\!:\!A$ of $\mathsf{ILL}(\mathbb{C})$, $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M = N : A$ iff

$$[\![\_;\Delta \vdash \Phi(M)\!:\!A]\!]^{\mathcal{L}} = [\![\_;\Delta \vdash \Phi(N)\!:\!A]\!]^{\mathcal{L}}$$

in every $\mathsf{DILL}(\mathbb{C})$-model $\mathcal{L}$.

Proof We have via completeness that $[\![\_;\Delta \vdash \Phi(M)\!:\!A]\!]_{\mathcal{L}} = [\![\_;\Delta \vdash \Phi(N)\!:\!A]\!]_{\mathcal{L}}$ iff $\_;\Delta \vdash_{\mathsf{DILL}(\mathbb{C})} \Phi(M) = \Phi(N) : A$, and by translation results that this is true iff $\Delta \vdash_{\mathsf{ILL}(\mathbb{C})} M = N : A$. $\square$

Chapter 4

Linear Type Theories

We now present a generalised linear type-theory, which consists of a basic structural part similar to the structural part of DILL, operator rules for operators in a given signature, and axiomatic equalities over terms given by a primitive axiom set. This linear type-theory is general enough to have as instances both $\mathsf{DILL}(\mathbb{C})$ and also all the static action calculi of Milner.

4.1 Introduction

Operators

Historically, many intuitionistic logics and typing systems have been constructed on the familiar foundation of the intuitionistic axiom $\Gamma, A \vdash A$ and cut rule

$$\frac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash B}$$

including some typing systems which are not based on any other logical structure. In order to consider such systems, we recall Aczel's general binding operators [Acz80]. In an natural deduction formulation, consider the usual rule for $\vee$-elimination:

$$\frac{\begin{array}{cc} (A) & (B) \\ \vdots & \vdots \\ C & C \end{array} \quad A \vee B}{C}$$

where formulae A and B are discharged from the assumptions. More generally, we will allow a general operator rule to take an arbitrary number of proofs as arguments, and to discharge any assumptions from those proofs, giving a deduction of any formula. A suitable definition of operator along these lines encompasses all the common introduction and elimination rules of most logics based on these structural rules. Now consider annotations of proofs with terms in such a system. The type theoretic formulation corresponding to the $\vee$-elimination rule given above involves a "cases construction":

$$\frac{\Gamma, x{:}A \vdash u{:}C \quad \Gamma, y{:}B \vdash v{:}C \quad \Gamma \vdash t{:}A \vee B}{\Gamma \vdash \mathsf{cases}((x{:}A)u, (y{:}B)v, t){:}C}$$

Here, notice that the discharging of assumptions A and B in the operator rule corresponds via the Curry-Howard correspondence to the binding of the variables x and y in u and v, the annotations of the corresponding deduction, respectively. The typing rule for a general such operator is:

$$\frac{\Gamma, \vec{x}_1{:}\vec{A}_1 \vdash t_1{:}B_1 \ldots \Gamma, \vec{x}_r{:}\vec{A}_r \vdash t_r{:}B_r \qquad \Gamma \vdash u_1{:}C_1 \ldots \Gamma \vdash u_s{:}C_s}{\Gamma \vdash \mathsf{K}((\vec{x}_1{:}\vec{A}_1)t_1, \ldots, (\vec{x}_r{:}\vec{A}_r)t_r, u_1, \ldots, u_s){:}B}$$

where each $\vec{x}_i{:}\vec{A}_i$ denotes the sequence of distinct variables which are bound in the ith component. This general theory can be pushed through into the semantics as well, yielding a model in which operators are interpreted as natural transformations between the hom-sets representing proofs of the appropriate types.

Linearity

Considering $\mathsf{DILL}(\mathbb{C})$ and its properties, we notice that many of the characterising rules, which in $\mathsf{ILL}(\mathbb{C})$ involve the type constructor !, in $\mathsf{DILL}(\mathbb{C})$ arise as properties of the structural rules. Consider the 'structural fragment' of the logic $\mathsf{DILL}(\mathbb{C})$, which consists of the following rules:

$$(Int - Ax)\ \Gamma, A; \_ \vdash A \qquad\qquad (Lin - Ax)\ \Gamma; A \vdash A$$

and the (derivable) cut rules:

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, A \vdash B}{\Gamma; \Delta_1, \Delta_2 \vdash B}\ (L - Cut) \qquad \frac{\Gamma; \_ \vdash A \quad \Gamma, A; \Delta_2 \vdash B}{\Gamma; \Delta \vdash B}\ (I - Cut)$$

We can see within the rules which are admissible in this very restricted system that we clearly have intuitionistic structure, given by the intuitionistic axiom and cut rules, and correspondingly linear structure, given by the linear axiom and cut rules. In the same way as many systems are viewed as intuitionistic by virtue of the structural rules they have as a basis, many systems are viewed and described as 'linear' due to their use of the linear structural rules and context maintenance disciplines, whilst not being based on the full linear logic. Hence we might well wish to consider a system of general operators based on the structural fragment of $\mathsf{DILL}(\mathbb{C})$, along the same lines as that described above.

First, however, there is one more important generalisation to make. When we consider the logic $\mathsf{DILL}(\mathbb{C})$, we see that a sequent $\Gamma; \Delta \vdash A$ is equivalent to a sequent $\_; !\Gamma; \Delta \vdash A$. This reflects an underlying fact about the logic, which is that all types are viewed as linear, and the intuitionistic fragment of the logical system is constructed over the linear types. Now in many applications, for example Moggi's computational λ-calculus [Mog89], exactly the opposite view is taken, which is to say that all types are essentially intuitionistic (or *value types*), and that the linearly constructed type-system over them is viewed as a a system of computations. Clearly, in order to account for these equally valid and interchangeable views, it makes sense for us to allow both intuitionistic and linear types. In keeping however with the approach of $\mathsf{DILL}(\mathbb{C})$, which is focussed more on the linearity of a system, we will view intuitionistic types or formulae as a subset of linear types or formulae.

Therefore, we will take a general sequent $\Gamma; \Delta \vdash A$ in which the formulae in Γ must be intuitionistic, and those in Δ, A are linear, although since each intuitionistic type is also linear they may be intuitionistic. If we write Q for a general intuitionistic formula and A for a general linear formula, then the structural rules are as follows:

$$(Int - Ax)\ \Gamma, Q; \_ \vdash Q \qquad\qquad (Lin - Ax)\ \Gamma; A \vdash A$$

with the (derivable) cut rules:

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, A \vdash B}{\Gamma; \Delta_1, \Delta_2 \vdash B}\ (L - Cut) \qquad \frac{\Gamma; \_ \vdash Q \quad \Gamma, Q; \Delta_2 \vdash A}{\Gamma; \Delta \vdash A}\ (I - Cut)$$

and the new rule:

$$\frac{\Gamma; \Delta_1 \vdash Q \quad \Gamma, Q; \Delta_2 \vdash A}{\Gamma; \Delta_1, \Delta_2 \vdash A}\ (F)$$

Now this new F-rule is needed because in general, we have another cut-like operation which allows us to use the image of the intuitionistic structure (weakening and contraction) in the linear part *on an intuitionistic type*. This was not needed previously as we had no intuitionistic types.

Given this framework, a general operator will be able to bind arbitrary sequences of linear and intuitionistic variables in each of its arguments. It is also convenient to allow a sequent-style presentation of the operator theory, in which as well as discharging assumptions (binding variables) an operator instance can introduce new assumptions (free variables). This generalisation does not change the expressive power of the system, but is convenient for the representation of the operators we shall consider. We will discuss this further after the typing system has been introduced.

The final amendment which we will make to the straightforward introduction of operators into this framework is in considering naturality. As can be seen simply by considering the form of the typing rule for the cases-construct and its corresponding operator, substitution for any variable in the intuitionistic context Γ commutes through the application of the operator transparently. This corresponds to the interpretation of operators of this kind as natural transformations in the semantics. However, in the case where we have two distinct parts to the typing context, we might ask what the corresponding property is on interpretation. It turns out that to allow such transparent substitution on the intuitionistic context is to insist that an operator be natural in the linear category over the intuitionistic category, and to allow such transparent substitution in the linear context is to insist that an operator be natural in the linear category over the linear category.

Considering the operators we will be using, not all of them are linearly natural in each argument, and so we allow a general operator to have two sets of arguments, one in which it is only intuitionistically natural and one in which it is linearly natural as well.

4.2 The Generalised Logic

Firstly, we present the propositions of the logic. We assume two sets $\mathsf{M}_I \subseteq \mathsf{M}_L$ of primitive intuitionistic propositions and primitive linear propositions, ranged over by $Q, R \ldots$ and $A, B \ldots$ respectively. (Note that we have overloaded $A, B \ldots$ to refer both to formulae of DILL and linear arities; where it is not clear from the context, we shall explicitly state which reading is intended.) Now a *generalised logical context* is a pair consisting of a set of intuitionistic formulae and a multiset of linear formulae. In the usual way, we write such a pair as $\Gamma; \Delta$, where Γ is the *intuitionistic* part, or set of intuitionistic formulae, and Δ is the *linear* part, or multiset of linear formulae.

We also assume two sets $\mathcal{O}_I \subseteq \mathcal{O}_L$ of operators, ranged over by $O \ldots$. To each of these operators is associated an arity of the form:

$$\frac{(\vec{Q}_1; \vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r; \vec{A}_r)B_r \quad ; \quad (\vec{Q}'_1; \vec{A}'_1)B'_1, \quad \ldots, \quad (\vec{Q}'_s; \vec{A}'_s)B'_s}{(\vec{Q}''; \vec{A}'')B''}$$

where we insist that any operator in $\mathcal{O}_I$ which we will call *intuitionistic*, must have an arity of the form:

$$\frac{(\vec{Q}_1; \vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r; \vec{A}_r)B_r;}{(;)R}$$

Together, the sets M_I, M_L, $\mathcal{O}_I$ and $\mathcal{O}_L$ make up a *generalised signature* ; we let such generalised signatures be ranged over by $\mathbb{O} = (\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$.

The general form of the arity of an operator given above should be read as saying that the operator O has r arguments in which it is intuitionistically natural, and s arguments in which it is intuitionistically and linearly natural. We consider operators which have arguments that they are not linearly natural in because several operators of interest fall into this category, notably the !-operator of $\mathsf{DILL}(\mathbb{C})$ and the operators of Milner's action calculi, which all are only intuitionistically natural. The general form for the arity of an intuitionistic operator also given above is more restricted because of certain properties we will require of all such operators, in the equality of the type-theory. The leading example of an intuitionistic operator is given by the !-introduction of $\mathsf{DILL}(\mathbb{C})$, which in this framework will be an intuitionistic operator of arity:

$$\frac{(;)A;}{(;)!A}$$

because intuitively it represents the lifting of a general linear term to be in the intuitionistic world.

Though the general operator signature is quite complex, it would be possible to start with a signature containing just intuitionistically natural operators, and to define these less primitive notions using the equality theory; for example we could equip a simple type theory with a linearly natural operator by giving it a family of intuitionistically natural operators and adding certain equality judgements to the set of axioms of the type theory.

Before proceeding, we introduce some notation. Many operators that we will introduce have a general arity in which either r or s is zero, or they have no binding behaviour for a given argument. We will represent this absence of arguments or binding behaviour by the absence of any marker- for example, the operator ! with no linearly natural arguments and just one intuitionistically natural argument with no binding behaviour has arity:

$$\frac{(;)A \quad ;}{(;)!A}$$

We can now present the logic, which we call $\mathsf{Lin}(\mathbb{O})$.

Definition 4.2.1 (The Logic Lin($\mathbb{O}$))
We say that a sequent $\Gamma; \Delta \vdash A$ can be derived, for a linear formula A and a

generalised logical context $\Gamma;\Delta$, if this can be shown using the following rules:

$$(Int - Ax)\ \Gamma, Q; \_ \vdash Q \qquad\qquad (Lin - Ax)\ \Gamma; A \vdash A$$

$$\frac{\Gamma;\Delta_1 \vdash Q \quad Q,\Gamma;\Delta_2 \vdash A}{\Gamma;\Delta_1,\Delta_2 \vdash A}\ F$$

$$\frac{\begin{array}{ll}\Gamma,\vec{Q}_1;\vec{A}_1 \vdash B_1 \ \ldots\ \Gamma,\vec{Q}_r;\vec{A}_r \vdash B_r & \Gamma;\Delta'_1 \vdash Q''_1 \ \ldots\ \Gamma;\Delta'_{r'} \vdash Q''_{r'} \\ \Gamma,\vec{Q}'_1;\Delta_1,\vec{A}'_1 \vdash B'_1 \ \ldots\ \Gamma,\vec{Q}'_s;\Delta_s,\vec{A}'_s \vdash B'_s & \Gamma;\Delta''_1 \vdash A''_1 \ \ldots\ \Gamma;\Delta''_{s'} \vdash A''_{s'}\end{array}}{\Gamma;\Delta \vdash B''}\ (O)$$

where $\Delta = \Delta_1,\ldots,\Delta_s,\Delta'_1,\ldots\Delta'_{r'},\Delta''_1,\ldots,\Delta''_{s'}$ and where this last rule may be used for any operator $O \in \mathcal{O}_L$ having arity:

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r;\vec{A}_r)B_r \quad ; \quad (\vec{Q}'_1;\vec{A}'_1)B'_1, \quad \ldots, \quad (\vec{Q}'_s;\vec{A}'_s)B'_s}{(\vec{Q}'';\vec{A}'')B''}$$

where $\vec{Q}'' = Q''_1 \ldots Q''_{r'}$ and $\vec{A}'' = A''_1 \ldots A''_{s'}$.

We briefly explain these rules. The axioms are self-explanatory, and familiar from the presentation of DILL. The F-rule, which resembles a non-conventional intuitionistic cut, is justified because it is only allowed in the case where the 'cut' formula is intuitionistic. For the operator rule, we assume a derivation of each of the sequents appearing above the line in order to deduce the sequent below the line. The rule assumes firstly premises for the linearly natural and the intuitionistically natural arguments of the operator, and secondly premises demonstrating the propositions which are the fresh inputs of the conclusion of the operator. These are necessary so that both the appropriate cut rules will be admissible for the logic, or equivalently to give an adequate notion of substitution in the type theory.

The Structural rules

Given the operator rule presented above, we have the following three structural rules, as expected; a weakening rule and linear and intuitionistic cut rules:

$$\frac{\Gamma;\Delta \vdash A}{\Gamma,Q;\Delta \vdash A}\ Weak$$

$$\frac{\Gamma;\Delta_1 \vdash A \quad \Gamma;\Delta_2,A \vdash B}{\Gamma;\Delta_1,\Delta_2 \vdash B}\ L \qquad\qquad \frac{\Gamma;\_ \vdash Q \quad \Gamma,Q;\Delta \vdash A}{\Gamma;\Delta \vdash A}\ I$$

These are all admissible. We note that the intuitionistic cut is redundant in this logical setting since it is a special case of the F-rule. This redundancy will disappear in the type-theory because these two rules will have different actions on terms, in general.

4.3 The Typing System $\mathsf{Lin}(\mathbb{O})$

Having given this logic, we can now proceed to annotate its contexts and derivations with variables and terms respectively. We assume the same set of variables used in $\mathsf{DILL}(\mathbb{C})$, X, ranged over as before by $x, y \ldots$, and construct pre-terms as follows:

Definition 4.3.1 (Pre-Terms)
We define *pre-terms over a signature* $(\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$, ranged over by $v, w \ldots$ as follows:

$$\begin{aligned} v ::= & x \mid \mathsf{let}\ x{:}Q\ \mathsf{be}\ v\ \mathsf{in}\ v \\ & \mid O((\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})v, \ldots, (\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})v; (\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})v, \ldots, (\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})v)(\vec{v};\vec{v}) \end{aligned}$$

where we have an operator clause for every operator $O \in \mathcal{O}_L$.

We omit typing information in pre-terms where convenient. The binding behaviour of the let construct is familiar, and the operator construct

$$O((\vec{x}_1;\vec{y}_1)v_1, \ldots, (\vec{x}_r;\vec{y}_r)v_r; (\vec{x}'_1;\vec{y}'_1)w_1, \ldots, (\vec{x}'_s;\vec{y}'_s)w_s)(v'_1, \ldots, v'_{r'}; w'_1, \ldots, w'_{s'})$$

binds the variables $\vec{x}_i$ and $\vec{y}_i$ in v_i for all i from 1 to r, and the variables $\vec{x}'_j$ and $\vec{y}'_j$ in w_j respectively for all $j = 1 \ldots s$. We assume the usual capture-avoiding substitution $v\{w/x\}$ and we may also use a simultaneous form $v\{\vec{w}/\vec{x}\}$.

We can now define the notion of the multiset of *free variables* of a pre-term v, which we write $\mathsf{FV}(v)$, unambiguously overloading our previous notation. In the following, we again use the mixed complement $M - S$, where M is a multiset and S is a set, for the multiset resulting when all copies of anything in S are removed from M.

$$\begin{aligned} \mathsf{FV}(x) &= \{x\} \\ \mathsf{FV}(\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w) &= (\mathsf{FV}(w) - \{x\}) \cup \mathsf{FV}(v) \\ & \mathsf{FV}(O((\vec{x}_1;\vec{y}_1)v_1, \ldots, (\vec{x}_r;\vec{y}_r)v_r; (\vec{x}'_1;\vec{y}'_1)w_1, \ldots, (\vec{x}'_s;\vec{y}'_s)w_s)(\vec{v}';\vec{w}')) \\ & \quad = (\bigcup_{i=1\ldots r} \mathsf{FV}(v_i) - \{\vec{x}_i\vec{y}_i\}) \cup (\bigcup_{j=1\ldots s} \mathsf{FV}(w_j) - \{\vec{x}'_j\vec{y}'_j\}) \cup \\ & \quad (\bigcup_{i'=1\ldots r'} \mathsf{FV}(v'_{i'})) \cup (\bigcup_{j'=1\ldots s'} \mathsf{FV}(w'_{j'})) \end{aligned}$$

We again recall the definitions of appendix A, and add to them by saying that a *linear typing* is a typing of a variable from the set X with a linear type, and conversely an *intuitionistic typing* is a typing of a variable with an intuitionistic type. Then an *intuitionistic typing sequence* is a sequence of intuitionistic typings, and similarly for linear typing sequences. Then a *typing context* is a pair of an intuitionistic typing sequence and a linear typing sequence, written as usual as $\Gamma;\Delta$ where Γ is the intuitionistic part and Δ is the linear part.

We now give the typing rules of the generalised type system over a signature $\mathbb{O}$, which we will refer to again as $\mathsf{Lin}(\mathbb{O})$.

Definition 4.3.2 (The Typing Rules of Lin($\mathbb{O}$))
We say a typing judgement $\Gamma;\Delta \vdash v\!:\!A$ can be derived, for a typing context $\Gamma;\Delta$, a pre-term v and a linear type A, when this can be shown using the following rules:

$$(I - Ax)\ \Gamma, x\!:\!Q, \Gamma';\_ \vdash x\!:\!Q \qquad\qquad (L - Ax)\ \Gamma; x\!:\!A \vdash x\!:\!A$$

$$\frac{\Gamma;\Delta_1 \vdash v\!:\!Q \quad x\!:\!Q,\Gamma;\Delta_2 \vdash w\!:\!A}{\Gamma;\Delta \vdash \mathsf{let}\ x\!:\!Q\ \mathsf{be}\ v\ \mathsf{in}\ w\!:\!A}\ (F) \ \text{(where } \Delta = \Delta_1\#\Delta_2)$$

$$\frac{\begin{array}{ll}\Gamma,\vec{x}_1\!:\!\vec{Q}_1;\vec{y}_1\!:\!\vec{A}_1 \vdash v_1\!:\!B_1 \ \ldots\ \Gamma,\vec{x}_r\!:\!\vec{Q}_r;\vec{y}_r\!:\!\vec{A}_r \vdash v_r\!:\!B_r & \Gamma;\Delta'_1 \vdash v'_1\!:\!Q''_1 \ \ldots\ \Gamma;\Delta'_{r'} \vdash v'_s\!:\!Q''_{r'} \\ \Gamma,\vec{x}'_1\!:\!\vec{Q}'_1;\Delta_1,\vec{y}'_1\!:\!\vec{A}'_1 \vdash w_1\!:\!B'_1 \ \ldots & \Gamma;\Delta''_1 \vdash w'_1\!:\!A''_1 \ \ldots\ \Gamma;\Delta''_{s'} \vdash w'_{s'}\!:\!A''_{s'} \\ \qquad \Gamma,\vec{x}'_s\!:\!\vec{Q}'_s;\Delta_s,\vec{y}'_s\!:\!\vec{A}'_s \vdash w_s\!:\!B'_s & \end{array}}{\Gamma;\Delta \vdash O((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(v'_1,\ldots,v'_{r'};w'_1,\ldots,w'_{s'})\!:\!B''}\ (O)$$

where $\Delta = \Delta_1\#\ldots\#\Delta_s\#\Delta'_1\#\ldots\#\Delta'_{r'}\#\Delta''_1\ldots\Delta''_{s'}$, and there is an instance of this last rule for any operator O with arity

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r;\vec{A}_r)B_r \quad ; \quad (\vec{Q}'_1;\vec{A}'_1)B'_1, \quad \ldots, \quad (\vec{Q}'_s;\vec{A}'_s)B'_s}{(\vec{Q}'';\vec{A}'')B''}$$

where $\vec{Q}'' = Q''_1\ldots Q''_{r'}$ and $\vec{A}'' = A''_1\ldots A''_{s'}$.

Definition 4.3.3 (Term)
For a typing context $\Gamma;\Delta$ and a linear type A, a $\Gamma;\Delta$-*term* v having type A of $\mathsf{Lin}(\mathbb{O})$ is a pre-term of $\mathsf{Lin}(\mathbb{O})$ such that we can derive $\Gamma;\Delta \vdash v\!:\!A$. We will often say "the term $\Gamma;\Delta \vdash v\!:\!A$" meaning "the $\Gamma;\Delta$ term v of type A".

We now say that, given a typing judgement $\Gamma;\Delta \vdash v\!:\!A$, a variable $x \in \mathsf{FV}(v)$ is an *intuitionistic free variable* of v if it occurs in $\mathsf{dom}(\Gamma)$, and $x \in \mathsf{FV}(v)$ is an *linear free variable* of v if it occurs in $\mathsf{dom}(\Delta)$.

Operators and Natural Deduction

Now that we have presented the basic typing system and shown how operators are assigned arities, it is worth discussing the process by which a sequent-style operator rule becomes a natural deduction typing rule. Our presentation of operators as primitives has been sequent-style from the outset, since we allow new assumptions in the result of an operator. If we were to directly adopt the operator arity rule as the logical rule for the operator, then in our natural deduction system a cut into one of these new assumptions would not be eliminable. In fact, this problem is a generalisation of a problem which historically arose in connection with the promotion rule of ILL, which was first presented for example in [Abr93] as:

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\ Prom$$

In that rule, we intuitively bind all the inputs $!\Gamma$ and introduce fresh ones of the same types, since this operator is not linearly natural. However, if we take this rule then it is not possible to eliminate a cut into one of the assumptions. The solution to this problem as proposed by Benton *et al.* was to incorporate the possibility of a cut into any of the assumptions, giving the rule:

$$\frac{\Delta_1 \vdash !B_1 \quad \ldots \quad \Delta_r \vdash !B_r \quad !B_1 \ldots !B_r \vdash A}{\Delta_1, \ldots \Delta_r \vdash !A}\ Prom'$$

This is precisely the approach we have adopted, incorporating the possibility of a cut into any new assumption in the output of an operator, and this approach is the canonical way of turning a sequent system into a natural deduction one.

We choose to work with sequent-style operators partly because they are easier to work with and also since Milner's action calculi have a sequent-style approach inherited from their categorical underpinning.

We can now prove the following typing properties of $\mathsf{Lin}(\mathbb{O})$.

Lemma 4.3.4 (Typing Properties)
We have the following in the system $\mathsf{Lin}(\mathbb{O})$:

Free Variables I If $\Gamma; \Delta \vdash v\!:\!A$, then the underlying set of the multiset $\mathsf{FV}(v)$ is a subset of $\mathsf{dom}(\Gamma) \cup \mathsf{dom}(\Delta)$.

Free Variables II If $\Gamma; x\!:\!A, \Delta \vdash v\!:\!B$, then x occurs precisely once in $\mathsf{FV}(v)$.

Strengthening If $\Gamma, x\!:\!Q; \Delta \vdash v\!:\!B$ and $x \notin \mathsf{FV}(v)$, then $\Gamma; \Delta \vdash v\!:\!B$.

I-Transfer If $\Gamma; \Delta, x\!:\!Q \vdash v\!:\!A$, then $\Gamma, x\!:\!Q; \Delta \vdash v\!:\!A$.

Unique Derivation Given a typing context $\Gamma; \Delta$ and a linear type A, for each $\Gamma; \Delta$ term v of type A there is a unique derivation of the typing judgement $\Gamma; \Delta \vdash v{:}A$. Further, given a pre-term v ad a typing context $\Gamma; \Delta$, there is at most one A such that v is a $\Gamma; \Delta$-term of type A.

The proofs of these properties are simple. For example, we prove that every term has a unique derivation.

Proof We can prove this by induction over the pre-term structure of v. If the pre-term is a variable, it is clear that there is only one possible derivation, which will use the intuitionistic or linear axiom depending on where the variable is typed in the context. If the pre-term is a let-clause, say $\mathsf{let}\ x{:}Q\ \mathsf{be}\ v\ \mathsf{in}\ w$, we know that all the linear free variables of v and w are typed in Δ. But then since each distinct free variable must occur either in v or w, we know that $\Delta = \Delta_1 \# \Delta_2$ where Δ_2 is the unique subsequence of Δ which types precisely the linear free variables of v and Δ_2 is the unique subsequence which types precisely the linear free variables in w. Now, if we take $\Delta = \Delta_1' \# \Delta_2'$, where $\Delta_1 \neq \Delta_1'$ and $\Delta_2 \neq \Delta_2'$, we can see by our free variable lemma that v will not be a $\Gamma; \Delta_1' - Q$ term and w will not be a $\Gamma; \Delta_2'$ term of type A. Hence the only possible typing derivation of the $\mathsf{let}\ x{:}Q\ \mathsf{be}\ v\ \mathsf{in}\ w$ is constructed from the unique derivation of the typings $\Gamma; \Delta_1 \vdash v{:}Q$ and $\Gamma; \Delta_2 \vdash w{:}A$. If the pre-term is an operator instance

$$O((\vec{x}_1; \vec{y}_1)v_1, \ldots, (\vec{x}_r; \vec{y}_r)v_r; (\vec{x}_1'; \vec{y}_1')w_1, \ldots, (\vec{x}_s'; \vec{y}_s')w_s)(\vec{v}'; \vec{w}')$$

we have precisely the multiary case of the same problem, where we have a unique partition of Δ into subsequences based on the linear free variables of the $\vec{w}\vec{v}'\vec{w}'$. □

The Substitution Rules

We can now give typed versions of the two cut rules given earlier, with some other typed structural rules:

$$\frac{\Gamma, x{:}Q, y{:}R, \Gamma'; \Delta \vdash v{:}A}{\Gamma, y{:}R, x{:}Q, \Gamma'; \Delta \vdash v{:}A}\ (I - Exch) \qquad \frac{\Gamma; \Delta, y{:}B, x{:}A, \Delta' \vdash v{:}C}{\Gamma; \Delta, x{:}A, y{:}B, \Delta' \vdash v{:}C}\ (L - Exch)$$

$$\frac{\Gamma; \Delta \vdash v{:}A}{\Gamma, x{:}Q; \Delta \vdash v{:}A}\ (Weak) \qquad \frac{\Gamma, x{:}Q, y{:}Q; \Delta \vdash v{:}A}{\Gamma, x{:}Q; \Delta \vdash v\{x/y\}{:}A}\ (Cont)$$

$$\frac{\Gamma; \_ \vdash w{:}Q \quad \Gamma, x{:}Q; \Delta \vdash v{:}A}{\Gamma; \Delta \vdash v\{w/x\}{:}A}\ (I - Cut) \qquad \frac{\Gamma; \Delta_1 \vdash w{:}A \quad \Gamma; \Delta_2, x{:}A \vdash v{:}B}{\Gamma; \Delta \vdash v\{w/x\}{:}B}\ (L - Cut)$$

where in the linear cut rule, $\Delta = \Delta_1 \# \Delta_2$. These are all admissible typing rules. In particular, the linear exchange is admissible by virtue of the merging that we have incorporated into the F and O rules.

Constants

It will be convenient to consider a particular class of operators, those with arity

$$\frac{;}{(;)A}$$

for some $A \in \mathsf{M}_L$. An operator O of this form for a given A will have a typing rule of the form:

$$\overline{\Gamma; \_ \vdash O(;)(;) : A}$$

We will refer to such an operator as a *constant of arity* A, and abbreviate the term form $O(;)(;)$ as O where convenient.

4.4 The Type Theory $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$

We will now present the equality judgement of the type theory. This has two parts, in a similar way to the typing system. We have the structural equality judgements for congruence and the F-equality, and we also assume an *axiom set* of typed equalities (to capture the behaviour of the operators).

First we need to define the notion of term-context, in order to define congruence. We define a *pre-context*, written $C[\_]$, as follows:

$$\begin{aligned}
C[_] ::= _ \mid & \,\mathsf{let}\ x\ \mathsf{be}\ C[_]\ \mathsf{in}\ v \mid \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ C[_] \mid \\
& O((\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})C[_], \ldots, (\vec{x};\vec{x})v; (\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})v)(\vec{v};\vec{v}) \\
& \mid O((\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})v; (\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})C[_], \ldots, (\vec{x};\vec{x})v)(\vec{v};\vec{v}) \\
& \mid O((\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})v; (\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})v)(v \ldots C[_] \ldots v; \vec{v}) \\
& \mid O((\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})v; (\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})v)(\vec{v}; v \ldots C[_] \ldots v)
\end{aligned}$$

Now define the *instantiation* of a pre-context $C[\_]$ by a pre-term v, written $C[v]$, as the pre-context $C[\_]$ but with the unique occurrence of $\_$ replaced by the pre-term v. It is easy to show that the instantiation of any pre-context by a pre-term is a pre-term.

Definition 4.4.1 (Term Context)
A $\Gamma; \Delta - A/\Gamma'; \Delta' - B$-*term context* of $\mathsf{Lin}(\mathbb{O})$ is a pre-context $C[\_]$ such that for any term $\Gamma; \Delta \vdash v : A$ of $\mathsf{Lin}(\mathbb{O})$, we have a derivation of the typing judgement $\Gamma'; \Delta' \vdash C[v] : B$.

As before, this definition of contexts is not inductive, but could be equivalently presented in an inductive form.

An *equality judgement* over the typing system $\mathsf{Lin}(\mathbb{O})$ has the form $\Gamma; \Delta \vdash v = w{:}A$, where v and w are both $\Gamma; \Delta$-terms of type A.

An *axiom set* is a set of typed equality judgements. We let $\mathbb{A}$ range over axiom sets. We now define intuitionistic terms:

Definition 4.4.2 (Intuitionistic Term)
A $\Gamma; \_$-term is *intuitionistic* if it is an instance of the following inductive definition:

$$v ::= x \mid O((\vec{x};\vec{x})v, \ldots, (\vec{x};\vec{x})v;)(;)$$

for some intuitionistic operator O.

We can now define the derivable equality judgements of the type-theory $\mathsf{Lin}(\mathbb{O})$ over the axiom set $\mathbb{A}$.

Definition 4.4.3 (The Equality of Lin($\mathbb{O}$) over $\mathbb{A}$)
An equality judgement $\Gamma; \Delta \vdash v = w{:}A$ is derivable if it is present in the axiom set $\mathbb{A}$ or is derivable using the following rules:

$$(Refl)\ \Gamma; \Delta \vdash v = v{:}A \qquad \frac{\Gamma; \Delta \vdash v = w{:}A \quad \Gamma; \Delta \vdash w = w'{:}A}{\Gamma; \Delta \vdash v = w'{:}A}\ (Trans)$$

$$\frac{\Gamma; \Delta \vdash v = w{:}A}{\Gamma; \Delta \vdash w = v{:}A}\ (Sym) \qquad \frac{\Gamma; \Delta \vdash v = w{:}A}{\Gamma'; \Delta' \vdash C[v] = C[w]{:}B}\ (Cong)$$

where $C[\_]$ is a $\Gamma;\Delta - A/\Gamma';\Delta' - B$ term-context.

$$(F-\beta_v) \qquad \Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w = w\{v/x\}{:}A$$
where $\Gamma;\_\vdash v{:}Q$ is an intuitionistic term

$$(F-\eta) \qquad \Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ x = v{:}Q$$

$$(cc-1) \qquad \Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ (\mathsf{let}\ y\ \mathsf{be}\ w\ \mathsf{in}\ w') = \mathsf{let}\ y\ \mathsf{be}\ (\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w)\ \mathsf{in}\ w'{:}A$$
where x is not free in w' and y is not free in v

$$(cc-2) \qquad \Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ (\mathsf{let}\ y\ \mathsf{be}\ w\ \mathsf{in}\ w') = \mathsf{let}\ y\ \mathsf{be}\ w\ \mathsf{in}\ (\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w'){:}A$$
where x is not free in w and y is not free in v

$$(cc-3) \qquad \Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ O((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(\vec{v}';w'_1\ldots w'_{s'}) =$$
$$O((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(\vec{v}';w'_1\ldots(\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w'_i)\ldots w'_{s'}){:}A$$
where x does not occur free other than possibly in w'_i

$$(cc-4) \qquad \Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ O((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(v'_1\ldots v'_{r'};\vec{w}') =$$
$$O((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(v'_1\ldots(\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ v'_i)\ldots v'_{r'};\vec{w}'){:}A$$
where x does not occur free other than possibly in v'_i

$$(cc-5) \qquad \Gamma;\Delta \vdash \mathsf{let}\ x'\ \mathsf{be}\ w\ \mathsf{in}\ O((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(\vec{w}';\vec{v}') =$$
$$O((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}_i;\vec{y}_i)(\mathsf{let}\ x\ \mathsf{be}\ w\ \mathsf{in}\ w_i),\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(\vec{w}';\vec{v}'){:}A$$
where x' does not occur free other than possibly in v_i

We refer to the type-theory with typing system $\mathsf{Lin}(\mathbb{O})$ over axiom set $\mathbb{A}$ as $\mathsf{Lin}(\mathbb{O},\mathbb{A})$.

In considering the axioms, the least intuitive are clearly the commuting conversions. These can be understood by noting that we allow the let -construct to commute into any linearly natural argument place of an operator, or into any of the terms which are incorporated for the free assumptions, as these are not bound by the operator.

Output Naturality

The notion of output naturality is less important than those of intuitionistic and linear naturality, but it is still worth defining as it will prove very convenient. In fact, every operator corresponding to an elimination rule of $\mathsf{DILL}(\mathbb{C})$ will be output-natural, and the equalities in the definition will be precisely the commuting conversions of these operators.

DEFINITION 4.4.4 (OUTPUT-PARAMETERISED FAMILY)
We say that a typing system $\mathsf{Lin}(\mathbb{O})$ has an *output-parameterised family of operators* O if O is a set of operators indexed by the set M_L, and the element of the

set indexed by the type (or proposition) C, written O_C, has arity;

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1,\ldots,(\vec{Q}_r;\vec{A}_r)(B_r);(\vec{Q}'_1;\vec{A}'_1)B'_1,\ldots,(\vec{Q}'_{s-1};\vec{A}'_{s-1})B'_{s-1},(\vec{Q}'_s;\vec{A}'_s)(C)}{(\vec{Q}'';\vec{A}'')(C)}$$

where position s is distinguished and $s \geq 1$. We will say that the output-parameterised family has arity:

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1,\ldots,(\vec{Q}_r;\vec{A}_r)(B_r);(\vec{Q}'_1;\vec{A}'_1)B'_1,\ldots,(\vec{Q}'_{s-1};\vec{A}'_{s-1})B'_{s-1},(\vec{Q}'_s;\vec{A}'_s)(\_)}{(\vec{Q}'';\vec{A}'')(\_)}$$

Definition 4.4.5 (Output Naturality)
Given a type-theory $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, an output-parameterised family O with arity:

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1,\ldots,(\vec{Q}_r;\vec{A}_r)(B_r);(\vec{Q}'_1;\vec{A}'_1)B'_1,\ldots,(\vec{Q}'_{s-1};\vec{A}'_{s-1})B'_{s-1},(\vec{Q}'_s;\vec{A}'_s)(\_)}{(\vec{Q}'';\vec{A}'')(\_)}$$

is *output natural* if assuming the following:

- that we have $\Gamma\vec{x}_i{:}\vec{Q}_i;\vec{y}_i{:}\vec{A}_i \vdash v_i{:}B_i$ for $i = 1\ldots r$
- that we have $\Gamma,\vec{y}'_j{:}\vec{Q}'_j;\Delta_j,\vec{x}_j{:}\vec{A}'_j \vdash w_j{:}B_j$ for $j = 1\ldots s$,
- that we have $\Gamma;\Delta',x''{:}B_s \vdash w{:}C$,
- that we have $\Gamma;\Delta''_{i'} \vdash w'_{i'}{:}Q''_{i'}$ for $i' = 1\ldots r'$, where $\vec{Q}'' = Q''_1\ldots Q''_{r'}$,
- and that we have $\Gamma;\Delta'''_{j'} \vdash v'_{j'}{:}A''_{j'}$ for $j' = 1\ldots s'$, where $\vec{A}'' = A''_1\ldots A''_{s'}$.

we have the following equality judgement:

$$\begin{aligned}\Gamma;\Delta \vdash & w\{\mathsf{O}_{B_s}((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(\vec{v}';\vec{w}')/x''\}\\ &= \mathsf{O}_C((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w\{w_s/x''\})(\vec{v}';\vec{w}'){:}C\end{aligned}$$

where $\Delta = \Delta_1\#\ldots\#\Delta_s\#\Delta'\#\Delta'_1\#\ldots\#\Delta'_{r'}\#\Delta''_1\#\ldots\#\Delta''_{s'}$. We let $\mathsf{ONat}(\mathbb{O},O)$ be the set of all such equalities for an operator O in the typing system $\mathsf{Lin}(\mathbb{O})$.

We note here that since any operator which is part of an output-parameterised set of operators in a given typing system $\mathsf{Lin}(\mathbb{O})$ has at least one linearly-natural argument (by definition), no such operator can be intuitionistic.

As an example of an output-natural operator, consider the set of operators given by the $\otimes - L$-rule of $\mathsf{DILL}(\mathbb{C})$. In this framework, it is an output-natural operator set of arity:

$$\frac{;(;AB)(\_)}{(;A\otimes B)(\_)}$$

and the equalities specified by the definition of output-naturality are just the commuting conversions of this term construct in $\mathsf{DILL}(\mathbb{C})$.

Chapter 5

The Semantics of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$

We will now give a semantics for the type theory $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$. Just as the underlying structural rules of the logic and type-theory capture the essence of the interaction between linear and intuitionistic behaviour, we expect the models of the underlying structural rules of the type theory to have just the essential structure required to model this interaction. In fact, the models we will use for $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ will be based on a cartesian category interpreting the intuitionistic types, a symmetric monoidal category interpreting the linear types, and a monoidal functor from the cartesian category to the symmetric monoidal category. Operators will then be modelled as natural transformations on hom-sets, and theories will be imposed as sets of equalities on arrows. Having given this framework, we then get soundness and completeness results for the models, and further we get an initiality result for an appropriate category of small models and morphisms.

5.1 The Interpretation

We refer once again to the primitive categorical definitions of appendix A.2, and their numbered equalities.

The *carrier* of a $\mathsf{Lin}(\mathbb{O})$-model is a triple $(\mathcal{C}, \mathcal{S}, F)$ such that $\mathcal{C}$ is a strict cartesian category, $\mathcal{S}$ is a strict symmetric monoidal category and $F : \mathcal{C} \to \mathcal{S}$ is a strict monoidal functor.

Definition 5.1.1 (Lin($\mathbb{O}$)-Interpretation)
An *interpretation* of the typing system $\mathsf{Lin}(\mathbb{O})$, which we write $\mathcal{G}$, is a carrier $(\mathcal{C}, \mathcal{S}, F)$ together with:

- primitive interpretation functions $[\![\_]\!]^{\mathcal{G}}_{\mathsf{M}_I} : \mathsf{M}_I \to \mathsf{obj}(\mathcal{C})$ and $[\![\_]\!]^{\mathcal{G}}_{\mathsf{M}_L} : \mathsf{M}_L \to \mathsf{obj}(\mathcal{S})$ such that for all $Q \in \mathsf{M}_I$ we have $[\![Q]\!]^{\mathcal{G}}_{\mathsf{M}_L} = F([\![Q]\!]^{\mathcal{G}}_{\mathsf{M}_I})$,

- for each operator $O \in \mathcal{O}_L$ having arity

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r;\vec{A}_r)B_r \quad ; \quad (\vec{Q}'_1;\vec{A}'_1)B'_1, \quad \ldots, \quad (\vec{Q}'_s;\vec{A}'_s)B'_s}{(\vec{Q}'';\vec{A}'')B''}$$

  a natural transformation

$$\begin{aligned}\llbracket O\rrbracket^{\mathcal{G}}_{\mathcal{O}_L} :& (\times_{i=1\ldots r}\mathcal{S}(F(_=)\otimes\llbracket\vec{Q}_i;\vec{A}_i\rrbracket^{\mathcal{G}},\llbracket B_i\rrbracket^{\mathcal{G}}_{\mathsf{M}_L}))\times \\ &(\times_{j=1\ldots s}\mathcal{S}(F(_=)\otimes\llbracket\vec{Q}'_j\rrbracket\otimes(_{-j})\otimes\llbracket\vec{A}'_j\rrbracket^{\mathcal{G}},\llbracket B'_j\rrbracket^{\mathcal{G}}_{\mathsf{M}_L})) \\ &\qquad\to\mathcal{S}(F(_=)\otimes\llbracket\vec{Q}''\rrbracket\otimes(\otimes_{j=1\ldots s}(_{-s}))\otimes\llbracket\vec{A}''\rrbracket^{\mathcal{G}},\llbracket B''\rrbracket^{\mathcal{G}}_{\mathsf{M}_L})\end{aligned}$$

  which is natural independently in each of the $s+1$ arguments $(_=)$ and $(_{-1}),\ldots,(_{-s})$, and where the interpretation $\llbracket\_\rrbracket^{\mathcal{G}}$ is extended to arbitrary contexts in the obvious way, as given shortly,

- for each operator $O \in \mathcal{O}_I$ having arity

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r;\vec{A}_r)B_r;}{(;)R}$$

  a natural transformation

$$\begin{aligned}\llbracket O\rrbracket^{\mathcal{G}}_{\mathcal{O}_I} : \times_{i=1\ldots r}\,\mathcal{S}(F(_=)\otimes\llbracket\vec{Q}_i;\vec{A}_i\rrbracket^{\mathcal{G}},\llbracket B_i\rrbracket^{\mathcal{G}}_{\mathsf{M}_L}) \\ \to\mathcal{C}(_=,\llbracket Q''\rrbracket^{\mathcal{G}}_{\mathcal{C}})\end{aligned}$$

  where the interpretation is again given shortly, such that for all objects X of $\mathcal{C}$ and all arrows $f_i : F(X)\otimes\llbracket\vec{Q}_i;\vec{A}_i\rrbracket\to\llbracket B_i\rrbracket$ where $i=1\ldots r$ in $\mathcal{S}$,

$$\llbracket O\rrbracket^{\mathcal{G}}_{\mathcal{O}_L}(X)(f_1,\ldots,f_r) = F(\llbracket O\rrbracket^{\mathcal{G}}_{\mathcal{O}_I}(X)(f_1,\ldots,f_r))$$

Where the particular interpretation we are referring to is clear, we will omit the superscript $\mathcal{G}$ on the interpretation function $\llbracket\_\rrbracket$.

We now proceed to extend the definition of the interpretation function $\llbracket\_\rrbracket^{\mathcal{G}}$ to terms for an arbitrary model $\mathcal{G}$ of $\mathsf{Lin}(\mathbb{O})$.

Definition 5.1.2 (The Interpretation on Contexts)
First, for a sequence of linear types $A_1\ldots A_r$, define $\llbracket A_1\ldots A_r\rrbracket^{\mathcal{G}} = A_1\otimes\ldots\otimes A_r$. Further, define $\llbracket Q_1\ldots Q_s\rrbracket^{\mathcal{G}}_{\mathcal{C}} = Q_1\times\ldots\times Q_s$ for a sequence of intuitionistic types $Q_1\ldots Q_s$. Now, for a pair of a sequence of intuitionistic types and a sequence of linear types $\vec{Q};\vec{A}$, define $\llbracket\vec{Q};\vec{A}\rrbracket^{\mathcal{G}} = F(\llbracket\vec{Q}\rrbracket^{\mathcal{G}}_{\mathcal{C}})\otimes\llbracket\vec{A}\rrbracket^{\mathcal{G}}$. Finally overload these notations to sequences of typings and typing contexts using the function $|\_|$; for example, for a generalised typing context $\Gamma;\Delta$ define $\llbracket\Gamma;\Delta\rrbracket^{\mathcal{G}} = \llbracket|\Gamma|;|\Delta|\rrbracket^{\mathcal{G}}$. Note that $\llbracket\Gamma;\Delta\rrbracket^{\mathcal{G}} = \llbracket\_;\Gamma,\Delta\rrbracket^{\mathcal{G}}$.

We now define some useful arrows:

$$\begin{aligned}
\mathsf{disc}'_{\vec{Q}} &: [\![\vec{Q}]\!] \to I \\
\mathsf{dup}'_{\vec{Q}} &: [\![\vec{Q}]\!] \to [\![\vec{Q}]\!] \otimes [\![\vec{Q}]\!] \\
\mathsf{merge}_{\Delta_1,\Delta_2,\Delta} &: [\![\Delta]\!] \to [\![\Delta_1]\!] \otimes [\![\Delta_2]\!] \\
\mathsf{mmerge}^r_{\Delta_1,\ldots,\Delta_r,\Delta} &: [\![\Delta]\!] \to [\![\Delta_1]\!] \otimes \ldots \otimes [\![\Delta_r]\!] \\
\mathsf{str}'_{\Gamma,\Delta_1,\Delta_2,\Delta} &: [\![\Gamma;\Delta]\!] \to [\![\Gamma;\Delta_1]\!] \otimes [\![\Gamma;\Delta_2]\!] \\
\mathsf{mstr}^r_{\Gamma,\Delta_1,\ldots,\Delta_r,\Delta} &: [\![\Gamma;\Delta]\!] \to [\![\Gamma;\Delta_1]\!] \otimes \ldots \otimes [\![\Gamma;\Delta_r]\!]
\end{aligned}$$

as follows:

$$\begin{aligned}
\mathsf{disc}'_{\vec{Q}} &= F(\mathsf{d}_{[\![\vec{Q}]\!]_C}) \\
\mathsf{dup}'_{\vec{Q}} &= F(\mathsf{c}_{[\![\vec{Q}]\!]_C}) \\
\mathsf{merge}_{\Delta_1,\Delta_2,\Delta} &= \begin{cases} \mathsf{id}_I & \text{where } \Delta = \Delta_1 = \Delta_2 = _ \\ \mathsf{id}_{[\![A]\!]} \otimes \mathsf{merge}_{\Delta'_1,\Delta_2,\Delta'} & \text{where } \Delta_1 = x{:}A, \Delta'_1 \text{ and } \Delta = x{:}A, \Delta' \\ \mathsf{id}_{[\![A]\!]} \otimes \mathsf{merge}_{\Delta_1,\Delta'_2,\Delta'}; & \\ \mathsf{s}_{[\![A]\!],[\![\Delta_1]\!]} \otimes \mathsf{id}_{[\![\Delta'_2]\!]} & \text{where } \Delta_2 = x{:}A, \Delta'_2 \text{ and } \Delta = x{:}A, \Delta' \end{cases} \\
\mathsf{mmerge}^r_{\Delta_1,\ldots,\Delta_r,\Delta} &= \begin{cases} \mathsf{id}_{[\![\Delta]\!]} & \text{if } r = 1 \\ \mathsf{merge}_{\Delta_1,(\Delta_2,\ldots,\Delta_{s+2})}; \mathsf{id}_{[\![\Delta_1]\!]} \otimes \mathsf{mmerge}^{s+1}_{\Delta_2,\ldots,\Delta_{s+2}} & \text{where } r = s+2 \end{cases} \\
\mathsf{str}'_{\Gamma,\Delta_1,\Delta_2,\Delta} &= \mathsf{dup}'_\Gamma \otimes \mathsf{merge}_{\Delta_1,\Delta_2,\Delta}; \mathsf{id}_{[\![\Gamma]\!]} \otimes \mathsf{s}_{[\![\Gamma]\!],[\![\Delta]\!]} \otimes \mathsf{id}_{[\![\Delta_2]\!]} \\
\mathsf{mstr}^r_{\Gamma,\Delta_1,\ldots,\Delta_r,\Delta} &= \begin{cases} \mathsf{id}_{[\![\Gamma;\Delta]\!]} & \text{if } r = 1 \\ \mathsf{str}_{\Gamma,\Delta_1,(\Delta_2,\ldots,\Delta_{s+2})}; \mathsf{id}_{[\![\Gamma;\Delta_1]\!]} \otimes \mathsf{mstr}^{s+1}_{\Gamma,\Delta_2,\ldots,\Delta_{s+2}} & \text{if } r = s+2 \end{cases}
\end{aligned}$$

Definition 5.1.3 (The Interpretation on Terms)
We will interpret a term $\Gamma; \Delta \vdash v{:}A$ by induction over the unique derivation of the typing judgement.

Intuitionistic Axiom In this case, we take

$$[\![\Gamma, x{:}Q, \Gamma'; \_ \vdash x{:}Q]\!] = \mathsf{disc}'_\Gamma \otimes \mathsf{id}_{[\![Q]\!]} \otimes \mathsf{disc}'_{\Gamma'}$$

Linear Axiom In this case, we take

$$[\![\Gamma; x{:}A \vdash x{:}A]\!] = \mathsf{disc}'_\Gamma \otimes \mathsf{id}_{[\![A]\!]}$$

The F Rule Assuming that $[\![\Gamma;\Delta_1 \vdash v{:}Q]\!] = f$ and $[\![\Gamma, x{:}Q; \Delta_2 \vdash w{:}A]\!] = g$, we take

$$[\![\Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w{:}A]\!] = \mathsf{str}'_{\Gamma,\Delta_1,\Delta_2,\Delta}; f \otimes \mathsf{id}_{[\![\Gamma;\Delta_2]\!]}; \mathsf{s}_{[\![\Gamma]\!],[\![Q]\!]} \otimes \mathsf{id}_{[\![\Delta_2]\!]}; g$$

The Operator Rule Given an operator O of arity:

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1, \quad \dots, \quad (\vec{Q}_r;\vec{A}_r)B_r \quad ; \quad (\vec{Q}'_1;\vec{A}'_1)B'_1, \quad \dots, \quad (\vec{Q}'_s;\vec{A}'_s)B'_s}{(\vec{Q}'';\vec{A}'')B''}$$

where $\vec{Q}' = Q'_1 \dots Q'_s$ and $\vec{A}' = A'_1 \dots A'_{s'}$, assume

- that $[\![\Gamma, \vec{x}_i{:}\vec{Q}_i; \vec{y}_i{:}\vec{A}_i \vdash v_i{:}B_i]\!] = f_i$ for $i = 1 \dots r$,
- that $[\![\Gamma; \vec{x}'_j{:}\vec{Q}'_j; \Delta_j, \vec{y}_j{:}{:}\vec{A}'_j \vdash w_j{:}B'_j]\!] = g_j$ for $j = 1 \dots s$,
- that $[\![\Gamma; \Delta'_{i'} \vdash v_{i'}{:}Q''_{i'}]\!] = f'_{i'}$ for $i' = 1 \dots r'$,
- that $[\![\Gamma; \Delta''_{j'} \vdash w'_{j'}{:}A''_{j'}]\!] = g'_{j'}$ for $j' = 1 \dots s'$

Now we take

$$\begin{gathered}[\![\Gamma; \Delta \vdash O((\vec{x}_1;\vec{y}_1)v_1 \dots (\vec{x}_r;\vec{y}_r)v_r; (\vec{x}'_1;\vec{y}'_1)w_1 \dots (\vec{x}'_s;\vec{y}'_s)w_s)(\vec{v}';\vec{w}'){:}B'']\!] = \\ \mathsf{mstr}^{r'+s+s'}_{\Gamma,\Delta'_1,\dots,\Delta'_{r'},\Delta_1,\dots,\Delta_s,\Delta''_1,\dots,\Delta''_{s'},\Delta}; (f'_1 \otimes \dots \otimes f'_{r'} \otimes \mathsf{id}_{[\![\Delta_1,\dots,\Delta_s]\!]} \otimes \text{‘}g'_1 \otimes \dots \otimes g'_{s'}); \\ [\![O]\!]_{\mathcal{O}_L}([\![\Gamma]\!]_{\mathcal{C}}, [\![\Delta_1]\!], \dots, [\![\Delta_s]\!])(f_1, \dots, f_r, g_1, \dots, g_s)\end{gathered}$$

We can also make an auxiliary definition:

Definition 5.1.4 (The Interpretation on Intuitionistic Terms)
Define the interpretation $[\![\_]\!]^{\mathcal{G}}_{\mathcal{C}}$ which takes intuitionistic terms $\Gamma; \_ \vdash v{:}A$ to arrows $[\![\Gamma; \_]\!]^{\mathcal{G}}_{\mathcal{C}} \to [\![A]\!]^{\mathcal{G}}_{\mathsf{M}_I}$ in the cartesian part of the model as follows:

Intuitionistic Axiom In this case, we take

$$[\![\Gamma, x{:}Q, \Gamma'; \_ \vdash x{:}Q]\!]_{\mathcal{C}} = \pi^{r+1+s}_{r+1}$$

where $\Gamma = y_1{:}R_1 \dots y_r{:}R_r$ and $\Gamma' = y'_1{:}R'_1 \dots y'_s{:}R'_s$.

The Operator Rule In this case, we take

$$[\![\Gamma; \_ \vdash O((\vec{x}_1;\vec{y}_1)v_1, \dots, (\vec{x}_r;\vec{y}_r)v_r;)(;){:}R]\!]_{\mathcal{C}} = [\![O]\!]_{\mathcal{C}}([\![\Gamma]\!]_{\mathcal{C}})(f_1, \dots, f_r)$$

where for $i = 1 \dots r$ we have $[\![\Gamma, \vec{x}_i{:}\vec{Q}_i; \vec{y}_i{:}\vec{A}_i \vdash v_i{:}B_i]\!] = f_i$ in $\mathcal{S}$.

We now define the models of the type theory $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$.

Definition 5.1.5 ($\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-Model)
A $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ model $\mathcal{G}$ is a $\mathsf{Lin}(\mathbb{O})$ interpretation such that: for each typed equality judgement $\Gamma; \Delta \vdash v = w{:}A$ in $\mathbb{A}$, we have $[\![\Gamma; \Delta \vdash v{:}A]\!]^{\mathcal{G}} = [\![\Gamma; \Delta \vdash w{:}A]\!]^{\mathcal{G}}$.

5.2 Soundness

We now show that the equality judgement of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ is soundly mapped into any model of the type theory $\mathsf{Lin}(\mathbb{O},\mathbb{A})$.

First we state two lemmas on the interpretations of the two admissible substitution rules.

Lemma 5.2.1 (Intuitionistic Substitution)
For any $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ model $\mathcal{G}$, given $[\![\Gamma,\Gamma';\_\vdash v{:}Q]\!]^{\mathcal{G}} = f$, where v is intuitionistic and $[\![\Gamma,x{:}Q,\Gamma';\Delta\vdash w{:}A]\!]^{\mathcal{G}} = g$, we have:

$$[\![\Gamma,\Gamma';\Delta\vdash w\{v/x\}{:}A]\!]^{\mathcal{G}} = \mathsf{str}'_{(\Gamma,\Gamma'),\Delta,\_,\Delta}; f\otimes \mathsf{id}_{[\![\Gamma,\Gamma';\Delta]\!]}; \mathsf{s}_{[\![Q]\!],[\![\Gamma]\!]}\otimes \mathsf{id}_{[\![\Gamma';\Delta]\!]}; g$$

Lemma 5.2.2 (Linear Substitution)
For any $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ model $\mathcal{G}$, given the interpretations $[\![\Gamma;\Delta_1\vdash v{:}A]\!]^{\mathcal{G}} = f$ and $[\![\Gamma;\Delta_2,x{:}A,\Delta_2'\vdash w{:}B]\!]^{\mathcal{G}} = g$, we have:

$$[\![\Gamma;\Delta\vdash w\{v/x\}{:}B]\!]^{\mathcal{G}} = \mathsf{str}'_{\Gamma,\Delta_1,(\Delta_2,\Delta_2'),\Delta}; f\otimes \mathsf{id}_{[\![\Gamma;\Delta_2,\Delta_2']\!]}; \mathsf{s}_{[\![A]\!],[\![\Gamma;\Delta_2]\!]}\otimes \mathsf{id}_{[\![\Delta_2']\!]}; g$$

where $\Delta = \Delta_1\#(\Delta_2,\Delta_2')$.

Theorem 5 (Soundness)
Given a $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ model $\mathcal{G}$ and a provable equality $\Gamma;\Delta\vdash v = w{:}A$ in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, we have that:

$$[\![\Gamma;\Delta\vdash v{:}A]\!]^{\mathcal{G}} = [\![\Gamma;\Delta\vdash w{:}A]\!]^{\mathcal{G}}$$

in the symmetric monoidal part of $\mathcal{G}$.

Proof First we observe that equality of arrows is an equivalence relation, and further that given a $\Gamma;\Delta - A/\Gamma';\Delta' - A'$ context $C[\_]$ and a term $\Gamma;\Delta\vdash t{:}A$, the interpretation of $\Gamma';\Delta'\vdash C[t]{:}A'$ in the internal language of the category contains as a sub-term the interpretation of $\Gamma;\Delta\vdash t{:}A$. This implies that our context-equality rule and equivalence rules are sound. Now by definition any typed equality in $\mathbb{A}$ is soundly interpreted in $\mathcal{G}$, and hence we need only to show that the 7 axiomatic equalities of the let construct are soundly mapped into $\mathcal{G}$. Here, we give as examples the proofs for the first two equalities.

- Considering the equality $\Gamma;\Delta\vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w = w\{v/x\}{:}A$, we have that:

$$\begin{aligned}&[\![\Gamma;\Delta\vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w{:}A]\!]\\ &= \mathsf{str}'_{\Gamma,\_,\Delta,\Delta}; (f\otimes \mathsf{id}_{[\![\Gamma;\Delta]\!]}); \mathsf{s}_{[\![|\Gamma|]\!],[\![Q]\!]_{\mathsf{M}_L}}\otimes \mathsf{id}_{[\![|\Delta|]\!]}; g\end{aligned}$$

  where $[\![\Gamma;\_\vdash v{:}Q]\!] = f$ and $[\![\Gamma,x{:}Q;\Delta\vdash w{:}A]\!] = g$, which is precisely $[\![\Gamma,y{:}Q,\Gamma';\Delta\vdash v\{y/x\}{:}A]\!]$ according to our lemma.

- Considering the equality $\Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ x = v{:}Q$, we have that:

$$\begin{aligned}
&[\![\Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ x{:}Q]\!]\\
&= \mathsf{str}'_{\Gamma,\Delta,_,\Delta}; f \otimes \mathsf{id}_{[\![\Gamma;\Delta]\!]}; \mathsf{s}_{[\![|\Gamma|]\!],[\![Q]\!]_{\mathsf{M}_L}} \otimes \mathsf{id}_{[\![|\Delta|]\!]}; (\mathsf{disc}_\Gamma \otimes \mathsf{id}_{[\![Q]\!]_{\mathsf{M}_L}})\\
&= \mathsf{str}'_{\Gamma,\Delta,_,\Delta}; (f \otimes \mathsf{disc}_\Gamma)\\
&= f
\end{aligned}$$

again as required.

Constants

We consider the interpretation of a constant in this semantics. By definition, a constant O of arity A in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ has an interpretation in a $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model $\mathcal{G}$ as follows:

$$[\![O]\!]^{\mathcal{G}}_{\mathcal{O}_L} : \{i\} \to \mathcal{S}(F(\_), [\![A]\!]^{\mathcal{G}}_{\mathsf{M}_L})$$

However, natural transformations of this form are isomorphic to arrows $\mathcal{S}(I, [\![A]\!]^{\mathcal{G}}_{\mathsf{M}_L})$, and so we will frequently refer to the interpretation of a constant as having this latter form.

We now want to show completeness of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ with respect to its models, which is to say that if two $\Gamma;\Delta$ terms of type A have the same interpretation in every $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ model, then they are provably equal in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$. This is a familiar result for type theories and their models, and it is commonly proved by the construction of a term model, in which the objects are types and the morphisms are terms. In such a term model, we use the structure of the type-theory to build the categorical structure required of the model. However, such constructions normally depend on the existence of the appropriate functors in the model as type constructors, for example as $\otimes$ occurs explicitly in $\mathsf{DILL}(\mathbb{C})$. Type-theories such as ours which have no type-constructors as basic must therefore be treated differently. The most one can normally hope for in these cases is that there exists a term category in which the objects and morphisms are freely constructed in a simple way from the types and terms of the theory; the best example of this is the construction of a term cartesian category from sequences of types and terms of the basic intuitionistic type-theory having only axiom and cut rules. Operators can be added in this example, and the result extended to show that given any operator *over the objects of the free strict cartesian category on the types of the theory*, there exists a set of operators such that the term model of the type theory over this set is isomorphic to the free strict cartesian category with the original operator.

This result tells us that working in the type theory without an explicit product constructor is the same as working in the type theory with that constructor added, even when we are allowed arbitrary operators over the types of the theory with explicit product constructors. As an example of the method, imagine a nullary operator (a constant) $\overline{(A)B \times C}^{\,c}$ in the theory with products. Clearly there is no one operator which represents this in the theory without explicit products, since $B \times C$ is not a type of that theory. However, we can represent the operator as a pair of operators c_1, c_2 having arities $\overline{(A)B}^{\,c_1}$ and $\overline{(A)C}^{\,c_2}$, both of which can exist in the theory without products, and prove that the resulting type theory is equivalent to the type theory with products over the original operator c.

Unfortunately, there is no result of this kind for SMC's and the basic linear type theory except for the case with no operators. The problem is that given for example a constant $\overline{(A)B \otimes C}^{\,c'}$ in the theory with tensor, there is no equivalent set of operators in the theory without tensors- it is not possible to represent this as a pair since linearity demands that B and C be produced and used together. This behaviour is inherited by our system, since it clearly incorporates the basic linear type theory but without an explicit tensor type constructor.

The implication of this discussion is that our type theory is only expressive enough to represent SMC's having a certain subclass of the obvious operators, ie those not mentioning the tensor or the unit in the outputs of either the arguments or the results of the operator.

5.3 The Term Model

It is normal to define the term model of a type-theory using categories having as objects and arrows elementary constructions over the types and terms of the type theory. In particular, it is normal in the case of a linear type-theory to construct a symmetric monoidal term category using sequences of types and terms of the type theory. However, in this case such a construction is not rich enough in arrows, as can be seen from the fact that any interpretation of $\mathsf{Lin}(\mathbb{O})$ has in its s.m.c. part an arrow $FX \to FX \otimes FX$ for any object X of $\mathcal{C}$. The natural next step is therefore to allow constructions of the form $\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ ...(\mathsf{let}\ y\ \mathsf{be}\ v'\ \mathsf{in}\ w_1 \dots w_r)$ which augment the sequence construction sufficiently to allow us to express the correct arrows. Unfortunately, once we have made this construction certain obvious equalities appear between constructed elements, notably the equalities which would be commuting conversions in a type-theoretic setting. Whilst the construction could be pushed through, it seems clearer to present a slightly amended type-theory which we can use directly to construct the term-model, and then to relate this extended type theory to the original type theory.

We will therefore define the term $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model by giving its objects and morphisms as the types and terms of this slightly extended type theory, and then show that this new type theory is a full and faithful extension of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$. Hence the completeness result follows.

Definition 5.3.1 (The Type Theory $\mathrm{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$)
For a signature $\mathbb{O} = (\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$ and an axiom set $\mathbb{A}$ we define the type theory $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ as $\mathsf{Lin}((\mathsf{M}_I^{\otimes}, \mathsf{M}_L^{\otimes}, \mathcal{O}_I^{\otimes}, \mathcal{O}_L^{\otimes}), \mathbb{A}^{\otimes})$, where we define $\mathsf{M}_I^{\otimes}$, $\mathsf{M}_L^{\otimes}$, $\mathcal{O}_I^{\otimes}$, $\mathcal{O}_L^{\otimes}$ and $\mathbb{A}^{\otimes}$ as follows:

- $\mathsf{M}_I^{\otimes}$ is the set of sequences of length one each containing an element of M_I, which we will refer to briefly as *singleton sequences*.
- $\mathsf{M}_L^{\otimes}$ is the set of sequences of elements of M_L. We will write a sequence $\vec{A}$ of elements of M_L as $\otimes\vec{A}$ and may write the empty sequence of such elements as I.

We will refer to typing contexts of $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ using the familiar $\Gamma;\Delta$, and we define a function to convert $\mathsf{Lin}(\mathbb{O})$ typing contexts to $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ typing contexts. On sequences of linear types $A_1 \dots A_r$, define $\overbrace{A_1 \dots A_r}$ as the sequence of singleton sequences containing A_1 to A_r respectively. We extend this function to sequences of typings and hence typing contexts by defining $\overbrace{x{:}A} = x{:}\overbrace{A}$. We also extend $\overbrace{\ }$ to pre-terms simply by applying it to each type annotation in the pre-term, but when we omit type annotations on pre-terms, we also omit $\overbrace{\ }$ on pre-terms.

Although in fact one might argue that $\overbrace{\Gamma;\Delta} = \Gamma;\Delta$, by identifying singletons and the single element they contain, we retain the syntax here to help make the distinction between the two type theories clear.

Now,

- $\mathcal{O}_L^{\otimes}$ is the set containing an operator O of arity

$$\frac{(\overbrace{\vec{Q}_1;\vec{A}_1})\,\overbrace{B_1},\quad \dots,\quad (\overbrace{\vec{Q}_r;\vec{A}_r})\,\overbrace{B_r}\quad;\quad (\overbrace{\vec{Q}'_1;\vec{A}'_1})\,\overbrace{B'_1},\quad \dots,\quad (\overbrace{\vec{Q}'_s;\vec{A}'_s})\,\overbrace{B'_s}}{(\overbrace{\vec{Q}'';\vec{A}''})\,\overbrace{A''}}$$

  for each operator $O \in \mathcal{O}_L$ having arity:

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1,\quad \dots,\quad (\vec{Q}_r;\vec{A}_r)B_r\quad;\quad (\vec{Q}'_1;\vec{A}'_1)B'_1,\quad \dots,\quad (\vec{Q}'_s;\vec{A}'_s)B'_s}{(\vec{Q}'';\vec{A}'')B''}$$

with the addition of a weak tensor operator $\otimes^r_{\vec{B}}$ of arity:

$$\frac{;\quad (;)B_1, \quad \ldots, \quad (;)B_r}{(;) \otimes (B_1 \ldots B_r)}$$

for each $r \neq 1$, such that each B_i is a singleton sequence.

- $\mathcal{O}^{\otimes}_I$ is the subset of $\mathcal{O}^{\otimes}_L$ containing those operators induced by operators in $\mathcal{O}_I \subseteq \mathcal{O}_L$.
- $\mathbb{A}^{\otimes}$ is the set containing $\widehat{\Gamma; \Delta} \vdash v = w \colon \widehat{A}$ for each equality $\Gamma; \Delta \vdash v = w \colon A$ in $\mathbb{A}$.

It is fairly obvious that given a term $\Gamma; \Delta \vdash v \colon A$ of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$, we have that $\widehat{\Gamma; \Delta} \vdash v \colon \widehat{A}$ in $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$. We will write $\otimes^r(;()v_1, \ldots, ()v_r)(;)$ as $\otimes(v_1, \ldots, v_r)$ for clarity, and we define the abbreviation $\otimes(v) = v$, noting that there is no instance of the operator when $r = 1$ by definition.

It is important to note that although we are now using arbitrary sequences of the types of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ for types, the operators of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$ have arities only involving singleton sequences, with the exception of the tensor operators. In particular, any set of operators which is output natural in $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ will not be output natural in $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$, since there can be no instance of the operator family for any case where the indexing type is not a singleton sequence. This is a consequence of our preceding discussion, where we remarked that only a certain subclass of the operators which could be given over the type set with explicit tensor can be given in $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$.

Now say that a pre-term of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$ is in canonical form if it is an instance of the following inductive definition, where the v are pre-terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$:

$$v_c ::= \otimes\vec{v} \mid \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ v_c$$

Now we can prove a crucial lemma:

Lemma 5.3.2
Given a typing judgement $\widehat{\Gamma; \Delta} \vdash v \colon \otimes\vec{A}$ of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$, v is identically in canonical form with the various pre-terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ in v being terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$. Note that the typing contexts $\Gamma'; \Delta'$ which the pre-terms are typable in are uniquely determined from the typing context $\Gamma; \Delta$ and the form of the pre-term.

This lemma bears some explanation before we prove it. The reason we have defined the tensor of the theory $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$ weakly is so that this lemma is provable. This lemma then shows that the only terms provably of tensor type in

$\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$ are simply built up from tensors of terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ with shared intuitionistic variables. Further, an important implication of the lemma is that in the case where $\otimes\vec{A}$ is a singleton B, v is identically a $\Gamma; \Delta$-term of type B *in* $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$.

Proof We prove this by induction over the unique derivation of the typing judgement $\overbrace{\Gamma; \Delta} \vdash v{:}\otimes(A_1 \dots A_s)$.

Intuitionistic Axiom In this case we must have a derivation $\overbrace{\Gamma', x'{:}Q'}; \_ \vdash x' : Q'$, where the sequence $\otimes(\vec{A})$ must be the singleton Q, and hence the term is in the correct form.

Linear Axiom In this case, we must have a derivation $\overbrace{\Gamma; x'{:}A'} \vdash x' : A'$ and hence again we have that the derivation is in the correct form.

F Rule In this case we must have a derivation:

$$\frac{\overbrace{\Gamma; \Delta''_1} \vdash w{:}Q' \quad \overbrace{\Gamma, x'{:}Q'; \Delta''_2} \vdash v{:}\otimes(A_1 \dots A_s)}{\overbrace{\Gamma; \Delta} \vdash \mathsf{let}\ x'\ \mathsf{be}\ w\ \mathsf{in}\ v{:}\otimes(B_1 \dots B_s)}$$

where $\Delta = \Delta''_1 \# \Delta''_2$. However, in this case, we know by induction that there exist terms *of* $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ $\vec{v}', \vec{v}''$ and variables $\vec{x}$ such that v is:

$$\mathsf{let}\ x_1\ \mathsf{be}\ v'_1\ \mathsf{in} \dots \mathsf{let}\ x_r\ \mathsf{be}\ v'_r\ \mathsf{in}\ \otimes\vec{v}''$$

But since w must be a $\Gamma; \Delta''_2$-term of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ by induction, we have that $\overbrace{\Gamma; \Delta} \vdash \mathsf{let}\ x'\ \mathsf{be}\ w\ \mathsf{in}\ v{:}\otimes(A_1 \dots A_s)$ has the correct form.

Operator Rule In this case we need to consider first the operators of $\mathcal{O}_L$. Since these are operators over the types M_L, it is impossible that one should exist with result having output of a tensor type. Hence they can be disregarded. Now considering an instance of the tensor rule, we have a derivation:

$$\frac{\overbrace{\Gamma; \Delta''_1} \vdash v_1{:}A_1, \quad \dots, \quad \overbrace{\Gamma; \Delta''_s} \vdash v_s{:}A_s}{\overbrace{\Gamma; \Delta} \vdash \otimes\vec{v}{:}\otimes(A_1 \dots A_s)}$$

where $\Delta = \Delta''_1 \# \dots \# \Delta''_s$ (and the A_s are singletons). However, by induction on the the $\overbrace{\Gamma; \Delta''_i} \vdash v_i{:}A_i$ for $i = 1 \dots s$, we know that these must be $\Gamma; \Delta''_i$-terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$, and hence $\overbrace{\Gamma; \Delta} \vdash \otimes\vec{v} : \otimes(A_1 \dots A_s)$ has the correct form.

□

We can now define the term model of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$, $\mathcal{G}_T(\mathbb{O}, \mathbb{A})$.

Definition 5.3.3 (The Linear Term Category)
We define the linear term category, $\mathcal{S}_T$, as follows:

- The objects of $\mathcal{S}_T$ are the sequences of linear types of $\mathsf{Lin}(\mathbb{O})$.

- $\mathcal{S}_T(\vec{A},\vec{B}) = \{[(\vec{x},v)^{\vec{A},\vec{B}}] | \_; \overbrace{\vec{x}:\vec{A}} \vdash v:\otimes(\vec{B})\}$, where we write $[(\vec{x},v)^{\vec{A},\vec{B}}]$ to denote the equivalence class of $(\vec{x},v)^{\vec{A},\vec{B}}$ under the equivalence $\equiv'$ defined by
$$(\vec{x},v)^{\vec{A},\vec{B}} \equiv' (\vec{y},w)^{\vec{A},\vec{B}} \text{ if } \_; \overbrace{\vec{x}:\vec{A}} \vdash v = w\{\vec{x}/\vec{y}\}:\otimes\vec{B}$$
As in previous definitions of this sort, we write $[\vec{x},v]^{\vec{A},\vec{B}}$ for $[(\vec{x},v)]^{\vec{A},\vec{B}}$, omit type information where possible and assume that when writing the equivalence classes of arrows $[\vec{x},v]$ and $[\vec{y},w]$, the variable sequences are chosen to make all the variables in the concatenation $\vec{x}\vec{y}$ distinct, unless explicitly identified.

- $\mathsf{id}_{\vec{A}} = [\vec{x},\otimes\vec{x}]$

- $[\vec{x},v];[\vec{y},w] =$
$$[\vec{x}, \mathsf{let}\ x_1\ \mathsf{be}\ v_1'\ \mathsf{in}\ \ldots \mathsf{let}\ v_r'\ \mathsf{be}\ v_r'\ \mathsf{in}\ (w\{\vec{v}''/\vec{y}\})]$$
where by our previous lemma v is $\mathsf{let}\ x_1'\ \mathsf{be}\ v_1'\ \mathsf{in}\ \ldots \mathsf{let}\ x_r'\ \mathsf{be}\ v_r'\ \mathsf{in}\ \otimes\vec{v}''$.

It is easy to show that these definitions give a category. We proceed to define a strict s.m. structure on $\mathcal{S}_T$. First define the tensor on objects, which are sequences of types, as concatenation. Then define $[\vec{x},v]\otimes[\vec{y},w]$ as

$$[\vec{x}\vec{y}, \mathsf{let}\ x_1'\ \mathsf{be}\ v_1'\ \mathsf{in}\ \ldots \mathsf{let}\ x_r'\ \mathsf{be}\ v_r'\ \mathsf{in}\ \mathsf{let}\ y_1'\ \mathsf{be}\ w_1'\ \mathsf{in}\ \ldots \mathsf{let}\ y_{r'}'\ \mathsf{be}\ w_{r'}'\ \mathsf{in}\ \otimes(\vec{v}''\vec{w}'')]$$

where again by our previous lemma v is $\mathsf{let}\ x_1'\ \mathsf{be}\ v_1'\ \mathsf{in}\ \ldots \mathsf{let}\ x_r'\ \mathsf{be}\ v_r'\ \mathsf{in}\ \otimes\vec{v}''$ and w is $\mathsf{let}\ y_1'\ \mathsf{be}\ w_1'\ \mathsf{in}\ \ldots \mathsf{let}\ y_{r'}'\ \mathsf{be}\ w_{s'}'\ \mathsf{in}\ \otimes\vec{w}''$.

In order to define the intuitionistic term category, we need an auxiliary definition.

Definition 5.3.4 (Intuitionistic Equality)
We say that two intuitionistic terms $\Gamma; \_ \vdash v:Q$ and $\Gamma; \_ \vdash w:Q$ are *intuitionistically equal*, which we write $\Gamma; \_ \vdash v =_I w:Q$, if they are identically equal, or if v is $O((\vec{x}_1;\vec{y}_1)v_1', \ldots, (\vec{x}_r;\vec{y}_r)v_r';)(;)$, w is $O((\vec{x}_1;\vec{y}_1)w_1', \ldots, (\vec{x}_r;\vec{y}_r)w_r';)(;)$ and

$$\Gamma, \vec{x}_i:\vec{Q}_i; \vec{y}_i:\vec{A}_i \vdash v_i' = w_i':B_i$$

for all $i = 1\ldots r$.

Definition 5.3.5 (The Intuitionistic Term Category)
We define the intuitionistic term category, $\mathcal{C}_T$, as follows:

- The objects of $\mathcal{C}_T$ are sequences of intuitionistic types of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$.
- The arrows are given:

$$\mathcal{C}_T(\vec{Q}, \vec{R}) = \{[(\vec{x}, \vec{v})^{\vec{Q},\vec{R}}] \mid \widehat{\vec{x}{:}\vec{Q}};\_ \vdash v_i{:}R_i \text{ (where the } v_i \text{ are intuitionistic for } i = 1 \ldots s)\}$$

  where $\vec{R} = R_1 \ldots R_s$ and we write $[(\vec{x}, \vec{v})^{\vec{Q},\vec{R}}]$ to denote the equivalence class of $(\vec{x}, \vec{v})^{\vec{Q},\vec{R}}$ under the equivalence $\equiv'_C$ defined by

$$(\vec{x}, \vec{v}) \equiv'_C (\vec{y}, \vec{w}) \text{ if } \vec{v} = v_1 \ldots v_r, \vec{w} = w_1 \ldots w_r \text{ and } \vec{x}{:}\vec{Q};\_ \vdash v_i =_I w_i\{\vec{y}/\vec{x}\}{:}R_i \text{ for } i = 1 \ldots r$$

  Again, we write $[\vec{x}, \vec{v}]^{\vec{Q},\vec{R}}$ for $[(\vec{x}, \vec{v})]^{\vec{Q},\vec{R}}$, omit type information where possible and assume that when writing the equivalence classes of arrows $[\vec{x}, \vec{v}]$ and $[\vec{y}, \vec{w}]$, the variable sequences are chosen to make all the variables in the concatenation $\vec{x}\vec{y}$ distinct, unless explicitly identified.

- $\mathsf{id}_{\vec{Q}} = [\vec{x}, \vec{x}]$
- $[\vec{x}, \vec{v}]; [\vec{y}, \vec{w}] = [\vec{x}, \vec{w}\{\vec{v}/\vec{y}\}]$

We can equip $\mathcal{C}_T$ with a cartesian structure by defining the product to be concatenation on sequences, and giving the arrow structure as follows:

$$\langle [\vec{x}, \vec{v}], [\vec{x}, \vec{w}] \rangle = [\vec{x}, \vec{v}\vec{w}]$$
$$\mathsf{p}_i = [\vec{x}, x_i]$$

This construction is familiar, as it is almost identical to that which we used for the intuitionistic part of the term model of $\mathsf{DILL}(\mathbb{C})$. Similar proofs show that these definitions make $\mathcal{C}_T$ into a cartesian category.

We can now define the functor $F_T : \mathcal{C}_T \to \mathcal{S}_T$ as the identity on objects of $\mathcal{C}_T$, and on arrows by:

$$F([\vec{x}, \vec{v}]) = [\vec{y}, \mathsf{let}\ \vec{x}\ \mathsf{be}\ \vec{y}\ \mathsf{in}\ \otimes\vec{v}]$$

We can easily check that this definition is functorial, and further that F is a strict monoidal functor. We now define the term model $\mathcal{G}_T(\mathbb{O}, \mathbb{A})$ of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$.

Definition 5.3.6 (The Term Lin$(\mathbb{O}, \mathbb{A})$-Model)
The carrier of the term model $\mathcal{G}_T(\mathbb{O}, \mathbb{A})$ is the triple $(\mathcal{C}_T, \mathcal{S}_T, F_T)$, and the primitive interpretation functions are given as follows:

- $[\![\_]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}_{\mathsf{M}_I}$ takes the intuitionistic types to the singleton sequences containing them in $\mathcal{C}_T$.

- $[\![\_]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}_{\mathsf{M}_L}$ takes the linear types to the singleton sequences containing them in $\mathcal{S}_T$. Clearly for all $Q \in \mathsf{M}_I$ we have $[\![Q]\!]^{\mathcal{G}_T}_{\mathsf{M}_L} = F([\![Q]\!]^{\mathcal{G}_T}_{\mathsf{M}_I})$.

- Given an operator $O \in \mathcal{O}_I$ having arity
$$\frac{(\vec{Q}_1;\vec{A}_1)B_1, \quad \dots, \quad (\vec{Q}_r;\vec{A}_r)B_r;}{(;)R}$$
define the natural transformation $[\![O]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}_{\mathcal{O}_I}$ at $\vec{Q}'$ on arrows $[\vec{x}'\vec{x}_i\vec{y}_i, v_i]$ for $i = 1 \dots r$ to be the arrow
$$[(\vec{x}', O((\vec{x}_1;\vec{y}_1)v_1, \dots, (\vec{x}_r;\vec{y}_r)v_r;)()]$$

- Given an operator $O \in \mathcal{O}_L$ having arity
$$\frac{(\vec{Q}_1;\vec{A}_1)B_1, \quad \dots, \quad (\vec{Q}_r;\vec{A}_r)B_r \quad ; \quad (\vec{Q}'_1;\vec{A}'_1)B'_1, \quad \dots, \quad (\vec{Q}'_s;\vec{A}'_s)B'_s}{(\vec{Q}'';\vec{A}'')B''}$$
define the natural transformation $[\![O]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}_{\mathcal{O}_L}$ to be that which when applied at $\vec{R}, \vec{C}_1 \dots \vec{C}_s$ to the arrows $[\vec{z}\vec{x}_i\vec{y}_i, v_i]$ for $i = 1 \dots r$ and $[\vec{z}\vec{x}'_j\vec{z}_j\vec{y}'_j, w_j]$ for $j = 1 \dots s$ gives the arrow
$$[\vec{z}\vec{x}''\vec{z}'_1 \dots \vec{z}'_s\vec{y}'', O((\vec{x}_1;\vec{y}_1)(v_1), \dots, (\vec{x}_r;\vec{y}_r)(v_r); (\vec{x}'_1;\vec{y}'_1)(w_1), \dots, (\vec{x}'_s;\vec{y}'_s)(w_s))(\vec{x}'';\vec{y}'')]$$
Clearly these two definitions satisfy the condition on the interpretation of operators in the two categories.

In order to show that this is indeed a model, we need to observe that the conditions on $\mathsf{Lin}(\mathbb{O})$-interpretations and $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-models hold. This is easily done once we have proved the following lemma.

Lemma 5.3.7
Given a term $\vec{x}{:}\vec{Q};\vec{y}{:}\vec{A} \vdash v{:}B$ of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ such that $[\![\vec{x}{:}\vec{Q};\vec{y}{:}\vec{A} \vdash v{:}B]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})} = [\vec{x}\vec{y}, w]^{\vec{Q}\vec{A},B}$, we can derive the equality judgement in $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$:
$$\overbrace{\vec{x}{:}\vec{Q};\vec{y}{:}\vec{A}} \vdash w = v{:}B$$

5.4 Completeness

We now present a lemma relating $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ to $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$.

Lemma 5.4.1
Given an equality judgement $\overbrace{\Gamma;\Delta} \vdash v = w : \overbrace{A}$ in $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$, we have an equality judgement $\Gamma;\Delta \vdash v = w{:}A$ in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$.

Proof Firstly, we know that both v and w are $\Gamma;\Delta$ terms of type A in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ by the lemma 5.3.2. We prove the lemma by induction over the structure of the proof of the equality judgement $\widehat{\Gamma;\Delta} \vdash v = w\colon \widehat{A}$ in $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$. If the last rule was any of the reflexivity, symmetry or transitivity rules then by the induction hypothesis the proof is easy. If the last rule was the term-context rule then it is easy to see that the $\widehat{\Gamma;\Delta} - \widehat{A} \,/\, \widehat{\Gamma';\Delta'} - \widehat{B}$ term context of $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ used in the rule is also a $\Gamma;\Delta - A/\Gamma'\Delta' - B$ term context of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, and hence that the result follows from the induction hypothesis. Considering the axiomatic equalities, those generated from equalities in $\mathbb{A}$ are clearly the images of equalities in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, the naturality equations clearly must mention the operator $\otimes$ and hence are not terms in the appropriate context (since if they were they would be pre-terms of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, which is a contradiction) and finally, the axiomatic equalities which are commuting conversions for the let -construct $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ are the images of instances of the same equalities in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$. $\square$

This shows that the embedding from $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ into $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ is faithful. We also know that it is full by virtue of lemma 5.3.2.

We can use these results to relate the term model and $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ as follows:

LEMMA 5.4.2
The arrows $\vec{A} \to B$ in the symmetric monoidal category of the term model, $\mathcal{S}_T$, are isomorphic to the terms $\_;\vec{x}\colon\vec{A} \vdash v\colon B$ of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ quotiented by provable equality.

Proof Firstly consider arrows $\vec{A} \to B$ in $\mathcal{S}_T$. These are equivalence classes of terms of $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ $\_;\widehat{\vec{x}\colon\vec{A}} \vdash v\colon B$ quotiented by term equality and α-conversion on free variables. But we know that such terms of $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ are isomorphic to terms of $\mathsf{Lin}(\mathbb{O})$ $\_;\vec{x}\colon\vec{A} \vdash w\colon B$ by our lemmas relating the two type theories, and hence the result follows. $\square$

Now we can prove soundness and completeness:

THEOREM 6 (SOUNDNESS AND COMPLETENESS)
Given $\Gamma;\Delta$ terms v and w of type A in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, $\Gamma;\Delta \vdash v = w\colon A$ if and only if $[\![\Gamma;\Delta \vdash v\colon A]\!]^{\mathcal{G}} = [\![\Gamma;\Delta \vdash w\colon A]\!]^{\mathcal{G}}$ for all $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-models $\mathcal{G}$.

Proof This proof is largely by standard means. We already have the forward direction thanks to our proof of soundness. To show the other direction, assume that:

$$[\![\Gamma;\Delta \vdash v\colon A]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})} = [\vec{x}\vec{y}, v'] = [\vec{x}\vec{y}, w'] = [\![\Gamma;\Delta \vdash w\colon A]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}$$

where $\Gamma = \vec{x} : \vec{Q}$ and $\Delta = \vec{y} : \vec{B}$, for the particular $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model $\mathcal{G}_T(\mathbb{O}, \mathbb{A})$. Now by lemma 5.3.7 and the definition of equality in $\mathcal{G}_T(\mathbb{O}, \mathbb{A})$ we have that:

$$\widehat{\Gamma; \Delta} \vdash v = v' : \widehat{A} \quad \text{and} \quad \widehat{\Gamma; \Delta} \vdash w = w' : \widehat{A}$$

and $\widehat{\_; \Gamma, \Delta} \vdash v' = w' : \widehat{A}$.

From this last equality we can construct an equality judgement $\widehat{\Gamma; \Delta} \vdash v' = w' : \widehat{A}$ using the $\widehat{\Gamma; \Delta} - \widehat{A} \,/\, \widehat{\_; \Gamma, \Delta} - \widehat{A}$ term context $\_$, and hence by transitivity we have the equality judgement $\widehat{\Gamma; \Delta} \vdash v = w : \widehat{A}$. But now using our last lemma, we have the required equality judgement of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$:

$$\Gamma; \Delta \vdash v = w : A$$

□

5.5 Initiality

We now define suitable $\mathsf{Lin}(\mathbb{O})$-maps and prove that the term model $\mathcal{G}_T(\mathbb{O}, \mathbb{A})$ is (strictly) initial in the category of small $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-categories and $\mathsf{Lin}(\mathbb{O})$-maps.

Definition 5.5.1 (Lin($\mathbb{O}$)-Maps)
Define a $\mathsf{Lin}(\mathbb{O})$-map $\mathcal{F} : \mathcal{G} \to \mathcal{G}'$ for $\mathsf{Lin}(\mathbb{O})$-interpretations $\mathcal{G}$ and $\mathcal{G}'$ having carriers $(\mathcal{C}, \mathcal{S}, F)$ and $(\mathcal{C}', \mathcal{S}', F')$ respectively as a pair of functors $(\mathcal{F}_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}', \mathcal{F}_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}')$ such that:

- $\mathcal{F}_{\mathcal{C}}$ is strict cartesian and $\mathcal{F}_{\mathcal{S}}$ is strict symmetric monoidal,
- the following diagram commutes:

$$\begin{tikzcd} \mathcal{C} \arrow[r, "F"] \arrow[d, "\mathcal{F}_{\mathcal{C}}"'] & \mathcal{S} \arrow[d, "\mathcal{F}_{\mathcal{S}}"] \\ \mathcal{C}' \arrow[r, "F'"] & \mathcal{S}' \end{tikzcd}$$

- $\mathcal{F}_{\mathcal{C}}([\![\_]\!]^{\mathcal{G}}_{\mathsf{M}_I}) = [\![\_]\!]^{\mathcal{G}'}_{\mathsf{M}_I} : \mathsf{M}_I \to \mathsf{obj}(\mathcal{C}')$,
- $\mathcal{F}_{\mathcal{S}}([\![\_]\!]^{\mathcal{G}}_{\mathsf{M}_L}) = [\![\_]\!]^{\mathcal{G}'}_{\mathsf{M}_L} : \mathsf{M}_L \to \mathsf{obj}(\mathcal{S}')$,
- for each operator $O \in \mathcal{O}_I$ with arity

$$\frac{(\vec{Q}_1; \vec{A}_1)B_1, \quad \dots, \quad (\vec{Q}_r; \vec{A}_r)B_r;}{(;)R}$$

object X of $\mathcal{C}$ and arrows $f_i : F(X) \otimes [\![\vec{Q}_i; \vec{A}_i]\!] \to [\![B_i]\!]$ for $i = 1 \ldots r$ of $\mathcal{S}$,

$$\mathcal{F}_{\mathcal{C}}([\![O]\!]^{\mathcal{G}}_{\mathcal{O}_I}(X)(f_1 \ldots f_r)) = [\![O]\!]^{\mathcal{G}'}_{\mathcal{O}_I}(\mathcal{F}_{\mathcal{C}}(X)(\mathcal{F}_{\mathcal{S}}(f_1), \ldots, \mathcal{F}_{\mathcal{S}}(f_r)))$$

- for each $O \in \mathcal{O}_L$, object X of $\mathcal{C}$, objects $Y_1 \ldots Y_s$ and arrows of $\mathcal{S}$ $f_i : F(X) \otimes [\![\vec{Q}_i; \vec{A}_i]\!] \to [\![B_i]\!]$ for $i = 1 \ldots r$ and $g_j : F(X) \otimes [\![\vec{Q}'_j]\!] \otimes Y_j \otimes [\![\vec{A}'_j]\!] \to [\![B'_j]\!]$ for $j = 1 \ldots s$,

  $$\begin{aligned}&\mathcal{F}_{\mathcal{S}}([\![O]\!]^{\mathcal{G}}_{\mathcal{O}_L}(X, Y_1, \ldots, Y_s)(f_1 \ldots f_r, g_1 \ldots g_s)) = \\ &\quad [\![O]\!]^{\mathcal{G}'}_{\mathcal{O}_L}(\mathcal{F}_{\mathcal{C}}(X), \mathcal{F}_{\mathcal{S}}(Y_1), \ldots, \mathcal{F}_{\mathcal{S}}(Y_s))(\mathcal{F}_{\mathcal{S}}(f_1), \ldots, \mathcal{F}_{\mathcal{S}}(f_r), \mathcal{F}_{\mathcal{S}}(g_1), \ldots, \mathcal{F}_{\mathcal{S}}(g_s))\end{aligned}$$

Now define the category of small $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-models, $\mathsf{Cat}_{\mathsf{Lin}}(\mathbb{O}, \mathbb{A})$, to be the category having objects the small $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-models and morphisms the $\mathsf{Lin}(\mathbb{O})$-morphisms between them.

Now we will prove that each $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model canonically yields a $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$-model, simply by interpreting sequences of types as the tensor of the interpretation and interpreting the tensor operator on a sequence of terms as the obvious tensor of the interpretations of the terms, with the intuitionistic context duplicated.

Lemma 5.5.2

Given a $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model $\mathcal{G}$, we can construct a $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$-model $\mathcal{G}'$ as follows:

- The carrier of $\mathcal{G}'$ will be the carrier of $\mathcal{G}$, $(\mathcal{C}, \mathcal{S}, F)$.

- The primitive interpretation function $[\![\_]\!]^{\mathcal{G}'}_{\mathsf{M}_I^{\otimes}}$, which takes singleton sequences of elements of M_I to objects of $\mathcal{C}$, is just the function which takes the singleton sequence containing Q to $[\![Q]\!]^{\mathcal{G}}_{\mathsf{M}_I}$.

- The primitive interpretation function $[\![\_]\!]^{\mathcal{G}'}_{\mathsf{M}_L^{\otimes}}$, which takes sequences of elements of M_L to objects of $\mathcal{S}$, is just the function which takes the sequence $A_1 \ldots A_r$ to $[\![A_1]\!]^{\mathcal{G}}_{\mathsf{M}_L} \otimes \ldots \otimes [\![A_r]\!]^{\mathcal{G}}_{\mathsf{M}_L}$. These two definitions clearly satisfy the equality $[\![\_]\!]^{\mathcal{G}'}_{\mathsf{M}_L^{\otimes}} = F([\![\_]\!]^{\mathcal{G}'}_{\mathsf{M}_I^{\otimes}})$ over singleton sequences of intuitionistic types.

- Since $[\![\widehat{\Gamma; \Delta}]\!]^{\mathcal{G}'} = [\![\Gamma; \Delta]\!]^{\mathcal{G}}$ on objects of $\mathcal{S}$, as is easily seen from the definitions, the natural transformation $[\![O]\!]^{\mathcal{G}'}_{\mathcal{O}_L^{\otimes}}$ is just the natural transformation $[\![O]\!]^{\mathcal{G}}_{\mathcal{O}_L}$ on operators of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$, and similarly for $\mathcal{O}_I$. Clearly this definition satisfies the condition on these interpretation functions.

- For the tensor operator of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$,

  $$\frac{;(;)A_1, \quad \ldots, \quad (;)A_r}{(;) \otimes (A_1 \ldots A_r)} \otimes^r_{\Gamma, \Delta_1, \ldots \Delta_r, \vec{A}}$$

say $[\![\otimes^r]\!](X, Y_1, \ldots, Y_r)^{\mathcal{G}'}_{\mathcal{O}^{\otimes}_L}$ is the function which given $f_i : F(X) \otimes Y_i \to [\![\vec{A}]\!]$ for $i = 1 \ldots r$, takes value:

$$\mathsf{n}^r_{X,Y_1,\ldots,Y_r}; (f_1 \otimes \ldots \otimes f_r)$$

where $\mathsf{n}^r_{X,Y_1,\ldots,Y_r}$ is the obvious map taking

$$F(X) \otimes A_1 \otimes \ldots \otimes A_r \to \otimes(F(X) \otimes A_1), \ldots, (F(X) \otimes A_r)$$

It can now easily be shown that $\mathcal{G}'$ is a $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$-model, and further that on terms of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$ $\widehat{\Gamma; \Delta} \vdash v : A$ which are the images of terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ $\Gamma; \Delta \vdash v : A$, we have:

$$[\![\widehat{\Gamma; \Delta} \vdash v : A]\!]^{\mathcal{G}'} = [\![\Gamma; \Delta \vdash v : A]\!]^{\mathcal{G}'}$$

in the s.m. category $\mathcal{S}$.

Definition 5.5.3 (Canonical Morphism)
Given a $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model $\mathcal{G}$, define the morphism $\mathcal{FT}^{\mathcal{G}} : \mathcal{G}_T(\mathbb{O}, \mathbb{A}) \to \mathcal{G}$ as follows, where $\mathcal{FT}^{\mathcal{G}} = (\mathcal{FT}^{\mathcal{G}}_{\mathcal{C}}, \mathcal{FT}^{\mathcal{G}}_{\mathcal{S}})$:

- On objects of $\mathcal{C}_T$, which are sequences of intuitionistic types of $\mathsf{Lin}(\mathbb{O})$, $\vec{Q}$, define $\mathcal{FT}_{\mathcal{C}}(\vec{Q}) = [\![\vec{Q}]\!]^{\mathcal{G}}_{\mathcal{C}}$.

- On arrows of $\mathcal{C}_T$, which are equivalence classes $[\vec{x}, \vec{v}]^{\vec{Q},\vec{R}}$ for intuitionistic terms v of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$, define $\mathcal{FT}_{\mathcal{C}}([\vec{x}, \vec{v}])$ as $\langle f_1, \ldots, f_r \rangle$, where $f_i = [\![\_; \widehat{\vec{x}:\vec{A}} \vdash v_i : \otimes\vec{B}]\!]^{\mathcal{G}'}_{\mathcal{C}}$ for $i = 1 \ldots r$, where $\mathcal{G}'$ is the model of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$ which is generated from the $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model $\mathcal{G}$. Note that since the models have the same carrier, the interpretation takes arrows of $\mathcal{C}_T$ to arrows in the cartesian part of $\mathcal{G}$ as required.

- On objects of $\mathcal{S}_T$, which are sequences of linear types of $\mathsf{Lin}(\mathbb{O})$, $\vec{A}$, define $\mathcal{FT}_{\mathcal{S}}(\vec{A}) = [\![\vec{A}]\!]^{\mathcal{G}}$.

- On arrows of $\mathcal{S}_T$, which are equivalence classes $[\vec{x}, v]^{\vec{A},\vec{B}}$ for terms v of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$, define $\mathcal{FT}_{\mathcal{S}}([\vec{x}, v])$ as $[\![\_; \widehat{\vec{x}:\vec{A}} \vdash v : \otimes\vec{B}]\!]^{\mathcal{G}'}$ where $\mathcal{G}'$ is the model of $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$ which is generated from the $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model $\mathcal{G}$. Note that since the models have the same carrier, the interpretation takes arrows of $\mathcal{S}_T$ to arrows in the s.m.c. part of $\mathcal{G}$ as required.

We now prove two auxiliary lemmas:

Lemma 5.5.4

Given a term $\overbrace{\vec{y}{:}\vec{Q};\vec{x}{:}\vec{A}} \vdash v:\otimes\vec{B}$ of $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$, we have that for an arbitrary $\mathsf{Lin}(\mathbb{O})$-morphism $(\mathcal{F}_{\mathcal{C}},\mathcal{F}_{\mathcal{S}}) : \mathcal{G}_T(\mathbb{O},\mathbb{A}) \to \mathcal{G}'$:

$$\mathcal{F}_{\mathcal{S}}([\vec{y}'\vec{x},\mathsf{let}\ \vec{y}\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ v]) = [\![\overbrace{\vec{y}{:}\vec{Q};\vec{x}{:}\vec{A}} \vdash v{:}\otimes\vec{B}]\!]^{\mathcal{G}'}$$

Proof The proof proceeds by induction over the derivation $\overbrace{\vec{y}{:}\vec{Q};\vec{x}{:}\vec{A}} \vdash v{:}\otimes\vec{B}$.

- In the case of an intuitionistic axiom instance $\overbrace{\vec{x}{:}\vec{Q},y{:}R,\vec{x}'{:}\vec{Q}'};\_ \vdash y{:}R$, we have that:

  $$[\vec{x}''y'\vec{x}''',\mathsf{let}\ \vec{x}y\vec{x}'\ \mathsf{be}\ \vec{x}''y'\vec{x}'''\ \mathsf{in}\ y] = F_T(\mathsf{d}_{[\![\vec{Q}]\!]_{\mathcal{C}}^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}} \times \mathsf{id}_{[\![R]\!]_{\mathcal{C}}^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}} \times \mathsf{d}_{[\![\vec{Q}']\!]_{\mathcal{C}}^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}})$$

  But all of these constructs are preserved by the fact that $F_T;\mathcal{F}_{\mathcal{S}} = \mathcal{F}_{\mathcal{C}};F$ and that $\mathcal{F}_{\mathcal{C}}$ strictly preserves the cartesian structure, and so we have:

  $$\mathcal{F}_{\mathcal{S}}([\vec{x}''y'\vec{x}''',\mathsf{let}\ \vec{x}y\vec{x}'\ \mathsf{be}\ \vec{x}''y'\vec{x}'''\ \mathsf{in}\ y]) = F(\mathsf{d}_{[\![\vec{Q}]\!]_{\mathcal{C}}^{\mathcal{G}}} \times \mathsf{id}_{[\![R]\!]_{\mathcal{C}}^{\mathcal{G}}} \times \mathsf{d}_{[\![\vec{Q}']\!]_{\mathcal{C}}^{\mathcal{G}}})$$

  as required.

- In the case of a linear axiom instance $\overbrace{\vec{y}{:}\vec{Q};x{:}A} \vdash x{:}A$ we have that:

  $$[\vec{y}'x,\mathsf{let}\ \vec{y}\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ x] = F_T(\mathsf{d}_{[\![\vec{Q}]\!]_{\mathcal{C}}^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}}) \otimes \mathsf{id}_{[\![A]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}}$$

  But again, these constructs are preserved since $\mathcal{F}$ preserves F and the monoidal structure, and so we have:

  $$\mathcal{F}_{\mathcal{S}}([\vec{y}'x,\mathsf{let}\ \vec{y}\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ x]) = F(\mathsf{d}_{[\![\vec{Q}]\!]_{\mathcal{C}}^{\mathcal{G}}}) \otimes \mathsf{id}_{[\![A]\!]^{\mathcal{G}}}$$

- In the case of an instance of the F-rule:

  $$\frac{\overbrace{\vec{y}{:}\vec{Q};\vec{x}'{:}\vec{A}'} \vdash v{:}Q' \quad \overbrace{\vec{y}{:}\vec{Q},y'{:}Q';\vec{X}''{:}\vec{A}''} \vdash w{:}\otimes\vec{B}}{\overbrace{\vec{y}{:}\vec{Q};\vec{x}{:}\vec{A}} \vdash \mathsf{let}\ y'\ \mathsf{be}\ v\ \mathsf{in}\ w{:}\otimes\vec{B}}\ (F)$$

  where $\vec{x}{:}\vec{A} = (\vec{x}'{:}\vec{A}')\#(\vec{x}''{:}\vec{A}'')$, we have that

  $$\begin{aligned}[\vec{y}''\vec{x},\mathsf{let}\ y'\ \mathsf{be}\ v\ \mathsf{in}\ w] = {} & \mathsf{str}'_{\vec{Q},\vec{A}',\vec{A}'',\vec{A}};([\vec{y}''\vec{x}',\mathsf{let}\ \vec{y}\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ v] \otimes \mathsf{id}_{[\![\vec{Q};\vec{A}'']\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}}); \\ & (\mathsf{s}_{[\![\vec{Q}]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})},[\![Q']\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}} \otimes \mathsf{id}_{[\![\vec{A}'']\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}});[\vec{y}''y''',\mathsf{let}\ \vec{y}y'\ \mathsf{be}\ \vec{y}''y'''\ \mathsf{in}\ w]\end{aligned}$$

  In this expression, the two arrows still in the form $[\ldots]$ are preserved by induction since they annotate the premises of the F-rule, and the rest of the structure excepting the str' construct is preserved by the fact that $\mathcal{F}_{\mathcal{S}}$ is strict symmetric monoidal. It is fairly easy to show that the str' construct is also preserved by considering its construction.

- In the case of an operator rule instance not due to the tensor operator, the proof proceeds similarly, as the operator clause must be preserved.

- In the case where we have an instance of the tensor introduction rule,

$$\frac{\overbrace{\vec{y}{:}\vec{Q};\vec{x}_1{:}\vec{A}_1} \vdash v_1{:}B_1, \quad \ldots, \quad \overbrace{\vec{y}{:}\vec{Q};\vec{x}_r{:}\vec{A}_r} \vdash v_r{:}B_r}{\overbrace{\vec{y}{:}\vec{Q};\vec{x}{:}\vec{A}} \vdash \otimes^r(v_1 \ldots v_r){:}\otimes(B_1 \ldots B_r)} \otimes^r$$

  where $\vec{x}{:}\vec{A} = (\vec{x}_1{:}\vec{A}_1)\# \ldots \#(\vec{x}_r{:}\vec{A}_r)$, we have that:

$$[\vec{y}'\vec{x}, \mathsf{let}\ \vec{y}\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ \otimes^r(v_1 \ldots v_r)] =$$
$$\mathsf{mstr}^r_{\vec{Q},\vec{A}_1\ldots\vec{A}_r,\vec{A}}; ([\vec{y}'\vec{x}_1, \mathsf{let}\ \vec{y}\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ v_1] \otimes \ldots \otimes [\vec{y}'\vec{x}_r, \mathsf{let}\ \vec{y}\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ v_r])$$

  But now since we can show that mstr^r is preserved by $\mathcal{F}_\mathcal{S}$, and we know that the tensor must be, we have the required result. □

Similarly, we can show:

Lemma 5.5.5
Given an arbitrary intuitionistic term $\overbrace{\vec{x}{:}\vec{Q};\_} \vdash v : R$ of $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$, (which also must be an intuitionistic term $\vec{x}:\vec{Q};\_ \vdash v:R$ of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$) we have that for an arbitrary $\mathsf{Lin}(\mathbb{O})$-morphism $(\mathcal{F}_\mathcal{C},\mathcal{F}_\mathcal{S}) : \mathcal{G}_T(\mathbb{O},\mathbb{A}) \to \mathcal{G}'$:

$$\mathcal{F}_\mathcal{C}([\vec{x},v]) = [\![\overbrace{\vec{x}{:}\vec{Q};\_} \vdash v{:}R]\!]^{\mathcal{G}'}_\mathcal{C}$$

This is proved in an exactly analogous way, except that there are only two cases, one for the variable which is preserved because it is a projection, and one for the operator case which follows from the previous lemma and the preservation of the operator interpretations by the $\mathsf{Lin}(\mathbb{O})$-morphism.

Lemma 5.5.6 (Uniqueness)
The morphism $\mathcal{FT}^\mathcal{G} : \mathcal{G}_T(\mathbb{O},\mathbb{A}) \to \mathcal{G}$ is a $\mathsf{Lin}(\mathbb{O})$-map, and it is the unique such $\mathsf{Lin}(\mathbb{O})$-map.

Proof We prove this by considering the definition. Assume we have an arbitrary $\mathsf{Lin}(\mathbb{O})$-map $\mathcal{F} : \mathcal{G}_T(\mathbb{O},\mathbb{A}) \to \mathcal{G}$.

On Objects of $\mathcal{C}_T$, which are sequences of intuitionistic types of $\mathsf{Lin}(\mathbb{O})$, we know that $\mathcal{F}_\mathcal{C}(Q_1 \ldots Q_r) = [\![Q_1]\!]^\mathcal{G}_{\mathsf{M}_I} \times \ldots \times [\![Q_r]\!]^\mathcal{G}_{\mathsf{M}_I}$ since $\mathcal{F}$ strictly preserves the cartesian structure. But we have:

$$[\![Q_1]\!]^\mathcal{G}_{\mathsf{M}_I} \times \ldots \times [\![Q_r]\!]^\mathcal{G}_{\mathsf{M}_I} = [\![Q_1 \ldots Q_r]\!]^\mathcal{G}_\mathcal{C} = \mathcal{FT}^\mathcal{G}_\mathcal{C}(Q_1 \ldots Q_r)$$

On Arrows of $\mathcal{C}_T$, which are equivalence classes $[\vec{x}, \vec{t}]^{\vec{Q},\vec{R}}$, by definition we have $\mathcal{F}_{\mathcal{C}}([\vec{x}, \vec{v}]^{\vec{Q},\vec{R}}) = \langle \mathcal{F}_{\mathcal{C}}([\vec{x}, v_1]), \ldots, \mathcal{F}_{\mathcal{C}}([\vec{x}, v_s]) \rangle$ since $\mathcal{F}_{\mathcal{C}}$ strictly preserves the cartesian structure. But now we can see by our lemma that these arrows $[\vec{x}, v_i]$ must all be mapped according to the definition of $\mathcal{FT}_{\mathcal{C}}$.

On Objects of $\mathcal{S}_T$, which are sequences of linear primes of $\mathsf{Lin}(\mathbb{O})$, we know that $\mathcal{F}_{\mathcal{S}}(A_1 \ldots A_s) = [\![A_1]\!]^{\mathcal{G}} \otimes \ldots \otimes [\![A_s]\!]^{\mathcal{G}}$ because $\mathcal{F}_{\mathcal{S}}$ is strict monoidal. But we have:

$$[\![A_1]\!]^{\mathcal{G}} \otimes \ldots \otimes [\![A_s]\!]^{\mathcal{G}} = [\![A_1 \ldots A_s]\!]^{\mathcal{G}} = \mathcal{FT}^{\mathcal{G}}_{\mathcal{S}}(A_1 \ldots A_s)$$

On Arrows of $\mathcal{S}_T$, which are equivalence classes $[\vec{x}, v]^{\vec{A},\vec{B}}$, we use our result; given

$$\mathcal{F}_{\mathcal{S}}([\vec{y}'\vec{x}, \mathsf{let}\ \vec{y}\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ v]) = [\![\overbrace{\vec{y}{:}\vec{Q}; \vec{x}{:}\vec{A}} \vdash v{:}\otimes\vec{B}]\!]^{\mathcal{G}'}$$

we can easily see that in the special case of arrows in $\mathcal{S}_T$, this means that $\mathcal{F}_{\mathcal{S}}$ agrees with $\mathcal{FT}_{\mathcal{S}}$. Hence it follows that the arbitrary map $\mathcal{F}$ is the same as the canonical map $\mathcal{FT}$, which is therefore unique. □

This immediately implies that $\mathcal{G}_T(\mathbb{O}, \mathbb{A})$ is initial in the category $\mathsf{Cat}_{\mathsf{Lin}}(\mathbb{O}, \mathbb{A})$.

5.6 Output Naturality

We now consider our defined notion of output naturality of a given operator in a type theory $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$. Clearly, just as we have specified that linearly natural operators be interpreted by appropriate natural transformations, we might expect that output-natural operators be interpreted as natural transformations which are natural in the output-place of the appropriate argument. However, it is immediately obvious that not every model of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ interprets its output natural operators as such natural transformations, simply because in the model we only have candidates for the components of the natural transformations for objects which are the interpretation of types. We now proceed by defining a sub-class of models which do interpret output natural operators as natural transformations of this kind, and show how they relate to the original models.

Definition 5.6.1 (Output Natural Lin($\mathbb{O}$)-Interpretation)
Given a typing system $\mathsf{Lin}(\mathbb{O})$ with an output-parameterised set of operators O having arity:

$$\frac{(\vec{Q}_1; \vec{A}_1)B_1, \ldots, (\vec{Q}_r; \vec{A}_r)(B_r); (\vec{Q}'_1; \vec{A}'_1)B'_1, \ldots, (\vec{Q}'_{s-1}; \vec{A}'_{s-1})B'_{s-1}, (\vec{Q}'_s; \vec{A}'_s)(\_)}{(\vec{Q}''; \vec{A}'')(\_)}$$

we define an *output-natural-in-O interpretation* of the typing system $\mathsf{Lin}(\mathbb{O})$, which we again write $\mathcal{G}$, to be an interpretation of the typing system $\mathsf{Lin}(\mathbb{O})$ with a natural transformation

$$\begin{aligned}
[\![\mathsf{O}]\!]^{\mathcal{G}}_{\mathcal{O}_L} :& (\times_{i=1\ldots r}\mathcal{S}(F(_=) \otimes [\![\vec{Q}_i; \vec{A}_i]\!]^{\mathcal{G}}, [\![B_i]\!]^{\mathcal{G}}_{\mathsf{M}_L})) \times \\
& (\times_{j=1\ldots(s-1)}\mathcal{S}(F(_=) \otimes [\![\vec{Q}'_j]\!]^{\mathcal{G}} \otimes (_{-j}) \otimes [\![\vec{A}'_j]\!]^{\mathcal{G}}, [\![B'_j]\!]^{\mathcal{G}}_{\mathsf{M}_L})) \times \\
& (\mathcal{S}(F(_=) \otimes [\![\vec{Q}'_s]\!]^{\mathcal{G}} \otimes (_{-s}) \otimes [\![\vec{A}'_s]\!]^{\mathcal{G}}, (_\equiv))) \\
& \to \mathcal{S}(F(_=) \otimes [\![\vec{Q}'']\!]^{\mathcal{G}} \otimes (\otimes_{i=1\ldots s}(_{-s})) \otimes [\![\vec{A}'']\!]^{\mathcal{G}}, (_\equiv))
\end{aligned}$$

which is natural independently in each of the arguments $(_=)$,$(_\equiv)$ and $(_{-1})\ldots(_{-s})$ such that each operator O_A in the set has interpretation

$$[\![\mathsf{O}_A]\!]^{\mathcal{G}}_{\mathcal{O}_L} = [\![\mathsf{O}]\!]^{\mathcal{G}}_{\mathcal{O}_L}(_=, _{-1}, \ldots _{-s}, [\![A]\!]^{\mathcal{G}}_{\mathsf{M}_L})$$

We now note that each output-natural-in-O interpretation of a typing system $\mathsf{Lin}(\mathbb{O})$ induces an interpretation of $\mathsf{Lin}(\mathbb{O})$ by the obvious map which forgets the natural transformation $[\![\mathsf{O}]\!]_{\mathcal{O}_L}$.

This enables us to map the typing system $\mathsf{Lin}(\mathbb{O})$ into an output-natural-in-O interpretation of $\mathsf{Lin}(\mathbb{O})$.

Definition 5.6.2
Given a typing system $\mathsf{Lin}(\mathbb{O})$ with an output-parameterised set of operators O, we define the map $[\![\_]\!]^{\mathcal{G}}$ which takes the typing system $\mathsf{Lin}(\mathbb{O})$ to an output-natural-in-O interpretation $\mathcal{G}$ simply by defining it to be the map taking $\mathsf{Lin}(\mathbb{O})$ to the underlying interpretation of $\mathcal{G}$.

Given this map, we can go on to give a definition of output-natural $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model.

Definition 5.6.3 (Output-Natural-in-O $\mathrm{Lin}(\mathbb{O}, \mathbb{A})$-Model)
Given a typing system $\mathsf{Lin}(\mathbb{O})$ with an output-parameterised set of operators O, we define an *output-natural-in-O model* of the type theory $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$, which we also write $\mathcal{G}$, to be an output-natural-in-O interpretation of $\mathsf{Lin}(\mathbb{O})$ such that for each equality judgement $\Gamma; \Delta \vdash v = w{:}A$ in $\mathbb{A}$, we have

$$[\![\Gamma; \Delta \vdash v{:}A]\!]^{\mathcal{G}} = [\![\Gamma; \Delta \vdash w{:}A]\!]^{\mathcal{G}}$$

in the s.m. part of $\mathcal{G}$.

Given this definition, soundness is an easy corollary of soundness for our first definition of interpretation.

Lemma 5.6.4 (Soundness)
Given a typing system $\mathsf{Lin}(\mathbb{O})$ with an output-parameterised set of operators O, and an output-natural-in-O $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model $\mathcal{G}$, for every provable equality judgement $\Gamma; \Delta \vdash v = w{:}A$ of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, we have:

$$[\![\Gamma; \Delta \vdash v{:}A]\!]^{\mathcal{G}} = [\![\Gamma; \Delta \vdash w{:}A]\!]^{\mathcal{G}}$$

The important property of output-natural-in-O interpretations which distinguishes them from plain interpretations is the following:

Lemma 5.6.5
Given a typing system $\mathsf{Lin}(\mathbb{O})$ with an output-parameterised set of operators O and an output-natural-in-O interpretation $\mathcal{G}$, for every provable equality $\Gamma; \Delta \vdash v = w{:}A$ of the type theory $\mathsf{Lin}(\mathbb{O}, \mathsf{ONat}(\mathbb{O}, \mathsf{O}))$, we have:

$$[\![\Gamma; \Delta \vdash v{:}A]\!]^{\mathcal{G}} = [\![\Gamma; \Delta \vdash w{:}A]\!]^{\mathcal{G}}$$

This is easily provable adapting the proof of the previous soundness lemma. We only need to prove that the axiomatic equality judgements in the set $\mathsf{ONat}(\mathbb{O}, \mathsf{O})$ are soundly mapped to the output-natural-in-O interpretation, which follows by virtue of the fact that the operators in O are all interpreted as instances of a natural transformation.

Now we can define output-natural-in-O morphisms.

Definition 5.6.6 (Output-Natural-in-O Lin($\mathbb{O}$)-morphism)
Given a typing system $\mathsf{Lin}(\mathbb{O})$ with an output-parameterised set of operators O having arity

$$\frac{(\vec{Q}_1; \vec{A}_1)B_1, \ldots, (\vec{Q}_r; \vec{A}_r)(B_r); (\vec{Q}'_1; \vec{A}'_1)B'_1, \ldots, (\vec{Q}'_{s-1}; \vec{A}'_{s-1})B'_{s-1}, (\vec{Q}'_s; \vec{A}'_s)(\_)}{(\vec{Q}''; \vec{A}'')(\_)}$$

an *output-natural-in-O* $\mathsf{Lin}(\mathbb{O})$-morphism $\mathcal{F} : \mathcal{G} \to \mathcal{G}'$ between two output-natural-in-O $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-models is a $\mathsf{Lin}(\mathbb{O})$-morphism on the underlying $\mathsf{Lin}(\mathbb{O})$-interpretations of $\mathcal{G}$ and $\mathcal{G}'$ with the additional property that for any object X of $\mathcal{C}$, objects $Y_1, \ldots, Y_{s-1}$ and Y' of $\mathcal{S}$, and arrows of $\mathcal{S}$:

$$\begin{aligned}
&f_i : F(X) \otimes [\![\vec{Q}_i; \vec{A}_i]\!] \to [\![B_i]\!] \text{ for } i = 1 \ldots r \\
&g_j : F(X) \otimes [\![\vec{Q}'_j]\!] \otimes Y_j \otimes [\![\vec{A}'_j]\!] \to [\![B'_j]\!] \text{ for } j = 1 \ldots (s-1) \\
&g_s : F(X) \otimes [\![\vec{Q}'_j]\!] \otimes Y_j \otimes [\![\vec{A}'_j]\!] \to Y'
\end{aligned}$$

we have,

$$\begin{aligned}
&\mathcal{F}_{\mathcal{S}}([\![\mathsf{O}]\!]^{\mathcal{G}}_{\mathcal{O}_L}(X, Y_1, \ldots, Y_s, Y')(f_1, \ldots f_r, g_1, \ldots, g_s)) \\
&= [\![\mathsf{O}]\!]^{\mathcal{G}'}_{\mathcal{O}_L}(\mathcal{F}_{\mathcal{C}}(X), \mathcal{F}_{\mathcal{S}}(Y_1), \ldots \mathcal{F}_{\mathcal{S}}(Y_s), \mathcal{F}_{\mathcal{S}}(Y'))(\mathcal{F}_{\mathcal{S}}(f_1), \ldots \mathcal{F}_{\mathcal{S}}(f_r), \mathcal{F}_{\mathcal{S}}(g_1), \ldots, \mathcal{F}_{\mathcal{S}}(g_s))
\end{aligned}$$

We now write $\mathsf{Cat}^{\mathsf{O}}_{\mathsf{Lin}}(\mathbb{O},\mathbb{A})$ for the category of small output-natural-in-O models and morphisms. Clearly by the forgetful map mentioned earlier from output-natural-in-O $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-models to $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-models and its obvious extension to output-natural-in-O morphisms, we have a functor $\mathsf{Cat}^{\mathsf{O}}_{\mathsf{Lin}}(\mathbb{O},\mathbb{A}) \to \mathsf{Cat}_{\mathsf{Lin}}(\mathbb{O},\mathbb{A})$. In the opposite direction, given an a type theory $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ having an output-parameterised set of operators O, there may be no candidate natural transformation to make it into an output-natural-in-O model, or there may be one or many.

However, considering the term-model in particular, it is not the case that the term model $\mathcal{G}_T(\mathbb{O},\mathbb{A})$ for a type-theory $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ with an output-parameterised set of operators O can be extended to an output-natural-in-O model simply by picking an existing natural transformation for $[\![\mathsf{O}]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}_{\mathcal{O}_L}$. This is simply because there is no appropriate operator set, since no operator acts on arguments having as outputs for example 2-element sequences. All operators of $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$ act only on arguments with singleton outputs.

The Output-Natural Term Model

Since it is not the case that the term model $\mathcal{G}_T(\mathbb{O},\mathbb{A})$ can be extended to an output natural model, we need to reconsider the questions of completeness and initiality. We will sketch a variation on the construction of the term model which will give us an output-natural term $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model, and indicate how it can be proved initial for the category $\mathsf{Cat}^{\mathsf{O}}_{\mathsf{Lin}}(\mathbb{O},\mathbb{A})$.

First consider an extension of the type theory $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$.

Definition 5.6.7 (The Type Theory $\mathsf{Lin}^{\otimes}_{\mathsf{O}}(\mathbb{O},\mathbb{A})$)
For a signature $\mathbb{O} = (\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$ and an axiom set $\mathbb{A}$, such that the typing system $\mathsf{Lin}(\mathbb{O})$ has an output-parameterised set of operators O with arity

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1,\ldots,(\vec{Q}_r;\vec{A}_r)(B_r);(\vec{Q}'_1;\vec{A}'_1)B'_1,\ldots,(\vec{Q}'_{s-1};\vec{A}'_{s-1})B'_{s-1},(\vec{Q}'_s;\vec{A}'_s)(\_)}{(\vec{Q}'';\vec{A}'')(\_)}$$

we define the type theory $\mathsf{Lin}^{\otimes}_{\mathsf{O}}(\mathbb{O},\mathbb{A})$ as $\mathsf{Lin}((\mathsf{M}^{\otimes}_I, \mathsf{M}^{\otimes}_L, \mathcal{O}^{\otimes}_{L\,\mathsf{O}}), \mathbb{A}^{\otimes}_{\mathsf{O}})$, where $\mathsf{M}^{\otimes}_I$, $\mathsf{M}^{\otimes}_L$ and $\mathcal{O}^{\otimes}_I$ are as defined as for $\mathsf{Lin}^{\otimes}(\mathbb{O},\mathbb{A})$, and we define $\mathcal{O}^{\otimes}_{L\,\mathsf{O}}$ and $\mathbb{A}_{\mathsf{O}}$ as follows:

- $\mathcal{O}^{\otimes}_{L\,\mathsf{O}}$ is the set $\mathcal{O}^{\otimes}_L$ augmented with new operators $\mathsf{O}_{\otimes\vec{C}}$ of arity

  $$\frac{(\vec{Q}_1;\vec{A}_1)B_1,\ldots,(\vec{Q}_r;\vec{A}_r)(B_r);(\vec{Q}'_1;\vec{A}'_1)B'_1,\ldots,(\vec{Q}'_{s-1};\vec{A}'_{s-1})B'_{s-1},(\vec{Q}'_s;\vec{A}'_s)(\otimes\vec{C})}{(\vec{Q}'';\vec{A}'')(\otimes\vec{C})}$$

  for any non-singleton sequence $\otimes\vec{C} \in \mathsf{M}^{\otimes}_L$. Note that there already exists instances in the case where the sequence is a singleton.

- $\mathbb{A}^{\otimes}_{\mathsf{O}}$ is the set $\mathbb{A}^{\otimes}$ augmented for each output-natural set of operators O of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ with the equality judgements in the set $\mathsf{ONat}((\mathsf{M}^{\otimes}_{I}, \mathsf{M}^{\otimes}_{L}, \mathcal{O}^{\otimes}_{I}, \mathcal{O}^{\otimes}_{L\,\mathsf{O}}), \mathsf{O}')$, where O' is the set $\mathsf{O} \cup (\mathcal{O}^{\otimes}_{L\,\mathsf{O}} - \mathcal{O}^{\otimes}_{L})$, which is to say the set O with the extra operator instances added in the definition of $\mathcal{O}^{\otimes}_{L\,\mathsf{O}}$.

This type-theory differs from $\mathsf{Lin}^{\otimes}(\mathbb{O}, \mathbb{A})$ in that the output-natural set of operators O of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ is extended canonically to be an output-natural set of operators in $\mathsf{Lin}^{\otimes}_{\mathsf{O}}(\mathbb{O}, \mathbb{A})$. Now say that a pre-term of $\mathsf{Lin}^{\otimes}_{\mathsf{O}}(\mathbb{O}, \mathbb{A})$ is in canonical form if it is an instance of the following inductive definition, where the v are pre-terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$:

$$v_c ::= \otimes\vec{v} \mid \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ v_c \mid \mathsf{O}_{\vec{A}}((\vec{x};\vec{x})\vec{v}, \ldots, (\vec{x};\vec{x})\vec{v}; (\vec{x};\vec{x})\vec{v}, \ldots, (\vec{x};\vec{x})\vec{v}, (\vec{x};\vec{x})v_c)(\vec{v};\vec{v})$$

We then have the following extended version of lemma 5.3.2:

Lemma 5.6.8
Given a typing judgement $\widehat{\Gamma;\Delta} \vdash v{:}\otimes\vec{A}$ of $\mathsf{Lin}^{\otimes}_{\mathsf{O}}(\mathbb{O}, \mathbb{A})$, v is identically in canonical form with the various pre-terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ being terms.

This is proved in a very similar way to the original version of the lemma, with the one extension that we now have additional operators in $\mathcal{O}^{\otimes}_{\mathsf{O}}$ which may return results having outputs of tensor type. However, we know this can only occur when they are applied to an argument (in their output-parameterised place) which is of tensor type, and so we can use induction to show that the term is still in canonical form.

Given this result, we then proceed to define the term categories in much the same way as before, and define the *output-natural-in-O term model* of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ exactly as we do the term model of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ with the exception that we interpret the output-natural set of operators O in $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ into the appropriate natural transformation which exists in the term model by virtue of the extension of $\mathcal{O}$ to incorporate instances O_A for any type A of $\mathsf{M}^{\otimes}_{L}$ (which is a sequence of types of M_L) and the addition of the output-naturality equalities to $\mathbb{A}$.

Similarly, the definition of the canonical morphism from the output-natural-in-O term model to an arbitrary output-natural-in-O model is identical to the previous one with the exception that for the output-natural set of operators O we map the natural transformation which exists in the output-natural term model to the natural transformation in the arbitrary model.

Given this development, we have that the output-natural-in-O $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-term model, which we will call $\mathcal{G}^{\mathsf{O}}_{T}(\mathbb{O}, \mathbb{A})$, is initial in the category $\mathsf{Cat}^{\mathsf{O}}_{\mathsf{Lin}}(\mathbb{O}, \mathbb{A})$.

Multiary Output-Natural Models

Most of the leading examples of linear type-theories which we will present have more than one output-natural set of operators, and so we briefly consider the situation in this case. In fact, it is easy to extend all the definitions of "output-natural-in-O" constructions into the analogous "output-natural-in-O-and…-and-O'" forms, and the constructions go through as before, with the same results being provable.

Chapter 6

Action Calculi and Extensions

We now present an important example of a system having a linear structural framework, Milner's action calculus [Mil93c]. The action calculus is similar to our general linear type-theory in that it is a general framework in which many different systems can be presented, particularly process calculi. The leading example of an action calculus is the π-calculus [MPW92], but systems ranging from petri-nets to the λ-calculus have been shown to correspond to action calculi.

The essential idea is that we have certain underlying structure, including a monoidal structure and a discipline of *names*, and in addition, for each instance we have a *signature* containing a set of intuitionistically natural operators (called *controls*) and associated *reaction rules* which depend on the particular calculus we wish to describe. For example, there is a set of controls $\mathcal{K}_\pi$ and a set of reaction rules $\searrow_\pi$ such that the action calculus $\mathsf{AC}(\mathcal{K}_\pi, \searrow_\pi)$ is isomorphic to the π-calculus, and there are other sets of controls and reaction rules serving the same purpose for other languages such as petri-nets and the λ-calculus.

We will actually present a slight generalisation of Milner's original action calculi as found in [Mil96], which will incorporate a notion of *linear prime arity* in addition to Milner's *primes*, which have intuitionistic behaviour. This generalisation is motivated by our discussion of general linear type-theories, which can have both linear and intuitionistic primitive types; further, we have an example of an action calculus of this more general form which is not isomorphic to an action calculus as defined by Milner. This is the action calculus corresponding to the linear λ-calculus, which we will call $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$. However, we note that all Milner's action calculi are instances of this more general form.

We will also define several higher-order extensions of the basic action calculi, inspired variously by Milner's higher order action calculi [Mil94b] and the higher-order structure present in linear logic.

6.1 Action Calculi

We first assume a set P_I of *intuitionistic primes*, ranged over by $p, q \ldots$ and a set P_L of *linear primes*, ranged over by $l, k \ldots$, such that $\mathsf{P}_I \subseteq \mathsf{P}_L$. We can now define the set of *(tensor) arities* over the set of linear primes, written $\mathsf{M}(\mathsf{P}_L)$, as the set of sequences of primes in P_L. We use $m, n \ldots$ for (tensor) arities (henceforth just arities), and write (infix) $\otimes$ for concatenation and ε for the empty sequence. Further, we will omit the argument P_L of M where it is clear from the context.

We can now give the definition of signature:

Definition 6.1.1 (Action Calculus Signature)
An *action calculus signature* $\mathbb{K}$ is a triple $(\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$ of sets such that $\mathsf{P}_I \subseteq \mathsf{P}_L$ and each K in $\mathcal{K}$ has an arity of the form:

$$((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$$

where the m's and n's are from the set $\mathsf{M}(\mathsf{P}_L)$.

Definition 6.1.2 (Intuitionistic Action Calculus Signature)
An action calculus signature $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$ is *intuitionistic* if $\mathsf{P}_I = \mathsf{P}_L$.

Intuitionistic action calculus signatures give rise to intuitionistic action calculi, which are equivalent to Milner's original action calculi.

In order to simplify definitions, in the following we assume that a signature $\mathbb{K}$ has components $(\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$ unless otherwise stated, and similarly $\mathbb{K}'$ has components $(\mathsf{P}'_I, \mathsf{P}'_L, \mathcal{K}')$ etc.

In order to define the terms of the action calculus, we use the familiar variable set X. However, in this context these variables are known as *names*, and further it is necessary to assume that each name x has an associated *intuitionistic* prime arity p; we may write x^p to denote this.

Definition 6.1.3 (Terms)
Terms over an action calculus signature $\mathbb{K}$, written $a, b \ldots$, are constructed from the basic operators *identity*, $\mathbf{id}_m$, *permutation*, $\mathbf{p}_{m,n}$, *composition*, $\cdot$, *tensor*, $\otimes$, *abstraction*, $(x^p)\_$, *datum*, $\langle x^p \rangle$ and the control operators, K. A term a is assigned an *(action) arity* $a : m \to n$ where m, n are tensor arities, using the following rules:

$$\mathbf{id}_m : m \to m \qquad \frac{a : m \to m' \qquad b : m' \to n}{a \cdot b : m \to n}$$

$$\frac{a : m_1 \to n_1 \qquad b : m_2 \to n_2}{a \otimes b : m_1 \otimes m_2 \to n_1 \otimes n_2} \qquad \mathbf{p}_{m,n} : m \otimes n \to n \otimes m$$

$$\frac{a : m \to n}{(x^p)a : p \otimes m \to n} \qquad \langle x^p \rangle : \varepsilon \to p$$

$$\frac{a_1 : m_1 \to n_1 \quad \ldots \quad a_r : m_r \to n_r}{\mathsf{K}(a_1, \ldots, a_r) : m \to n}\ \mathsf{K}$$

where the control operator K from the signature $\mathbb{K}$ has the arity

$$((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$$

We will omit arity annotations, both in terms and of terms, where these are apparent. The construct $(x)a$ binds the free name x in the term a, and the name x occurs free in the term $\langle x \rangle$. We write $a\{b/\langle x \rangle\}$ to denote the usual capture-avoiding substitution of terms for subterms of the form $\langle x \rangle$. Note that all free occurrences of x are in subterms of this form.

Given a sequence of names $x_1^{p_1} \ldots x_r^{p_r}$, we write $|x_1 \ldots x_r|$ to denote the arity $p_1 \ldots p_r$. Further, we define the constructs $(x_1 \ldots x_r)a = (x_1) \ldots (x_r)a$ and $\langle x_1 \ldots x_r \rangle = \langle x_1 \rangle \otimes \ldots \otimes \langle x_r \rangle$, where this is the left-bracketed tensor. Note, however, that since the tensor will be strict, the choice is unimportant.

We give the equality on the terms, which is adapted slightly from that of Milner.

Definition 6.1.4 (The Theory AC)

An equality holds between two terms of AC in the equational theory AC if it can be proved using the following axioms, annotated with action arities, and the obvious reflexivity, transitivity and congruence rules:

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c \tag{A1}$$
$$a \cdot \mathbf{id}_n = a = \mathbf{id}_m \cdot a \tag{A2}$$
$$(a \cdot b) \otimes (c \cdot d) = (a \otimes c) \cdot (b \otimes d) \tag{A3}$$
$$\mathbf{id}_m \otimes \mathbf{id}_n = \mathbf{id}_{m \otimes n} \tag{A4}$$
$$a \otimes (b \otimes c) = (a \otimes b) \otimes c \tag{A5}$$
$$a \otimes \mathbf{id}_\varepsilon = a = \mathbf{id}_\varepsilon \otimes a \tag{A6}$$

$$\mathbf{p}_{m,n} \cdot (b \otimes a) = (a \otimes b) \cdot \mathbf{p}_{m',n'} \tag{ζ_1}$$
$$\mathbf{p}_{m \otimes n, m'} = (\mathbf{id}_m \otimes \mathbf{p}_{n,m'}) \cdot (\mathbf{p}_{m,m'} \otimes \mathbf{id}_n) \tag{ζ_2}$$
$$\mathbf{p}_{\varepsilon,m} = \mathbf{id}_m \tag{ζ_3}$$
$$\mathbf{p}_{m,n} \cdot \mathbf{p}_{n,m} = \mathbf{id}_{m \otimes n} \tag{ζ_4}$$

augmented with the two naming axioms:

$$(\langle y \rangle \otimes \mathbf{id}_m) \cdot (x)a = a\{\langle y \rangle / \langle x \rangle\} \tag{σ}$$
$$(x)((\langle x \rangle \otimes \mathbf{id}) \cdot a) = a, \text{ where } x \notin \mathit{fn}(a) \tag{δ}$$

It is an immediate consequence of these axioms that $\mathbf{id}_m = (\vec{x})\langle\vec{x}\rangle$ and $\mathbf{p}_{m,n} = (\vec{x}\vec{y})\langle\vec{y}\vec{x}\rangle$, where $|\vec{x}| = \vec{p} = m$ and $|\vec{y}| = \vec{q} = n$.

Definition 6.1.5 (Action Calculus)
The *action calculus* $\mathsf{AC}(\mathbb{K}, \searrow)$ is the quotient of the terms of $\mathsf{AC}(\mathbb{K})$ by the equational theory AC, together with the reaction relation $\searrow$, which is a transitive relation closed under tensor, composition, abstraction and equality, and such that there is no action a such that $\mathbf{id} \searrow a$.

We refer to the terms of the theory as *actions*. From now on we will consider only the static case, which is to say the case in which the reaction relation is empty. Possible extensions of these definitions and the following results to the case with a non-empty reaction relation are discussed in chapter 10.

We now briefly show how this definition relates to Milner's original definition, found in [Mil96].

Definition 6.1.6 (Intuitionistic Action Calculus)
An action calculus $\mathsf{AC}(\mathbb{K})$ is *intuitionistic* if it is constructed over an intuitionistic action calculus signature.

We can now show:

Lemma 6.1.7
An intuitionistic action calculus $\mathsf{AC}(\mathbb{K})$ where $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_I, \mathcal{K})$ is isomorphic to Milner's action calculus $\mathsf{AC}(\mathcal{K})$ built over the primes P_I, up to equality in the appropriate theory.

Proof This is proved by giving inverse translations. To translate from an intuitionistic action calculus $\mathsf{AC}(\mathbb{K})$ to the corresponding instance of Milner's action calculus, the translation is the identity on terms except on $\mathbf{p}_{m,n}$, which we map to the term $(\vec{x})(\vec{y})\langle\vec{y}\rangle \otimes \langle\vec{x}\rangle$. Conversely, to translate from Milner's action calculus back to the intuitionistic action calculus $\mathsf{AC}(\mathbb{K})$, we simply map terms to themselves. It is easy to show that these two translations are sound with respect to the appropriate theories, and they are inverse simply because in the action calculus $\mathsf{AC}(\mathbb{K})$, given sequences of *intuitionistic* primes m and n,

$$\mathbf{p}_{m,n} = (\vec{x})(\vec{y})\langle\vec{y}\rangle \otimes \langle\vec{x}\rangle$$

□

6.2 Generalised Linear Type Theory

We now present a generalised linear type-theory which corresponds to the static action calculus $\mathsf{AC}(\mathbb{K})$.

Definition 6.2.1 (The Type Theory $\mathrm{Lin}^{\mathrm{A}}(\mathbb{K})$)
Given an action calculus signature $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$, we define the type theory $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$ to be the instance $\mathsf{Lin}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}})$, where $\mathbb{O}^{\mathsf{A}} = (\mathsf{M}_I^{\mathsf{A}}, \mathsf{M}_L^{\mathsf{A}}, \mathcal{O}_I^{\mathsf{A}}, \mathcal{O}_L^{\mathsf{A}})$ and $\mathsf{M}_I^{\mathsf{A}}$, $\mathsf{M}_L^{\mathsf{A}}$, $\mathcal{O}_I^{\mathsf{A}}$, $\mathcal{O}_L^{\mathsf{A}}$ and $\mathbb{A}^{\mathsf{A}}$ are defined as follows:

- The set $\mathsf{M}_I^{\mathsf{A}}$ is the set of all singleton sequences of elements of P_I.
- The set $\mathsf{M}_L^{\mathsf{A}}$ is the set of all sequences of elements of P_L. For consistency with the action calculus, we may write $m, n \ldots$ for arbitrary sequences of elements of P_L $l_1 \ldots l_r$, and further we may write $\otimes$ for concatenation and ε for the empty sequence.
- The set $\mathcal{O}_I^{\mathsf{A}}$ is the empty set.
- The set $\mathcal{O}_L^{\mathsf{A}}$ contains:
  1. For each control K from $\mathcal{K}$ of arity

$$((\vec{l_1}, n_1), \ldots, (\vec{l_r}, n_r)) \rightarrow (\vec{l'}, n')$$

an operator O^{K} of arity

$$\frac{(;\vec{l_1})n_1, \quad \ldots, \quad (;\vec{l_r})n_r \quad ;}{(;\vec{l'})n'}$$

where we overload notation to write $\vec{l}$ for the sequence of singleton sequences of the elements of $\vec{l}$. We will abbreviate the general operator form

$$O^{\mathsf{K}}((;\vec{x}_1:\vec{l_1})v_1, \ldots, (;\vec{x}_r:\vec{l_r})v_r;)(;\vec{w})$$

as $\mathsf{K}((\vec{x}_1:\vec{l_1})v_1, \ldots, (\vec{x}_r:\vec{l_r})v_r, \vec{w})$

2. For each r and $m_1 \ldots m_r$, an operator $\otimes^R_{m_1 \ldots m_r}$ of arity:

$$\frac{; \quad (;)m_1 \quad \ldots \quad (;)m_r}{(;)m_1 \otimes \ldots \otimes m_r}$$

In terms of the type theory, we will write $v_1 \otimes \ldots v_r$ or occasionally $\otimes(v_1 \ldots v_r)$ as an abbreviation for $\otimes^R_{m_1 \ldots m_r}(;(;)v_1, \ldots (;)v_r)(;)$.

3. For each r and $m_1 \ldots m_r$, each of an output-parameterised set of operators $\otimes^L_{m_1 \ldots m_r}$ with arity:

$$\frac{; \quad (;m_1 \ldots m_r)(\_)}{(;m_1 \otimes \ldots \otimes m_r)(\_)}$$

In terms of the type theory, we will write $\mathsf{let} \otimes \vec{x}{:}\vec{m}\ \mathsf{be}\ \vec{v}\ \mathsf{in}\ w$ as an abbreviation for $\otimes^L_{m_1 \ldots m_r}(;(;\vec{x})w)(;v)$, where $\vec{m} = m_1 \ldots m_r$.

- The set $\mathbb{A}^{\mathsf{A}}$ contains:

1. All well-formed equalities of the form:

$$(\otimes - \beta) \;\; \Gamma;\Delta \vdash \mathsf{let} \otimes \vec{x}\vec{y}\ \mathsf{be}\ w_1 \otimes w_2\ \mathsf{in}\ v = \mathsf{let} \otimes \vec{x}\ \mathsf{be}\ w_1\ \mathsf{in}\ \mathsf{let} \otimes \vec{y}\ \mathsf{be}\ w_2\ \mathsf{in}\ v{:}n$$

where $\Gamma;\Delta_1 \vdash w_1{:}\otimes\vec{m}$, $\Gamma;\Delta_2 \vdash w_2{:}\vec{m}'$, $\Gamma;\Delta',\vec{x}{:}\vec{m},\vec{y}{:}\vec{m}' \vdash v{:}n$ and $\Delta = \Delta'\#\Delta_1\#\Delta_2$.

2. All well-formed equalities of the form:

$$(\otimes - \eta) \;\; \Gamma;\Delta \vdash \mathsf{let} \otimes \vec{x}\ \mathsf{be}\ v\ \mathsf{in} \otimes \vec{x} = v : \otimes\vec{m}$$

3. All well-formed equalities of the form:

$$(\otimes - 1) \;\; \Gamma;\Delta \vdash \otimes(\otimes\vec{v}_1, \ldots, \otimes\vec{v}_r) = \otimes(\vec{v}_1 \ldots \vec{v}_r){:}n$$

4. All well-formed equalities of the form:

$$(\otimes - 2) \;\; \Gamma; \Delta \vdash \otimes(v) = v{:}n$$

5. For any r, all equality judgements in any of the sets $\mathsf{ONat}(\mathbb{O}^{\mathsf{A}}, \otimes^{L}_{m_1 \ldots m_r})$ for any $m_1 \ldots m_r$.

Having given this definition, it is worth noting that we could present an equivalent type theory without any type constructors if we used general sequents of the form $\Gamma; \Delta \vdash \Delta'$, where the sequence Δ' on the right is intended to be read as a tensor of linear prime arities, as is the sequence Δ on the left. Given this logic we could annotate each sequent having such a form with a term v such that the typing $\Gamma; \Delta \vdash v{:}\Delta'$ would correspond to an action $a{:}|\Delta| \to |\Delta'|$ having free names in Γ. This is the approach taken in [BGHP97].

6.3 The Translations

We present the translations which make $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$ isomorphic to the action calculus $\mathsf{AC}(\mathbb{K})$.

Action Calculi to Type Theory

We first give a translation which will take the action calculus $\mathsf{AC}(\mathbb{K})$ to the type theory $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$. We define the function $(\_)^{\dagger}$ which takes pairs of a sequence of distinct variables and an action to pre-terms of $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$, where the sequence of variables has the same length as the sequence of prime arities m for the action $a{:}m \to n$.

Definition 6.3.1 (The Translation $(\_)^{\dagger}$)
We define the translation over the structure of actions as follows:

$$\begin{aligned}
(\vec{y}, \mathbf{id}_m)^{\dagger} &= \otimes(y_1 \ldots y_r) \text{ (where } m = l_1 \ldots l_r) \\
(\vec{y}, a \cdot b)^{\dagger} &= \mathsf{let}\ \otimes\vec{y}'\ \mathsf{be}\ (\vec{y}, a)^{\dagger}\ \mathsf{in}\ (\vec{y}', b)^{\dagger} \\
(\vec{y}\vec{y}', a \otimes b)^{\dagger} &= (\vec{y}, a)^{\dagger} \otimes (\vec{y}', b)^{\dagger} \text{ (where } \vec{y} = y_1 \ldots y_r \text{ and } a{:}l_1 \ldots l_r \to n) \\
(\vec{y}\vec{y}', \mathbf{p}_{m,n})^{\dagger} &= (\otimes\vec{y}') \otimes (\otimes\vec{y}) \text{ (where } \vec{y} = y_1 \ldots y_r \text{ and } m = l_1 \ldots l_r) \\
(y'\vec{y}, (x^p)a)^{\dagger} &= \mathsf{let}\ x\ \mathsf{be}\ y'\ \mathsf{in}\ (\vec{y}, a)^{\dagger} \\
(\varepsilon, \langle x^p \rangle)^{\dagger} &= x \\
(\vec{y}', \mathsf{K}(a_1 \ldots a_r))^{\dagger} &= \mathsf{K}((\vec{y}_1)(\vec{y}_1, a_1)^{\dagger} \ldots (\vec{y}_r)(\vec{y}_r, a_r)^{\dagger}, \vec{y}')
\end{aligned}$$

where the y' and y_i for all i are fresh in each clause.

We can now easily show by induction over the structure of actions the following lemma:

Lemma 6.3.2
Given an action $a : m \to n$ with free names $\vec{x}$ such that $|\vec{x}| = \vec{p}$, we have that $\vec{x}{:}\vec{p}; \vec{y}{:}\vec{l} \vdash (\vec{y}, a)^\dagger{:}n$ in $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$, where $\vec{y} = y_1 \dots y_r$ and $m = \vec{l} = l_1 \dots l_r$.

Further, we can show that this map translates equalities between terms in $\mathsf{AC}(\mathbb{K})$ to equality judgements of $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$. In order to do this we first prove an easy lemma.

Lemma 6.3.3
Given an action $a{:}m \to n$ with free names $\vec{x}$ such that $|\vec{x}| = \vec{p}$ and where $m = \vec{l} = l_1 \dots l_r$, we have that:

$$\vec{x}{:}\vec{p}; \vec{z}{:}\vec{l} \vdash \mathsf{let}\ \otimes\vec{y}\ \mathsf{be}\ \otimes\vec{z}\ \mathsf{in}\ (\vec{y}, a)^\dagger = (\vec{z}, a)^\dagger{:}n$$

Lemma 6.3.4 (Soundness)
Given action $a{:}m \to n$ and $b{:}m \to n$, whose free names are included in $\vec{x}$ such that $|\vec{x}| = \vec{p}$, we have that if $a = b$ in the theory of $\mathsf{AC}(\mathbb{K})$,

$$\vec{x}{:}\vec{p}; \vec{y}{:}\vec{l} \vdash (\vec{y}, a)^\dagger = (\vec{y}, b)^\dagger{:}n$$

where $\vec{y} = y_1 \dots y_r$ and $m = l_1 \dots l_r$.

Proof We prove this simply by considering each axiomatic equality of $\mathsf{AC}(\mathbb{K})$, since the congruence, symmetry, transitivity and reflexivity rules of $\mathsf{AC}(\mathbb{K})$ are duplicated in $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$. For brevity we will write $(\vec{y}, a)^\dagger$ as $a^\dagger_{\vec{y}}$ from now on, and we assume that in all the clauses below $\Gamma = \vec{x}{:}\vec{p}$.

A1) In this case, we need to show the equality

$$\begin{aligned}\Gamma; \vec{y}{:}\vec{l} \vdash &\mathsf{let}\ \otimes\vec{y}'\ \mathsf{be}\ a^\dagger_{\vec{y}}\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}''\ \mathsf{be}\ b^\dagger_{\vec{y}'}\ \mathsf{in}\ c^\dagger_{\vec{y}''} \\ &= \mathsf{let}\ \otimes\vec{y}''\ \mathsf{be}\ (\mathsf{let}\ \otimes\vec{y}'\ \mathsf{be}\ a^\dagger_{\vec{y}}\ \mathsf{in}\ b^\dagger_{\vec{y}'})\ \mathsf{in}\ c^\dagger_{\vec{y}''}{:}n'\end{aligned}$$

However, this is one of the output-naturality equalities for this operator.

A2) In this case, we need to show the equality:

$$\Gamma; \vec{y}{:}\vec{l} \vdash \mathsf{let}\ \otimes\vec{y}'\ \mathsf{be}\ a^\dagger_{\vec{y}}\ \mathsf{in}\ \otimes\vec{y}' = a^\dagger_{\vec{y}}{:}n$$

which is clearly just one of the η-equalities we have given for this type-theory.

A3) In this case, we need to show the equality:

$$\Gamma; \vec{y}_1{:}\vec{l}_1, \vec{y}_2{:}\vec{l}_2 \vdash \mathsf{let}\ \otimes\vec{y}_1'\ \mathsf{be}\ a^{\dagger}_{\vec{y}_1}\ \mathsf{in}\ b^{\dagger}_{\vec{y}_1'} \otimes \mathsf{let}\ \otimes\vec{y}_2'\ \mathsf{be}\ c^{\dagger}_{\vec{y}_2}\ \mathsf{in}\ d^{\dagger}_{\vec{y}_2'}$$
$$= \mathsf{let}\ \otimes\vec{y}_1'\vec{y}_2'\ \mathsf{be}\ a^{\dagger}_{\vec{y}_1} \otimes c^{\dagger}_{\vec{y}_2}\ \mathsf{in}\ b^{\dagger}_{\vec{y}_1'} \otimes d^{\dagger}_{\vec{y}_2'}{:}\otimes(n_1'n_2')$$

which holds via a use of $\otimes - \beta$ and some commuting conversions which are due to output naturality.

A4) In this case, we need to show the equality:

$$\_; \vec{y}_1{:}\vec{l}_1, \vec{y}_2{:}\vec{l}_2 \vdash (\otimes\vec{y}_1) \otimes (\otimes\vec{y}_2) = \otimes\vec{y}_1\vec{y}_2{:}\otimes(\vec{l}_1\vec{l}_2)$$

but this follows from our tensor equality.

A5) In this case we need to show the equality:

$$\Gamma; \vec{y}_1{:}\vec{l}_1, \vec{y}_2{:}\vec{l}_2, \vec{y}_3{:}\vec{l}_3 \vdash a^{\dagger}_{\vec{y}_1} \otimes (b^{\dagger}_{\vec{y}_2} \otimes c^{\dagger}_{\vec{y}_3}) = (a^{\dagger}_{\vec{y}_1} \otimes b^{\dagger}_{\vec{y}_2}) \otimes c^{\dagger}_{\vec{y}_3}{:}\otimes(n_1n_2n_3)$$

But again this follows directly from our tensor equality.

A6) In this case, we need to show the equality:

$$\Gamma; \vec{y}{:}\vec{l} \vdash a^{\dagger}_{\vec{y}} \otimes (\otimes\varepsilon) = a^{\dagger}_{\vec{y}}{:}n$$

and its symmetric variant. However, these again follow from our tensor equality.

ζ_1) In this case we need to show the the equality:

$$\Gamma; \vec{y}_1 : \vec{l}_2, \vec{y}_2 : \vec{l}_2 \vdash \mathsf{let}\ \otimes\vec{y}_2'\vec{y}'_1\ \mathsf{be}\ (\otimes\vec{y}_2) \otimes (\otimes\vec{y}_1)\ \mathsf{in}\ b^{\dagger}_{\vec{y}_2'} \otimes a^{\dagger}_{\vec{y}_1'} = a^{\dagger}_{\vec{y}_1'} \otimes b^{\dagger}_{\vec{y}_2'} : \otimes(n_1n_2)$$

which is true via the α-conversion result lemma 6.3.3.

ζ_2) In this case, we need to show the equality:

$$\Gamma; \vec{y}_1{:}\vec{l}_1, \vec{y}_2{:}\vec{l}_2, \vec{y}_3{:}\vec{l}_3 \vdash (\otimes\vec{y}_3) \otimes (\otimes\vec{y}_2\vec{y}_1) =$$
$$\mathsf{let}\ \otimes\vec{y}_1'\vec{y}_3'\vec{y}_2'\ \mathsf{be}\ (\otimes\vec{y}_1) \otimes (\otimes\vec{y}_3\vec{y}_2)\ \mathsf{in}\ (\otimes\vec{y}_3'\vec{y}_1') \otimes (\otimes\vec{y}_2'){:}\otimes(\vec{l}_3\vec{l}_1\vec{l}_2)$$

which is true by virtue of the tensor equality, the $\otimes - \beta$-equality and our result on α-conversion lemma 6.3.3.

ζ_3) In this case we need to show the equality:

$$\_; \vec{y}{:}\vec{l} \vdash \otimes(\vec{y}\varepsilon) = \otimes\vec{y}{:}\otimes\vec{l}$$

which is true by definition since ε is the empty sequence.

ζ_4) In this case we need to show the equality:

$$\_; \vec{y}_1{:}\vec{l}_1, \vec{y}_2{:}\vec{l}_2 \vdash \mathsf{let}\ \otimes\vec{y}_2'\vec{y}_1'\ \mathsf{be}\ \otimes\vec{y}_2\vec{y}_1\ \mathsf{in}\ \otimes\vec{y}_1'\vec{y}_2' = \otimes\vec{y}_1\vec{y}_2{:}\otimes(\vec{l}_1\vec{l}_2)$$

which follows by $\otimes - \beta$ and our α-conversion result lemma 6.3.3.

σ) In this case we need to show the equality:

$$\Gamma, y{:}p; \vec{z}{:}\vec{l} \vdash \mathsf{let}\ \otimes z''\vec{z}'\ \mathsf{be}\ y \otimes (\otimes\vec{z})\ \mathsf{in}\ \mathsf{let}\ x\ \mathsf{be}\ z''\ \mathsf{in}\ a^{\dagger}_{\vec{z}'} = a^{\dagger}_{\vec{z}}\{y/x\}{:}n$$

which follows by $\otimes - \beta$, $F - \beta$ and our α-conversion result lemma 6.3.3.

δ) In this case we need to show the equality:

$$\Gamma; y'{:}p, \vec{y}{:}\vec{l} \vdash \mathsf{let}\ x\ \mathsf{be}\ y'\ \mathsf{in}\ \mathsf{let}\ \otimes z'\vec{z}\ \mathsf{be}\ x \otimes \vec{y}\ \mathsf{in}\ a^{\dagger}_{z'\vec{z}} = a^{\dagger}_{y'\vec{y}}{:}n$$

which follows by $\otimes - \beta$, $cc - 4$, $F - \eta$ and an instance of linear naturality, using our α-conversion result lemma 6.3.3. □

Type Theory to Action Calculi

We now give a translation which maps the type theory $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$ to the action calculus $\mathsf{AC}(\mathbb{K})$. In contrast to the previous translations, we proceed in this case by defining an action corresponding to each $\Gamma; \Delta$ term.

Definition 6.3.5 (The Translation $(\_)^{\ddagger}$)
We define the translation $(\_)^{\ddagger}$ on $\Gamma; \Delta$-terms of $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$ as follows, where we write $(v)^{\ddagger}_{\Gamma;\Delta}$ for the action which is the image of the $\Gamma; \Delta$-term v.

$$\begin{aligned}
(x)^{\ddagger}_{\Gamma,x:p,\Gamma';_} &= \langle x \rangle \\
(x)^{\ddagger}_{\Gamma;x:m} &= \mathbf{id}_m \\
(\mathsf{let}\ x{:}p\ \mathsf{be}\ v\ \mathsf{in}\ w)^{\ddagger}_{\Gamma;\Delta} &= \mathsf{perm}_{\Delta,\Delta_1\Delta_2} \cdot ((v)^{\ddagger}_{\Gamma;\Delta_1} \otimes \mathbf{id}_{\otimes|\Delta_2|}) \cdot (x)(w)^{\ddagger}_{\Gamma,x:p,\Delta_2} \\
(\mathsf{K}((\vec{y}_1{:}\vec{l}_1)v_1 \ldots (\vec{y}_r{:}\vec{l}_r)v_r, \vec{w}))^{\ddagger}_{\Gamma;\Delta} &= \mathsf{mperm}_{\Delta,\Delta_1\ldots\Delta_s} \cdot ((w_1)^{\ddagger}_{\Gamma;\Delta_1} \otimes \ldots \otimes (w_s)^{\ddagger}_{\Gamma;\Delta_s}) \cdot \\
&\qquad \mathsf{K}((v_1)^{\ddagger}_{\Gamma;\vec{y}_1:\vec{l}_1} \ldots (v_r)^{\ddagger}_{\Gamma;\vec{y}_r:\vec{l}_r}) \\
(\otimes(v_1 \ldots v_r))^{\ddagger}_{\Gamma;\Delta} &= \mathsf{mperm}_{\Delta,\Delta_1\ldots\Delta_r} \cdot ((v_1)^{\ddagger}_{\Gamma;\Delta_1} \otimes \ldots \otimes (v_1)^{\ddagger}_{\Gamma;\Delta_1}) \\
(\mathsf{let}\ \otimes\vec{y}{:}\vec{m}\ \mathsf{be}\ v\ \mathsf{in}\ w)^{\ddagger}_{\Gamma;\Delta} &= \mathsf{perm}_{\Delta,\Delta_2,\Delta_1} \cdot (\mathbf{id}_{\otimes|\Delta_2|} \otimes (v)^{\ddagger}_{\Gamma;\Delta_1}) \cdot (w)^{\ddagger}_{\Gamma;\Delta_2,\vec{y}:\vec{m}}
\end{aligned}$$

where firstly, $\mathsf{perm}_{\Delta,\Delta_1,\Delta_2}$ is the canonical action with arity $\otimes|\Delta| \to (\otimes|\Delta_1|) \otimes (\otimes|\Delta_2|)$, where $\Delta = \Delta_1 \# \Delta_2$ and Δ_1 and Δ_2 are uniquely determined from the derivation of the terms v and w, and secondly $\mathsf{mperm}_{\Delta,\Delta_1\ldots\Delta_r}$ is the multiary version.

We now show that this translation is sound with respect to equality judgements $\Gamma;\Delta \vdash v = w{:}n$ in $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$. This is done by induction over the structure of equality judgements, but by familiar arguments we need only show it for the axiomatic equalities of $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$. First we prove a substitution lemma.

Lemma 6.3.6
Given that $\Gamma;\Delta_1 \vdash v{:}m$ and $\Gamma;\Delta_2, y{:}m \vdash w{:}n$, we have that:

$$(w\{v/y\})^{\ddagger}_{\Gamma;\Delta} = \mathsf{perm}_{\Delta,\Delta_2,\Delta_1} \cdot (\mathbf{id}_{|\Delta_2|} \otimes (v)^{\ddagger}_{\Gamma;\Delta_1}) \cdot (w)^{\ddagger}_{\Gamma;\Delta_2,y:m}$$

where $\Delta = \Delta_1 \# \Delta_2$.

Lemma 6.3.7 (Soundness)
Given an equality judgement $\Gamma;\Delta \vdash v = w : m$ of $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$, we have that the actions $(a)^{\ddagger}_{\Gamma;\Delta}$ and $(b)^{\ddagger}_{\Gamma;\Delta}$ are equal in the theory AC.

Proof This is proved by induction over the structure of equality judgements, but by familiar arguments we need only show it for the axiomatic equalities of $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$. Considering firstly the axiomatic equalities common to all the type-theories $\mathsf{Lin}(\mathbb{O})$, we have:

$F-\beta_V$) In this case we need to show the equality between actions:

$$\mathsf{perm}_{\Delta,\_\Delta} \cdot ((y)^{\ddagger}_{\Gamma,y:p,\Gamma';\_} \otimes \mathbf{id}_{\otimes|\Delta|}) \cdot (x)(w)^{\ddagger}_{\Gamma,y:p,\Gamma',x:p;\Delta} = (v\{x/y\})^{\ddagger}_{\Gamma,y:p,\Gamma';\Delta}$$

This is a direct consequence of the σ axiom of the action calculus, given the fact that the permutation $\mathsf{perm}_{\Delta,\_,\Delta}$ is just the identity.

$F-\eta$) In this case we need to show the equality between actions:

$$((v)^{\ddagger}_{\Gamma;\Delta} \otimes \mathbf{id}_{\varepsilon}) \cdot (x)(x)^{\ddagger}_{\Gamma,x:p;\_} = (v)^{\ddagger}_{\Gamma;\Delta}$$

But this follows directly as an instance of the δ equality.

cc's) In the case of the cc's (1-4) and cc-5 for the linearly natural operators $\otimes$ and $\mathsf{let}\ -\otimes$, the equalities are preserved by properties of tensor and composition.

Now, considering the axiomatic equalities in the set $\mathbb{A}^{\mathsf{A}}$, we have:

1. Considering the $\otimes-\beta$ equality, we need to show the equality between actions

$$\begin{aligned}\mathsf{perm}_{\Delta,\Delta_2,\Delta'} \cdot (\mathbf{id}_{\otimes|\Delta_2|} \otimes (\mathsf{perm}_{\Delta',\Delta'_1,\Delta'_2} \cdot (w_1)^{\ddagger}_{\Gamma;\Delta'_1} \otimes (w_2)^{\ddagger}_{\Gamma;\Delta'_1})) \cdot (v)^{\ddagger}_{\Gamma;\Delta_2,\vec{x}:\vec{m},\vec{y}:\vec{m}'} =\\ \mathsf{perm}_{\Delta,\Delta'',\Delta'_1} \cdot (\mathbf{id}_{|\Delta'|} \otimes (w_1)^{\ddagger}_{\Gamma;\Delta'_1}) \cdot \mathsf{perm}_{\Delta'',(\Delta_2,\vec{x}:\vec{m}),\Delta'_2}\\ \cdot (\mathbf{id}_{|\Delta_2,\vec{x}:\vec{m}|} \otimes (w_2)^{\ddagger}_{\Gamma;\Delta'_2}) \cdot (v)^{\ddagger}_{\Gamma;\Delta_2,\vec{x}:\vec{m},\vec{y}:\vec{m}'}\end{aligned}$$

where Δ'' is the unique sequence of typings satisfying $\Delta = \Delta''\#\Delta'_1$. This is true by virtue of tensor and composition properties, and the definition of perm.

2. Considering the $\otimes-\eta$ equality, we need to show the equality between terms:

$$\mathsf{perm}_{\Delta,\_,\Delta} \cdot (\mathbf{id}_\varepsilon \otimes (v)^\ddagger_{\Gamma;\Delta}) \cdot \mathbf{id}_{\otimes\vec{m}} = (v)^\ddagger_{\Gamma;\Delta}$$

   which is clear since $\mathsf{perm}_{\Delta,\_,\Delta} = \mathbf{id}_{|\Delta|}$.

3. Considering the $\otimes - 1$ equality, the appropriate equality holds by virtue of the uniqueness of the multi-permutation action mperm.

4. Considering the $\otimes - 2$ equality, we need to show the equality on actions;

$$\mathsf{mperm}_{\Delta,\Delta} \cdot (v)^\ddagger_{\Gamma;\Delta} = (v)^\ddagger_{\Gamma;\Delta}$$

   which clearly holds since by uniqueness $\mathsf{mperm}_{\Delta,\Delta}$ must be the identity.

5. Considering the $\mathsf{let}\ - n$ equality, we need to show the following equality on actions.

$$\begin{aligned}\mathsf{perm}_{\Delta,\Delta_1,\Delta'} \cdot (\mathbf{id}_{|\Delta_1|} \otimes (\mathsf{perm}_{\Delta',\Delta'_2,\Delta'_1} \cdot (\mathbf{id}_{|\Delta'_2|} \otimes (v')^\ddagger_{\Gamma;\Delta'_1} \cdot (w')^\ddagger_{\Gamma;\Delta'_2,\vec{x}:\vec{m}})))&\\ \cdot (w)^\ddagger_{\Gamma;\Delta_1,y:m'} =&\\ \mathsf{perm}_{\Delta,\Delta'',\Delta'_1} \cdot (\mathbf{id}_{|\Delta''|} \otimes (v')^\ddagger_{\Gamma;\Delta'_1}) \cdot \mathsf{perm}_{(\Delta'',\vec{x}:\vec{m}),\Delta_1,(\Delta'_2,\vec{x}:\vec{m})}&\\ \cdot (\mathbf{id}_{|\Delta_1|} \otimes (w')^\ddagger_{\Gamma;\Delta'_2,\vec{x}:\vec{m}}) \cdot (w)^\ddagger_{\Gamma;\Delta_1,y:m'}&\end{aligned}$$

   where Δ'' is the unique sequence of typings satisfying $\Delta = \Delta''\#\Delta'_1$. This clearly holds using the substitution lemma 6.3.6.

Hence we have completed the proof. □

The Translations are Inverse

We can now show that these two translations make an inverse pair.

Lemma 6.3.8 (Inverse Pair)
The translations $(\_)^\dagger$ and $(\_)^\ddagger$ are inverse up to provable equality, in the sense that for any action $a{:}m \to n$ of $\mathsf{AC}(\mathbb{K})$ having free names $\vec{x}$ where $m = l_1 \dots l_r$, $\vec{y} = y_1 \dots y_r$ and $|\vec{x}| = \vec{p}$,

$$((a)^\dagger_{\vec{y}})^\ddagger_{\vec{x}:\vec{p};\vec{y}:\vec{l}} = a$$

holds in the equational theory AC, and for any $\Gamma;\Delta$-term v of $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$ we have the equality judgement:

$$\Gamma;\Delta \vdash ((v)^\ddagger_{\Gamma;\Delta})^\dagger_{\vec{y}} = v{:}n$$

where n is the unique type ascribed to v in the context $\Gamma;\Delta$ and $\Delta = \vec{y}{:}\vec{m}$.

Proof We prove this as before by induction over the structure of actions and terms. Assume in the following that $\Gamma = \vec{x}:\vec{p}$. Firstly, considering actions $a : m \to n$, we have:

Identity In this case we have that:

$$((\mathbf{id}_{\vec{l}})^{\dagger}_{\vec{y}})^{\ddagger}_{\_;\vec{y}:\vec{l}} = (\otimes\vec{y})^{\ddagger}_{\_;\vec{y}:\vec{l}} = \mathsf{mperm}_{\vec{y}:\vec{l},(y_1:l_1)\ldots(y_r:l_r)} \cdot \mathbf{id}_{l_1} \otimes \ldots \otimes \mathbf{id}_{l_r}$$

but this is clearly equal to $\mathbf{id}_{\vec{l}}$.

Composition In this case, we have that:

$$\begin{aligned}((a \cdot b)^{\dagger}_{\vec{y}})^{\ddagger}_{\Gamma;\vec{y}:\vec{l}} &= (\mathsf{let}\ \otimes\vec{y}'\ \mathsf{be}\ (a)^{\dagger}_{\vec{y}}\ \mathsf{in}\ (b)^{\dagger}_{\vec{y}'})^{\ddagger}_{\Gamma;\vec{y}:\vec{l}} \\ &= \mathsf{perm}_{\vec{y}:\vec{l},\_,\vec{y}:\vec{l}} \cdot ((a)^{\dagger}_{\vec{y}})^{\ddagger}_{\Gamma;\vec{y}:\vec{l}} \cdot ((b)^{\dagger}_{\vec{y}'})^{\ddagger}_{\Gamma;\vec{y}':\vec{l}'}\end{aligned}$$

but by induction we have that this is equal to $a \cdot b$.

Tensor In this case we have that:

$$\begin{aligned}((a \otimes b)^{\dagger}_{\vec{y}_1\vec{y}_2})^{\ddagger}_{\Gamma:\vec{y}_1:\vec{l}_1,\vec{y}_2:\vec{l}_2} &= ((a)^{\dagger}_{\vec{y}_1} \otimes (b)^{\dagger}_{\vec{y}_2})^{\ddagger}_{\Gamma:\vec{y}_1:\vec{l}_1,\vec{y}_2:\vec{l}_2} \\ &= \mathsf{perm}_{(\vec{l}_1\vec{l}_2),\vec{l}_1,\vec{l}_2} \cdot ((a)^{\dagger}_{\vec{y}_1})^{\ddagger}_{\Gamma:\vec{y}_1:\vec{l}_1} \otimes ((a)^{\dagger}_{\vec{y}_2})^{\ddagger}_{\Gamma;\vec{y}:\vec{l}_2}\end{aligned}$$

but again by induction this is the identity.

Permutation In this case we have that:

$$\begin{aligned}((\mathbf{p}_{\vec{l}_1,\vec{l}_2})^{\dagger}_{\vec{y}_1\vec{y}_2})^{\ddagger}_{\_;\vec{y}_2:\vec{l}_2,\vec{y}_1:\vec{l}_1} &= (\otimes\vec{y}_2\vec{y}_1)^{\ddagger}_{\_;\vec{y}_2:\vec{l}_2,\vec{y}_1:\vec{l}_1} \\ &= \mathsf{perm}_{(\vec{l}_1\vec{l}_2),\vec{l}_2,\vec{l}_1} \cdot (\mathbf{id}_{\vec{l}_2} \otimes \mathbf{id}_{\vec{l}_1})\end{aligned}$$

but by uniqueness $\mathsf{perm}_{(\vec{l}_1\vec{l}_2),\vec{l}_2,\vec{l}_1}$ is just $\mathbf{p}_{\vec{l}_1,\vec{l}_2}$.

Abstraction In this case we have that:

$$\begin{aligned}(((x'^{q})a)^{\dagger}_{y'\vec{y}})^{\ddagger}_{\Gamma;y':q,\vec{y}:\vec{l}} &= (\mathsf{let}\ x'\ \mathsf{be}\ y'\ \mathsf{in}\ (a)^{\dagger}_{\vec{y}})^{\ddagger}_{\Gamma;y':q,\vec{y}:\vec{l}} \\ &= \mathsf{perm}_{(y':q,\vec{y}:\vec{l})(y':q),\vec{y}:\vec{l}} \cdot (\mathbf{id}_q \otimes \mathbf{id}_{\vec{l}})der : \cdot (x')((a)^{\dagger}_{\vec{y}})^{\ddagger}_{\Gamma,x':q;\vec{y}:\vec{l}}\end{aligned}$$

but by induction this is equal to $(x')a$.

Datum In this case we have that:

$$((\langle x^p\rangle)^{\dagger}_{\varepsilon})^{\ddagger}_{x:p;\_} = (x)^{\ddagger}_{x:p;\_} = \langle x\rangle$$

Control In this case we have:

$$\begin{aligned}((\mathsf{K}(a_1 \ldots a_r))^{\dagger}_{\vec{y}'})^{\ddagger}_{\Gamma;\vec{y}':\vec{l}'} &= (\mathsf{K}((\vec{y}_1)(a_1)^{\dagger}_{\vec{y}_1} \ldots (\vec{y}_r)(a_r)^{\dagger}_{\vec{y}_r}, \vec{y}'))^{\ddagger}_{\Gamma;\vec{y}':\vec{l}'} \\ &= \mathsf{mperm}_{(\vec{y}':\vec{l}),y'_1:l'_1 \ldots y'_s:l'_s} \cdot (\mathbf{id}_{l_1} \otimes \ldots \otimes \mathbf{id}_{l_s}) \cdot \\ &\qquad \mathsf{K}(((a_1)^{\dagger}_{\vec{y}_1})^{\ddagger}_{\Gamma;\vec{y}_1:\vec{l}_1} \ldots ((a_r)^{\dagger}_{\vec{y}_r})^{\ddagger}_{\Gamma;\vec{y}_r:\vec{l}_r})\end{aligned}$$

But again this is by induction just $\mathsf{K}(a_1 \ldots a_r)$.

For the second part, we prove the result over the structure of the unique derivation of the $\Gamma; \Delta$ term.

I-Ax In this case we have:

$$((x)^{\ddagger}_{\Gamma,x:p,\Gamma';\_})^{\dagger}_{\varepsilon} = (\langle x^p \rangle)^{\dagger}_{\varepsilon} = x$$

and so the appropriate equality judgement is easy by reflexivity.

L-Ax In this case we have:

$$((x)^{\ddagger}_{\Gamma;x:l})^{\dagger}_{x} = (\mathbf{id}_l)^{\dagger}_{x} = x$$

and so the appropriate equality judgement is easy by reflexivity.

F-Rule In this case we have:

$$\begin{aligned}&((\mathsf{let}\ x^p\ \mathsf{be}\ v\ \mathsf{in}\ w)^{\ddagger}_{\Gamma;\Delta})^{\dagger}_{\vec{y}_1\vec{y}_2} \\ &= (\mathsf{perm}_{\Delta,\Delta_2,\Delta_1} \cdot (\mathbf{id}_{|\Delta_2|} \otimes (v)^{\ddagger}_{\Gamma;\Delta_1}) \cdot (x)(w)^{\ddagger}_{\Gamma,x:p;\Delta_2})^{\dagger}_{\vec{y}_1\vec{y}_2} \\ &= \mathsf{let}\ \otimes\vec{y}_2'\vec{y}_1'\ \mathsf{be}\ \vec{y}\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_2'', z\ \mathsf{be}\ \otimes\vec{y}_2' \otimes ((v)^{\ddagger}_{\Gamma;\Delta_1})^{\dagger}_{\vec{y}_1'}\ \mathsf{in}\ \mathsf{let}\ x\ \mathsf{be}\ z\ \mathsf{in}\ ((w)^{\ddagger}_{\Gamma;\Delta_2})^{\dagger}_{\vec{y}_2''}\end{aligned}$$

and the required equality judgement is obtainable by induction using $\otimes - \eta$, $\otimes - \beta$ and cc equalities.

Control In this case we have:

$$\begin{aligned}&((\mathsf{K}((\vec{y}_1{:}\vec{l}_1)v_1 \ldots (\vec{y}_2{:}\vec{l}_2)v_2, \vec{w}))^{\ddagger}_{\Gamma;\Delta'})^{\dagger}_{\vec{y}'} \\ &= (\mathsf{mperm}_{\Delta',\Delta'_1 \ldots \Delta'_s} \cdot ((w_1)^{\ddagger}_{\Gamma;\Delta'_1} \otimes \ldots \otimes (w_s)^{\ddagger}_{\Gamma;\Delta'_s}) \cdot \mathsf{K}((v_1)^{\ddagger}_{\Gamma;\Delta_1} \ldots (v_r)^{\ddagger}_{\Gamma;\Delta_r}))^{\dagger}_{\vec{y}'} \\ &= \mathsf{let}\ \otimes\vec{z}_1 \ldots \vec{z}_s\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{z}'\ \mathsf{be}\ ((w_1)^{\ddagger}_{\Gamma;\Delta'_1})^{\dagger}_{\vec{y}_1'} \otimes \ldots \otimes ((w_s)^{\ddagger}_{\Gamma;\Delta'_s})^{\dagger}_{\vec{y}_s'} \\ &\qquad \mathsf{in}\ \mathsf{K}((\vec{y}_1)((v_1)^{\ddagger}_{\Gamma;\Delta_1})^{\dagger}_{\vec{y}_1} \ldots ((\vec{y}_r)(v_r)^{\ddagger}_{\Gamma;\Delta_r})^{\dagger}_{\vec{y}_r}, \vec{z})\end{aligned}$$

and the required equality can again be obtained by induction using $\otimes - \beta$, $\otimes - \eta$ and cc equalities.

Tensor-Intro In this case, we have:

$$\begin{aligned}((\otimes(v_1 \dots v_r))^{\ddagger}_{\Gamma;\Delta})^{\dagger}_{\vec{y}'} &= (\mathsf{mperm}_{\Delta,\Delta_1\dots\Delta_r} \cdot ((v_1)^{\ddagger}_{\Gamma;\Delta_1} \otimes \dots \otimes (v_r)^{\ddagger}_{\Gamma;\Delta_r}))^{\dagger}_{\vec{y}'} \\ &= \mathsf{let}\ \otimes\vec{y}_1 \dots \vec{y}_r\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ ((v_1)^{\ddagger}_{\Gamma;\Delta_1})^{\dagger}_{\vec{y}_1} \otimes \dots ((v_r)^{\ddagger}_{\Gamma;\Delta_r})^{\dagger}_{\vec{y}_r}\end{aligned}$$

and by induction using $\otimes - \beta$ and our α-conversion result lemma 6.3.3 the required equality can be deduced.

Tensor-Elim In this case, we have:

$$\begin{aligned}&((\mathsf{let}\ \otimes\vec{y}{:}\vec{m}\ \mathsf{be}\ v\ \mathsf{in}\ w)^{\ddagger}_{\Gamma;\Delta'})^{\dagger}_{\vec{y}'} \\ &= (\mathsf{perm}_{\Delta',\Delta_2,\Delta_1} \cdot (\mathbf{id}_{\otimes|\Delta_2|} \otimes (v)^{\ddagger}_{\Gamma;\Delta_1}) \cdot (w)^{\ddagger}_{\Gamma;\Delta_2,\vec{y}:\vec{m}})^{\dagger}_{\vec{y}'} \\ &= \mathsf{let}\ \otimes\vec{y}_1\vec{y}_2\ \mathsf{be}\ \vec{y}'\ \mathsf{in}\ \mathsf{let}\ \otimes\vec{y}_2'\vec{y}\ \mathsf{be}\ \otimes\vec{y}_2 \otimes ((v)^{\ddagger}_{\Gamma;\Delta_1})^{\dagger}_{\vec{y}_1}\ \mathsf{in}\ ((w)^{\ddagger}_{\Gamma;\Delta_2,\vec{y}:\vec{m}})^{\dagger}_{\vec{y}_2\vec{y}}\end{aligned}$$

and the required equality can again be obtained by induction using $\otimes - \beta$, $\otimes - \eta$ and *cc* equalities.

6.4 Extensions

We now present three extensions of action calculi, which add functional or higher-order behaviour to the basic action calculi. Essentially, these work by adding certain higher-order operators to the standard definition of action calculi, and certain axiomatic equalities on them to the theory of action calculi. This idea was first introduced by Milner in [Mil94b], where he extends the action calculus by adding an abstraction operator and an application operator similar to the abstraction and application of the λ-calculus. The first higher-order action calculus we present is just this one as amended by Hasegawa [HG], generalised to allow linear and intuitionistic primes in the same way as we have generalised Milner's action calculi.

Definition 6.4.1 (Higher-Order Action Calculus)
Given a signature $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$, the *higher-order action calculus* $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$ is given by extending the definition of generalised action calculi as follows:

1. the derived sets $\mathsf{P}^{\Rightarrow}_I$ and $\mathsf{P}^{\Rightarrow}_L$ of primes and the set $\mathsf{M}^{\Rightarrow}$ of arities are constructed from the following abstract grammars:

set of intuitionistic primes $\mathsf{P}^{\Rightarrow}_I$	$p ::= p \in \mathsf{P}_I \mid m \Rightarrow m$
set of linear primes $\mathsf{P}^{\Rightarrow}_L$	$l ::= l \in \mathsf{P}_L \mid p \in \mathsf{P}^{\Rightarrow}_I$
set of arities $\mathsf{M}^{\Rightarrow}$	$m ::= l \in \mathsf{P}^{\Rightarrow}_L \mid m \otimes m \mid \varepsilon$

2. the terms of $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$ are generated by the rules in definition 6.1.3, plus the rules

$$\frac{a : m \to n}{\boldsymbol{\lambda} a : \varepsilon \to m \Rightarrow n} \qquad\qquad \mathbf{ap} : (m \Rightarrow n) \otimes m \to n$$

3. the equational theory of $\mathsf{AC}_{\Rightarrow}$ is that generated from the axioms in definition 6.1.4, plus the axioms

$$((\boldsymbol{\lambda} a) \otimes \mathbf{id}_m) \cdot \mathbf{ap} = a \tag{β}$$

$$\boldsymbol{\lambda}((\langle x \rangle \otimes \mathbf{id}_m) \cdot \mathbf{ap}) = \langle x \rangle \tag{η_V}$$

$$(\boldsymbol{\lambda}(a) \otimes \mathbf{id}) \cdot (x)b = b\{\boldsymbol{\lambda}(a)/\langle x \rangle\} \tag{σ}$$

The action calculus $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$ is the quotient of the terms by the equality.

Extensionality The amendment made by Hasegawa [HG] to Milner's original definition is the addition of a η_V-equality similar to that given here in the case of Milner's action calculi. If we say that the η-equality for an action $a : \varepsilon \to (n \Rightarrow m)$ is the equality

$$\boldsymbol{\lambda}((a \otimes \mathbf{id}_m) \cdot \mathbf{ap}) = a$$

then in $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$ we can derive η-equalities for any copyable action of $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$, and in fact the converse is true. We can read this as saying that the functional structure is extensional over copyable actions, which we might regard as the intuitionistic subsystem of the action calculus. We could ask what would happen if we replaced η_V by the η-equality for all actions a of the appropriate arities. In [Mil94b], Milner points out that this addition collapses the structure of action calculi by making it simply that of a cartesian closed category. We can see this in the syntax since if we add the η rule in general, then all actions of the appropriate arity would become copyable and hence fall into the intuitionistic subsystem, so that this subsystem would then be the whole system.

We now present two systems based on a linear decomposition of the arrow constructor in the previous system. The first of these has just the $\boxtimes$ operator, which corresponds to the ! of linear logic, and which acts as a 'coding' operator in this setting. This was discovered independently by the author and Masahito Hasegawa.

Definition 6.4.2 (Action Calculus with Code)
Given a signature $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$, the *action calculus with code* $\mathsf{AC}_{\boxtimes}(\mathbb{K})$ is given by extending the definition of action calculi as follows:

1. the sets $\mathsf{P}_I^{\boxtimes}$ and $\mathsf{P}_L^{\boxtimes}$ of intuitionistic and linear primes and the set $\mathsf{M}^{\boxtimes}$ of arities are constructed from the following abstract grammars:

$$\begin{array}{ll} \text{set of intuitionistic primes } \mathsf{P}_I^{\boxtimes} & p ::= p \in \mathsf{P}_I \mid \boxtimes(m) \\ \text{set of linear primes } \mathsf{P}_L^{\boxtimes} & l ::= l \in \mathsf{P}_L \mid p \in \mathsf{P}_I^{\boxtimes} \\ \text{set of arities } \mathsf{M}^{\boxtimes} & m ::= l \in \mathsf{P}_L^{\boxtimes} \mid m \otimes m \mid \varepsilon \end{array}$$

2. the terms of $\mathsf{AC}_{\boxtimes}(\mathbb{K})$ are generated by the rules in definition 6.1.3, plus the rules

$$\frac{a : \varepsilon \to m}{\mathsf{code}(a) : \varepsilon \to \boxtimes(m)} \qquad \mathsf{decode} : \boxtimes(m) \to m$$

3. the equational theory of $\mathsf{AC}_{\boxtimes}$ is that generated from the axioms in definition 6.1.4, plus the axioms

$$\mathsf{code}(a) \cdot \mathsf{decode} = a \tag{β}$$

$$\mathsf{code}(\langle x \rangle \cdot \mathsf{decode}) = \langle x \rangle \tag{η}$$

$$(\mathsf{code}(a) \otimes \mathbf{id}) \cdot (x)b = b\{\mathsf{code}(a)/\langle x \rangle\} \tag{σ}$$

The action calculus $\mathrm{AC}_{\boxtimes}(\mathcal{K})$ is the quotient of the terms by the equality.

Secondly, we introduce a system which extends $\mathsf{AC}_{\boxtimes}(\mathbb{K})$ by adding the linear function type, with abstraction and application operators.

Definition 6.4.3 (Linear Higher-Order Action Calculus)
Given a signature $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$, the *linear* higher-order action calculus $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$ is given by extending the definition of action calculi as follows:

1. the sets P_I and P_L of intuitionistic and linear primes and the set $\mathsf{M}^{\boxtimes,\multimap}$ of arities are constructed as follows:

$$\begin{array}{ll} \text{set of intuitionistic primes } \mathsf{P}_I^{\boxtimes,\multimap} & p ::= p \in \mathsf{P}_I \mid \boxtimes(m) \\ \text{set of linear primes } \mathsf{P}_L^{\boxtimes,\multimap} & l ::= l \in \mathsf{P}_L \mid p \in \mathsf{P}_I^{\boxtimes,\multimap} \mid l \multimap n \\ \text{set of arities } M^{\boxtimes,\multimap} & m ::= l \in \mathsf{P}_L^{\boxtimes,\multimap} \mid m \otimes m \mid \varepsilon \end{array}$$

2. the terms of $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$ are generated by the rules in definition 6.1.3 and definition 6.4.2, plus the rules

$$\frac{a : m \otimes l \to n}{\boldsymbol{\lambda}_L^l a : m \to (l \multimap n)} \text{ where } l \in \mathsf{P}_L \qquad \mathbf{ap}_L^l : (l \multimap n) \otimes l \to n$$

3. the equational theory $\mathsf{AC}_{\boxtimes,\multimap}$ is the set of equations upon terms generated from the axioms in definitions 6.1.4 and 6.4.2, plus the axioms

$$((\boldsymbol{\lambda}^l_L a)\otimes \mathbf{id}_p)\cdot \mathbf{ap}^l_L = a \qquad (\beta)$$

$$\boldsymbol{\lambda}^l_L((a\otimes \mathbf{id}_p)\cdot \mathbf{ap}^l_L) = a \qquad (\eta)$$

$$\boldsymbol{\lambda}^l_L((x)a) = (x)(\boldsymbol{\lambda}^l_L a) \text{ where } a : m\otimes p \to n \qquad \text{(cc)}$$

$$\boldsymbol{\lambda}^l_L(a\otimes \mathbf{id}_p\cdot b) = a\cdot(\boldsymbol{\lambda}^l_L b) \qquad \text{(cc)}$$

$$\text{where } a : m \to n \text{ and } b : n\otimes p \to n'$$

The action calculus $\mathrm{AC}_{\boxtimes,\multimap}(\mathcal{K})$ is the quotient of the terms by the equality.

Extensionality Revisited We note that in contrast to the situation with the extended action calculus $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$, the η-equality holds in the system $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$ for all actions of the appropriate arity. However, the structure does not collapse to that of a CCC in the case of $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$ since it is no longer possible to prove that any action for which the η-equality holds is copyable. Hence this theory provides an account of extensional functions at a linear level.

We can extend the definition of the linear arrow type to arbitrary arities. Given an arity $m = l_1 \ldots l_r$, define on arities $m \multimap n = l_1 \multimap (\ldots(l_r \multimap n)..)$, and on actions $\boldsymbol{\lambda}^m_L(a) = \boldsymbol{\lambda}^{l_1}_L(\ldots\boldsymbol{\lambda}^{l_r}_L(a)..)$ and $\mathbf{ap}^m_L = (\mathbf{ap}^{l_1}_L \otimes \mathbf{id}_{l_2}\otimes \ldots l_r)\cdot\ldots\cdot\mathbf{ap}^{l_r}_L$. Note that this definition makes $\varepsilon \multimap m = m$, and correspondingly on term constructs makes $\boldsymbol{\lambda}^{\varepsilon}_L(a) = a$ and $\mathbf{ap}^{\varepsilon}_L = \mathbf{id}$.

Relating the Extensions

Having presented these extensions, we can give some relationships between them. It is clear firstly that there are trivial embeddings $\iota_1 : \mathsf{AC}(\mathbb{K}) \to \mathsf{AC}_{\boxtimes}(\mathbb{K})$ and $\iota_2 : \mathsf{AC}_{\boxtimes}(\mathbb{K}) \to \mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$. However, there also exist translations $\iota_3 : \mathsf{AC}_{\boxtimes}(\mathbb{K}) \to \mathsf{AC}_{\Rightarrow}(\mathbb{K})$ and $\iota_4 : \mathsf{AC}_{\Rightarrow}(\mathbb{K}) \to \mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$.

Definition 6.4.4 (The Translation ι_3)
Define the translation $\iota_3 : \mathsf{AC}_{\boxtimes}(\mathbb{K}) \to \mathsf{AC}_{\Rightarrow}(\mathbb{K})$ as the obvious embedding on shared constructs, augmented with:

$$\iota_3(\boxtimes(m)) = (\varepsilon \Rightarrow m)$$

$$\iota_3(\mathsf{code}(a)) = \boldsymbol{\lambda}(\iota_3(a))$$

$$\iota_3(\mathsf{decode}) = \mathbf{ap}$$

Definition 6.4.5 (The Translation ι_4)
Define the translation $\iota_4 : \mathsf{AC}_{\Rightarrow}(\mathbb{K}) \to \mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$ as the obvious embedding on shared constructs, augmented with:

$$\iota_4(m \Rightarrow n) = \boxtimes(m \multimap n)$$

$$\begin{aligned} \iota_4(\boldsymbol{\lambda}(a)) &= \mathsf{code}(\boldsymbol{\lambda}^m_L(\iota_4(a))) \text{ where } a : m \to n \\ \iota_4(\mathbf{ap}) &= \mathsf{decode} \otimes \mathbf{id} \cdot \mathbf{ap} \end{aligned}$$

$$\begin{tikzcd}
& \mathsf{AC}(\mathbb{K}) \arrow[d, "\iota_1"] \\
& \mathsf{AC}_{\boxtimes}(\mathbb{K}) \arrow[dl, "\iota_3"'] \arrow[d, "\iota_2"] \\
\mathsf{AC}_{\Rightarrow}(\mathbb{K}) \arrow[r, "\iota_4"'] & \mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})
\end{tikzcd}$$

Figure 6.1: Translations between $\mathsf{AC}(\mathbb{K})$ and its extensions

We summarise the translations in figure 6.1. It can easily be shown that the triangle at the bottom of this diagram commutes up to equality in $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$.

Extending $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$

We now extend our generalised linear type-theory $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$ to provide type-theories corresponding to the extended action calculi defined previously. We introduce some notation for these definitions, where E denotes any extension from $\boxtimes, \to$ or $\boxtimes, \multimap$:

- When we say that an operator set $\mathcal{O}^{\mathsf{A}^E}_L$ *contains all control operators* we will mean that it contains every operator O_{K} of $\mathcal{O}^{\mathsf{A}}_L$.
- When we say that an operator set $\mathcal{O}^{\mathsf{A}^E}_L$ *contains all* $\otimes^R_{m_1 \ldots m_r}$ *operators* we will mean that it contains instances of the operator $\otimes^R_{m_1 \ldots m_r}$ for all $m_1 \ldots m_r \in \mathsf{M}^{\mathsf{A}^E}_L$.

- When we say that an operator set $\mathcal{O}_L^{\mathsf{A}^E}$ *contains all* $\otimes^L_{m_1 \ldots m_r}$ *operators* we will mean that it contains instances of every operator in the output-parameterised set $\otimes^L_{m_1 \ldots m_r}$ for all $m_1 \ldots m_r \in \mathsf{M}_L^{\mathsf{A}^E}$.

- When we say that an axiom set $\mathbb{A}^{\mathsf{A}^E}$ *contains all* l*-equalities*, for some identifying label l, we will mean that contains every well-formed equality having the form of the equality l *in the typing system* $\mathsf{Lin}^{\mathsf{A}}(\mathbb{O}^E)$.

Definition 6.4.6 (The Type Theory $\mathrm{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$)
We define the type theory $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$ over a signature $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$ as the type theory $\mathsf{LNat}(\mathbb{O}^{\mathsf{A}^{\Rightarrow}}, \mathbb{A}^{\mathsf{A}^{\Rightarrow}})$ where $\mathbb{O}^{\mathsf{A}^{\Rightarrow}} = (\mathsf{M}_I^{\mathsf{A}^{\Rightarrow}}, \mathsf{M}_I^{\mathsf{A}^{\Rightarrow}}, \mathcal{O}_I^{\mathsf{A}^{\Rightarrow}}, \mathcal{O}_L^{\mathsf{A}^{\Rightarrow}})$ and we define these objects as follows:

- The set $\mathsf{M}_I^{\mathsf{A}^{\Rightarrow}}$ is the set of singleton sequences of elements of $\mathsf{P}_I^{\Rightarrow}$.

- The set $\mathsf{M}_L^{\mathsf{A}^{\Rightarrow}}$ is the set of sequences of elements of $\mathsf{P}_L^{\Rightarrow}$.

- The set $\mathcal{O}_L^{\mathsf{A}^{\Rightarrow}}$ contains all control operators, all $\otimes^R_{m_1 \ldots m_r}$ operators and all $\otimes^L_{m_1 \ldots m_r}$ operators and also:

  1. For each m and n in $\mathsf{M}_L^{\mathsf{A}^{\Rightarrow}}$, an operator $\boldsymbol{\lambda}_{m,n}$ of arity:
  $$\frac{(;m)n \quad ;}{(;)m \Rightarrow n}$$
  We will write the general operator application $\boldsymbol{\lambda}_{m,n}((;x)v;)(;)$ in the form $\lambda x{:}m.v$.

  2. For each m and n in $\mathsf{M}_L^{\mathsf{A}^{\boxtimes}}$ an operator **ap** of arity:
  $$\frac{;}{(;(m \Rightarrow n), m)n}$$
  We will write $\mathsf{ap}(;)(;vw)$ in the familiar way as vw.

- The set $\mathcal{O}_I^{\mathsf{A}^{\Rightarrow}}$ is the subset of $\mathcal{O}_L^{\mathsf{A}^{\Rightarrow}}$ containing for each m and n the operator $\boldsymbol{\lambda}_{m,n}$.

- The set $\mathbb{A}^{\mathsf{A}^{\Rightarrow}}$ contains all $\otimes - \beta, \otimes - \eta, \otimes - 1$ and $\otimes - 2$ equalities and also:

  1. All well formed equality judgements of the form:
  $$(\boldsymbol{\lambda} - \beta) \;\; \Gamma; \Delta \vdash (\lambda x.v)w = v\{w/x\}{:}m$$

  2. All well formed equality judgements of the form:
  $$(\boldsymbol{\lambda} - \eta_V) \;\; \Gamma, y{:}m \Rightarrow n; \_ \vdash \lambda x.(yx) = y{:}m \Rightarrow n$$

3. All the equality judgements in the sets $\mathsf{ONat}(\mathbb{O}^{\mathsf{A}^{\Rightarrow}}, \otimes^{L}_{m_1 \ldots m_r})$, for any $m_1 \ldots m_r$.

Definition 6.4.7 (The Type Theory $\mathrm{Lin}^{\mathsf{A}^{\boxtimes}}(\mathbb{K})$)
We define the type theory $\mathsf{Lin}^{\mathsf{A}^{\boxtimes}}(\mathbb{K})$ over a signature $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$ as the type theory $\mathsf{LNat}(\mathbb{O}^{\mathsf{A}^{\boxtimes}}, \mathbb{A}^{\mathsf{A}^{\boxtimes}})$ where $\mathbb{O}^{\mathsf{A}^{\boxtimes}} = (\mathsf{M}_I^{\mathsf{A}^{\boxtimes}}, \mathsf{M}_I^{\mathsf{A}^{\boxtimes}}, \mathcal{O}_I^{\mathsf{A}^{\boxtimes}}, \mathcal{O}_L^{\mathsf{A}^{\boxtimes}})$ and we define these objects as follows:

- The set $\mathsf{M}_I^{\mathsf{A}^{\boxtimes}}$ is the set of singleton sequences of elements of $\mathsf{P}_I^{\boxtimes}$.
- The set $\mathsf{M}_L^{\mathsf{A}^{\boxtimes}}$ is the set of sequences of elements of $\mathsf{P}_L^{\boxtimes}$.
- The set $\mathcal{O}_L^{\mathsf{A}^{\boxtimes}}$ contains all control operators, all $\otimes^{R}_{m_1 \ldots m_r}$ operators and all $\otimes^{L}_{m_1 \ldots m_r}$ operators and also:

  1. For each $m \in \mathsf{M}_L^{\mathsf{A}^{\boxtimes}}$ an operator code_m of arity:

  $$\frac{(;)m \quad ;}{(;)\boxtimes(m)}$$

  We will write the general operator application $\mathsf{code}_m((;)v;)(;)$ as $\mathsf{code}(v)$.

  2. For each $m \in \mathsf{M}_L^{\mathsf{A}^{\boxtimes}}$ an operator decode_m of arity:

  $$\frac{;}{(;\boxtimes(m))m}$$

  We will write the general operator application $\mathsf{decode}_m(;)(;v)$ as $\mathsf{decode}(v)$.

- The set $\mathcal{O}_I^{\mathsf{A}^{\boxtimes}}$ is the subset of $\mathcal{O}_L^{\mathsf{A}^{\boxtimes}}$ containing for each m the code_m operator.
- The set $\mathbb{A}^{\mathsf{A}^{\boxtimes}}$ contains all $\otimes-\beta, \otimes-\eta, \otimes-1$ and $\otimes-2$ equalities and also:

  1. All well formed equality judgements of the form:

  $$(\boxtimes-\beta) \;\; \Gamma; \_ \vdash \mathsf{decode}(\mathsf{code}(v)) = v{:}m$$

  2. All well formed equality judgements of the form:

  $$(\boxtimes-\eta) \;\; \Gamma; \_ \vdash \mathsf{code}(\mathsf{decode}(x)) = x{:}\boxtimes(m)$$

  3. All equality judgements in the sets $\mathsf{ONat}(\mathbb{O}^{\mathsf{A}^{\boxtimes}}, \otimes^{L}_{m_1 \ldots m_r})$, for all $m_1 \ldots m_r$.

Definition 6.4.8 (The Type Theory $\mathrm{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K})$)
We define the type theory $\mathsf{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K})$ over a signature $\mathbb{K} = (\mathsf{P}_I, \mathsf{P}_L, \mathcal{K})$ as the type theory $\mathsf{LNat}(\mathbb{O}^{\mathsf{A}^{\boxtimes,\multimap}}, \mathbb{A}^{\mathsf{A}^{\boxtimes,\multimap}})$ where $\mathbb{O}^{\mathsf{A}^{\boxtimes,\multimap}} = (\mathsf{M}_I^{\mathsf{A}^{\boxtimes,\multimap}}, \mathsf{M}_I^{\mathsf{A}^{\boxtimes,\multimap}}, \mathcal{O}_I^{\mathsf{A}^{\boxtimes,\multimap}}, \mathcal{O}_L^{\mathsf{A}^{\boxtimes,\multimap}})$ and we define these objects as follows:

- The set $\mathsf{M}_I^{\mathsf{A}^{\boxtimes,\multimap}}$ is the set of singleton sequences of elements of $\mathsf{P}_I^{\boxtimes,\multimap}$.
- The set $\mathsf{M}_L^{\mathsf{A}^{\boxtimes,\multimap}}$ is the set of sequences of elements of $\mathsf{P}_L^{\boxtimes,\multimap}$.
- The set $\mathcal{O}_L^{\mathsf{A}^{\boxtimes,\multimap}}$ contains all control operators, all $\otimes^R_{m_1\ldots m_r}$ operators and all $\otimes^L_{m_1\ldots m_r}$ operators and also:

  1. For each $m \in \mathsf{M}_L^{\mathsf{A}^{\boxtimes}}$ an operator code_m of arity:

  $$\frac{(;)m \quad ;}{(;)\boxtimes(m)}$$

  We will write the general operator application $\mathsf{code}_m((;)v;)(;)$ as $\mathsf{code}(v)$.

  2. For each $m \in \mathsf{M}_L^{\mathsf{A}^{\boxtimes}}$ an operator decode_m of arity:

  $$\frac{;}{(;\boxtimes(m))m}$$

  We will write the general operator application $\mathsf{decode}_m(;)(;v)$ as $\mathsf{decode}(v)$.

  3. For each $l \in \mathsf{P}_L^{\boxtimes,\multimap}$ and each $n \in \mathsf{M}_L^{\mathsf{A}^{\boxtimes,\multimap}}$, an operator $\boldsymbol{\lambda}^L_{l,n}$ of arity:

  $$\frac{; \quad (;l)n}{(;)l \multimap n}$$

  We will write the general operator application $\boldsymbol{\lambda}^L_{l,n}(;(;x)v)(;)$ as $\lambda^L x{:}\,l.v$.

  4. For each $l \in \mathsf{P}_L^{\mathsf{A}^{\boxtimes,\multimap}}$ and $n \in \mathsf{M}_L^{\mathsf{A}^{\boxtimes}}$ an operator $\mathsf{ap}^L_{l,n}$ of arity:

  $$\frac{;}{(;l,(l \multimap n))n}$$

  We will write $\mathsf{ap}^L_{l,n}(;)(;vw)$ as vw.

- The set $\mathcal{O}_I^{\mathsf{A}^{\boxtimes,\multimap}}$ is the subset of $\mathcal{O}_L^{\mathsf{A}^{\boxtimes,\multimap}}$ containing just the code_m operators for each $m \in \mathsf{M}_L^{\mathsf{A}^{\boxtimes,\multimap}}$.
- The set $\mathbb{A}^{\mathsf{A}^{\boxtimes,\multimap}}$ contains all $\otimes-\beta, \otimes-\eta, \otimes-1, \otimes-2, \boxtimes-\beta$ and $\boxtimes-\eta$ equalities and also:

  1. All well formed equality judgements of the form:

  $$(\lambda-\beta) \;\; \Gamma;\Delta \vdash (\lambda x.v)w = v\{w/x\}{:}m$$

  2. All well formed equality judgements of the form:

  $$(\lambda-\eta) \;\; \Gamma;\Delta \vdash \lambda x.(vx) = v{:}l \multimap n$$

3. All equality judgements in the sets $\mathsf{ONat}(\mathbb{O}^{\mathsf{A}^{\boxtimes,\multimap}}, \otimes^{L}_{m_1\ldots m_r})$ for $m_1 \ldots m_r \in \mathsf{M}_L^{\mathsf{A}^{\boxtimes,\multimap}}$.

We can see that the embeddings and translations $\iota_1 \ldots \iota_4$ induce similar embeddings and translations amongst our extended type theories. We overload notation to call these $\iota_1 \ldots \iota_4$ as well.

We can extend our functions $(\_)^{\dagger}$ and $(\_)^{\ddagger}$ to relate each extended action calculus with its corresponding type theory. Here, we just summarise all of these translations together by giving their definitions on the various arity and term constructs.

Definition 6.4.9 (The Extended Translation $(\_)^{\dagger}$)
We define the translation $(\_)^{\dagger}$ on the various extended action calculi as follows:

$$\begin{aligned}
(\vec{y}, \boldsymbol{\lambda}(a))^{\dagger} &= \lambda z.(\mathsf{let}\ \otimes\vec{y}\ \mathsf{be}\ z\ \mathsf{in}\ (z\vec{y}, a)^{\dagger}) \\
(y_1 y_2, \mathbf{ap})^{\dagger} &= y_1 y_2
\end{aligned}$$

$$\begin{aligned}
(\varepsilon, \mathsf{code}(a))^{\dagger} &= \mathsf{code}((\varepsilon, a)^{\dagger}) \\
(y, \mathsf{decode})^{\dagger} &= \mathsf{decode}(y)
\end{aligned}$$

$$\begin{aligned}
(\vec{y}, \boldsymbol{\lambda}^{l}_{L}(a))^{\dagger} &= \lambda_L z.(\vec{y}z, a)^{\dagger} \\
(y_1 y_2, \mathbf{ap}^{l}_{L})^{\dagger} &= y_1 y_2
\end{aligned}$$

Definition 6.4.10 (The Extended Translation $(\_)^{\ddagger}$)
We define the translation $(\_)^{\ddagger}$ on the various type theories corresponding to extended action calculi as follows:

$$\begin{aligned}
(\lambda x{:}m.v)^{\ddagger}_{\Gamma} &= \boldsymbol{\lambda}(v)^{\ddagger}_{\Gamma;x:m} \\
(vw)^{\ddagger}_{\Gamma;\Delta} &= \mathsf{perm}_{\Delta,\Delta_1,\Delta_2} \cdot ((v)^{\ddagger}_{\Gamma;\Delta_1} \otimes (w)^{\ddagger}_{\Gamma;\Delta_2}) \cdot \mathbf{ap}
\end{aligned}$$

$$\begin{aligned}
(\mathsf{code}(v))^{\ddagger}_{\Gamma;_} &= \mathsf{code}((v)^{\ddagger}_{\Gamma;_}) \\
(\mathsf{decode}(v))^{\ddagger}_{\Gamma;\Delta} &= (v)^{\ddagger}_{\Gamma;\Delta} \cdot \mathsf{decode}
\end{aligned}$$

$$\begin{aligned}
(\lambda x{:}l.v)^{\ddagger}_{\Gamma;\Delta} &= \boldsymbol{\lambda}^{l}_{L}(v)^{\ddagger}_{\Gamma;\Delta,x:l} \\
(vw)^{\ddagger}_{\Gamma;\Delta} &= \mathsf{perm}_{\Delta,\Delta_1,\Delta_2} \cdot ((v)^{\ddagger}_{\Gamma;\Delta_1} \otimes (w)^{\ddagger}_{\Gamma;\Delta_2}) \cdot \mathbf{ap}_{L}
\end{aligned}$$

We can now easily extend our results on the basic translations to these extensions.

Lemma 6.4.11 (The Extended Translations)
The extended translations $(\_)^\dagger$ and $(\_)^\ddagger$ make $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$ and $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$ isomorphic, they make $\mathsf{AC}_{\boxtimes}(\mathbb{K})$ and $\mathsf{Lin}^{\mathsf{A}^{\boxtimes}}(\mathbb{K})$ isomorphic, and they make $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$ and $\mathsf{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K})$ isomorphic.

6.5 Semantics

Since the theory of action calculi itself is a relatively recent development, there is less work on the semantics of action calculi than on that of ILL. A semantics was first given in the form of *control structures* [MMP95], which are symmetric monoidal categories with abstractors to represent the name-structure. A fibrational equivalent not requiring naming structure was given by Hermida and Power [HP95]. Then this was proved equivalent by Power to an alternative formulation using an adjunction [Pow96]. These models were used as the basis for the models of Gardner *et al.* [HG, BGHP97], and in this section we will use this definition. In the higher-order case, there has been less work, but definitions of models have been given [HG, BGHP97], which we will use in the following.

In this section, we will say "$\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model" to mean "output-natural-in-$\otimes^L_{m_1\ldots m_r}$-for-all-$r, m_1 \ldots m_r$ $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model".

The Basic Case

Since we are not using precisely the original definition of action calculi, we will need to recast the definition of [HG] slightly, to take account of our linear prime arities.

The *carrier* of an *action model* is a triple $(\mathcal{C}, \mathcal{S}, F)$, where $\mathcal{C}$ is a strict cartesian closed category, $\mathcal{S}$ is a symmetric monoidal closed category, and $F : \mathcal{C} \to \mathcal{S}$ is an injective-on-objects strict symmetric monoidal functor.

Definition 6.5.1 (Action Model)
Action models over an action calculus signature $\mathbb{K}$, ranged over by $\mathcal{A}\ldots$, are given by a carrier $(\mathcal{C}, \mathcal{S}, F)$ supplemented with:

- a function $[\![\_]\!]_{\mathsf{P}_I} : \mathsf{P}_I \to \mathsf{obj}(\mathcal{C})$ and a function $[\![\_]\!]_{\mathsf{P}_L} : \mathsf{P}_L \to \mathsf{obj}(\mathcal{S})$ such that for $p \in \mathsf{P}_I$, we have $[\![p]\!]_{\mathsf{P}_I} = [\![p]\!]_{\mathsf{P}_L}$ in $\mathcal{S}$,
- for each control K with arity $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$ we have a natural transformation

$$[\![K]\!]_{\mathcal{K}} : \prod_{i=1\ldots r} \mathcal{S}(F(\_) \otimes [\![m_i]\!], [\![n_i]\!]), \mathcal{S}(F(\_) \otimes [\![m]\!], [\![n]\!])$$

under the obvious interpretation of arities into $\mathcal{S}$ given by

$$[\![l_1 \ldots l_r]\!] = \otimes([\![l_1]\!] \ldots [\![l_r]\!])$$

Now this definition is very reminiscent of our definition of models for our general linear logic. We can in fact prove:

Lemma 6.5.2
Every action model gives rise to a $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model.

Proof This is easily shown; given an action model $\mathcal{A}$, we will construct a $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model $\mathcal{G}^{\mathcal{A}}$. Firstly, the carrier of $\mathcal{G}^{\mathcal{A}}$ is exactly the carrier of $\mathcal{A}$. The interpretation of intuitionistic prime arities in M_I is exactly the interpretation of P_I given by $\mathcal{A}$, since the two sets are the same, and the interpretation on elements of M_L, which are sequences of elements of P_L, is simply given by the strict tensor of the the interpretations of the elements of P_L using the interpretation function from $\mathcal{A}$. Finally, the interpretation of the operators corresponding to the controls is exactly the interpretation of the controls themselves, and the interpretation of the $\otimes^R$ and $\otimes^L$-operators is given in the obvious way using the tensor structure on the s.m.c. $\square$

We can also define action morphisms:

Definition 6.5.3 (Action Morphisms)
An *action morphism* over an action calculus signature $\mathbb{K}$, $\mathcal{F} : \mathcal{A} \to \mathcal{A}'$ between two action models $\mathcal{A}$ and $\mathcal{A}'$ having carriers $(\mathcal{C}, \mathcal{S}, F)$ and $(\mathcal{C}', \mathcal{S}', F')$ respectively is a pair $(\mathcal{F}_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}', \mathcal{F}_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}')$ of morphisms such that:

- $\mathcal{F}_{\mathcal{C}}$ is strict cartesian and $\mathcal{F}_{\mathcal{S}}$ is strict symmetric monoidal,
- the following diagram commutes:

$$\begin{tikzcd} \mathcal{C} \arrow[r, "F"] \arrow[d, "\mathcal{F}_{\mathcal{C}}"'] & \mathcal{S} \arrow[d, "\mathcal{F}_{\mathcal{S}}"] \\ \mathcal{C}' \arrow[r, "F'"] & \mathcal{S}' \end{tikzcd}$$

- $\mathcal{F}_{\mathcal{C}}([\![\_]\!]^{\mathcal{A}}_{\mathsf{P}_I}) = [\![\_]\!]^{\mathcal{A}'}_{\mathsf{P}_I} : \mathsf{P}_I \to \mathsf{obj}(\mathcal{C}')$,
- $\mathcal{F}_{\mathcal{S}}([\![\_]\!]^{\mathcal{A}}_{\mathsf{P}_L}) = [\![\_]\!]^{\mathcal{A}'}_{\mathsf{P}_L} : \mathsf{P}_L \to \mathsf{obj}(\mathcal{S}')$,

- for every control K with arity $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$ from the signature $\mathbb{K}$, and for every element X of $\mathcal{C}$ and morphisms of $\mathcal{S}$ $f_i : F(X) \otimes [\![m_i]\!] \to [\![n_i]\!]$ for $i = 1 \ldots r$, we have:

$$\mathcal{F}_{\mathcal{S}}([\![\mathsf{K}]\!]^{\mathcal{A}}_{\mathcal{K}}(X)(f_1, \ldots, f_r)) = [\![\mathsf{K}]\!]^{\mathcal{A}'}_{\mathcal{K}}(\mathcal{F}_{\mathcal{C}}(X))(\mathcal{F}_{\mathcal{S}}(f_1), \ldots, \mathcal{F}_{\mathcal{S}}(f_r))$$

Given this definition, we have:

LEMMA 6.5.4
Any action morphism $\mathcal{F}$ gives rise to a $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-morphism.

It is easy to check the necessary conditions, which are very close to those of the definition of $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-morphism.

We can now define the category $\mathsf{Cat}_{\mathsf{AC}}(\mathbb{K})$ of small action models and morphisms over a signature $\mathbb{K}$. Further, call the category of small $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-models and $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-morphisms $\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{A}}}(\mathbb{K})$. From our results, we can see that we have a functor $\mathsf{Cat}_{\mathsf{AC}}(\mathbb{K}) \to \mathsf{Cat}_{\mathsf{Lin}^{\mathsf{A}}}(\mathbb{K})$.

It is clear that a general $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model may fail to be an action model, firstly because the functor F of the carrier may well not be injective on objects. We can, however, use a result of Power and Robinson [PR94], in the intuitionistic case.

LEMMA 6.5.5
For an *intuitionistic* action calculus signature $\mathbb{K}$, any $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model gives rise to an action model.

Proof Given a $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model $\mathcal{G}$, we will construct an action model $\mathcal{A}^{\mathcal{G}}$. First note that by section 5, corollary 5.1 of [PR94] given that the carrier $\mathcal{G}$ is $(\mathcal{C}, \mathcal{S}, F)$, we have a strict symmetric monoidal category $\mathcal{S}'$, an identity-on-objects functor $F_1 : \mathcal{C} \to \mathcal{S}'$ and a fully faithful functor $F_2 : \mathcal{S}' \to \mathcal{S}$, such that $F = F_1; F_2$. These are unique up to unique isomorphism. Take the carrier of $\mathcal{A}^{\mathcal{G}}$ to be $(\mathcal{C}, \mathcal{S}', F_1)$. Then take the interpretation on intuitionistic prime arities to be $[\![\_]\!]_{\mathsf{M}_I} : \mathsf{P}_I \to \mathsf{obj}(\mathcal{C})$ of $\mathcal{G}$. Take the interpretation on linear prime arities (which because the signature is intuitionistic are precisely the same as the intuitionistic prime arities) to be given

$$[\![\_]\!]_{\mathsf{P}_L} = F_1([\![\_]\!]_{\mathsf{M}_I}) : \mathsf{P}_I \to \mathsf{obj}(\mathcal{S}')$$

Now note that using the natural transformations present in $\mathcal{S}$ that

$$[\![p_1 \otimes \ldots \otimes p_r]\!]_{\mathsf{M}_L} \simeq [\![p_1]\!] \otimes \ldots \otimes [\![p_r]\!]$$

Using these isomorphisms, and the full faithfulness of F_2, we can obtain natural transformations interpreting the controls over the carrier. $\square$

The Higher-Order Case

We now consider the case of Milner's higher-order extension of an action calculus, $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$. Again, we slightly adapt the definition given in [HG] to take account of our linear prime arities.

Definition 6.5.6 (Higher-Order Action Model)
Higher-order action models over an action calculus signature $\mathbb{K}$, again ranged over by $\mathcal{A}\ldots$, are action models over $\mathbb{K}$ such that the functor $F(\_)\otimes X : \mathcal{C} \to \mathcal{S}$ has a right adjoint $X \Rightarrow \_ : \mathcal{S} \to \mathcal{C}$, where the interpretation $\llbracket \_ \rrbracket_{\mathsf{P}_I} : \mathsf{P}_I \to \mathsf{obj}(\mathcal{C})$ is extended by saying that $\llbracket m \Rightarrow n \rrbracket_{\mathsf{P}_I} = \llbracket m \rrbracket_{\mathsf{P}_L} \Rightarrow \llbracket n \rrbracket_{\mathsf{P}_L}$.

We can now prove

Lemma 6.5.7
Each higher-order action model gives rise to a $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$-model.

Proof This is quite easy to show since we know that the higher-order action model is in fact an action model over signature $\mathbb{K}$, and hence we know that it gives rise to a $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model. However, this $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-model has the required interpretation of the new intuitionistic prime arities of $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$ by virtue of the functor $\Rightarrow$, and the abstraction and application are interpreted using natural transformations existing because of the adjunction. □

Definition 6.5.8 (Higher-Order Action Morphism)
A *higher-order action morphism* over an action calculus signature $\mathbb{K}$, $\mathcal{F} : \mathcal{A} \to \mathcal{A}'$, between two higher-order action calculus models $\mathcal{A}$ and $\mathcal{A}'$ is an action morphism $(\mathcal{F}_{\mathcal{C}}, \mathcal{F}_{\mathcal{S}})$ such that functors per-serve the functor $\Rightarrow$ and the adjunction (which is to say that the following diagrams commute):

$$\begin{tikzcd}
\mathcal{S} \arrow[r, "X \Rightarrow _"] \arrow[d, "\mathcal{F}_{\mathcal{S}}"'] & \mathcal{C} \arrow[d, "\mathcal{F}_{\mathcal{C}}"] \\
\mathcal{S}' \arrow[r, "\mathcal{F}_{\mathcal{C}}(X) \Rightarrow' _"'] & \mathcal{C}'
\end{tikzcd}$$

$$\begin{array}{ccc}
\mathcal{S}(F(X)\otimes Y_1, Y_2) & \xrightarrow{(_)^*} & \mathcal{C}(X, Y_1 \Rightarrow Y_2) \\
\Big\downarrow{\scriptstyle \mathcal{F}_\mathcal{S}} & & \Big\downarrow{\scriptstyle \mathcal{F}_\mathcal{C}} \\
\mathcal{S}'(\mathcal{F}_\mathcal{S}(F(X)\otimes Y_1), \mathcal{F}_\mathcal{S}(Y_2)) & & \mathcal{C}'(\mathcal{F}_\mathcal{C}(X), \mathcal{F}_\mathcal{C}(Y_1 \Rightarrow Y_2)) \\
|| & & || \\
\mathcal{S}'(F'\mathcal{F}_\mathcal{C}(X)\otimes' \mathcal{F}_\mathcal{S}(Y_1), \mathcal{F}_\mathcal{S}(Y_2)) & \xrightarrow[(_)^*]{} & \mathcal{C}'(\mathcal{F}_\mathcal{C}(X), \mathcal{F}_\mathcal{S}(Y_1) \Rightarrow' \mathcal{F}_\mathcal{S}(Y_2))
\end{array}$$

As before, we have:

LEMMA 6.5.9
Any higher-order action morphism gives rise to a $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$-morphism.

Proof The underlying action morphism gives rise to a $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-morphism, and this is seen also to preserve the added structure of $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$ since it preserves the adjunction which underlies that structure. □

Now we can define the category of small higher-order action models and morphisms, written $\mathsf{Cat}_{\mathsf{AC}^{\Rightarrow}}(\mathbb{K})$, and the category of small $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$-models and morphisms, written $\mathsf{Cat}^{\Rightarrow}_{\mathsf{Lin}^{\mathsf{A}}}(\mathbb{K})$. We then have a functor $\mathsf{Cat}_{\mathsf{AC}^{\Rightarrow}}(\mathbb{K}) \to \mathsf{Cat}^{\Rightarrow}_{\mathsf{Lin}^{\mathsf{A}}}(\mathbb{K})$ as before, but we also have the obvious forgetful reducts $\mathsf{Cat}_{\mathsf{AC}^{\Rightarrow}}(\mathbb{K}) \to \mathsf{Cat}_{\mathsf{AC}}(\mathbb{K})$ and $\mathsf{Cat}^{\Rightarrow}_{\mathsf{Lin}^{\mathsf{A}}}(\mathbb{K}) \to \mathsf{Cat}_{\mathsf{Lin}^{\mathsf{A}}}(\mathbb{K})$.

Following the definition of higher-order action models for Milner's higher-order action calculus $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$, we could define models for $\mathsf{AC}_{\boxtimes}(\mathbb{K})$, based on the functor F having an adjoint, and for $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$, based on F having an adjoint and $\mathcal{S}$ being strict symmetric monoidal closed. Then results analogous to those above relating $\mathsf{AC}_{\boxtimes}(\mathbb{K})$-models and $\mathsf{Lin}^{\mathsf{A}^{\boxtimes}}(\mathbb{K})$-models and morphisms, and $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$-models and $\mathsf{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K})$-models and morphisms would hold. This development is pursued in [BGHP97], and the reader is referred there for the details. However, we prefer to develop a more uniform account of higher-order behaviour in the context of our generalised linear type-theory. This we will give in the next chapter.

Chapter 7

Higher-Order Type Theories

In this chapter we give a general higher-order (functional) version of our generalised linear type-theory. We see how any generalised linear type-theory gives rise to a higher order extension, and show that the embedding of the original theory into the higher-order extension is conservative, using a semantic argument. We consider some consequences of this for our case study in action calculi.

Then, we introduce a higher-order type-theory which is isomorphic to our type theory $\mathsf{DILL}(\mathbb{C})$, hence showing that the notion of higher-order extension which we have given is well founded.

7.1 Higher-Order Type Theories

It will be the case that a higher-order generalised linear type-theory (which henceforth we will call just a higher-order type-theory) is an instance of a generalised type-theory with certain higher-order operators and axioms. We will need to give a generalised signature for this instance.

First, given type-sets M_I and M_L, define the *higher-order type sets* $\mathsf{M}_I^{\mathsf{H}}$ and $\mathsf{M}_L^{\mathsf{H}}$ as follows, inductively:

$$Q \in \mathsf{M}_I^{\mathsf{H}} ::= Q \in \mathsf{M}_I \mid !A \text{ (for } A \in \mathsf{M}_L^{\mathsf{H}})$$
$$A \in \mathsf{M}_L^{\mathsf{H}} ::= A \in \mathsf{M}_L \mid I \mid A \otimes A \mid A \multimap A \mid !A$$

We can now define a higher-order signature, which is essentially a signature in which the operators may use higher-order types over the primitive types given in the signature.

Definition 7.1.1 (Higher-Order Signature)
Define a *higher-order signature*, ranged over by $\mathbb{H}$, to be a quadruple $(\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$ such that $(\mathsf{M}_I^{\mathsf{H}}, \mathsf{M}_L^{\mathsf{H}}, \mathcal{O}_I, \mathcal{O}_L)$ is a generalised signature.

Now, given a higher-order signature $(\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$, define the *higher-order operator sets* $\mathcal{O}^{\mathsf{H}}_I$ and $\mathcal{O}^{\mathsf{H}}_L$ as follows, again inductively:

$$
\begin{aligned}
&O \in \mathcal{O}^{\mathsf{H}}_I ::= O \in \mathcal{O}_I \mid\, ! \\
&O \in \mathcal{O}^{\mathsf{H}}_L ::= O \in \mathcal{O}_L \mid I^R \mid I^L_A \mid \otimes^R_{A,B} \mid \otimes^L_{A,B-C} \mid \lambda_{A,B} \mid \mathsf{ap}_{A,B} \mid !_A \mid \mathsf{i}_A
\end{aligned}
$$

for $A, B, C \in \mathsf{M}^{\mathsf{H}}_L$, where:

- the operator I^R is a constant with arity I,
- the output-parameterised set of operators I^L have arity:

$$\frac{;\quad (;)(\_)}{(;I)(\_)}$$

- for each A and B, the operator $\otimes^R_{A,B}$ has arity:

$$\frac{;\quad (;)A \quad (;)B}{(;)A \otimes B}$$

- for each A and B, the output-parameterised set of operators $\otimes^L_{A,B}$ has arity:

$$\frac{;\quad (;A,B)(\_)}{(;A \otimes B)(\_)}$$

- for each A and B, the operator $\lambda_{A,B}$ has arity:

$$\frac{;\quad (;A)B}{(;)A \multimap B}$$

- for each A and B, the operator $\mathsf{ap}_{A,B}$ has arity:

$$\frac{;}{(;A, A \multimap B)B}$$

- for each A, the operator $!_A$ has arity:

$$\frac{(;)A \quad ;}{(;)!A}$$

- for each A, the operator i_A has arity:

$$\frac{;}{(;!A)A}$$

From now on, we will use standard abbreviations for these operators as follows:

$$
\begin{aligned}
* &\text{ for } I^R \\
\mathsf{let}\ * \ \mathsf{be}\ v\ \mathsf{in}\ w &\text{ for } I^L_A(;()w)(;v) \\
w \otimes v &\text{ for } \otimes^R_{A,B}\,(;()w,()v)() \\
\mathsf{let}\ x \otimes y : A \otimes B\ \mathsf{be}\ v\ \mathsf{in}\ w &\text{ for } \otimes^L_{A,B-C}\,(;(;x{:}A,y{:}B)w)(;v) \\
\lambda x{:}A.v &\text{ for } \lambda_{A,B}(;(;x{:}A)v)() \\
vw &\text{ for } \mathsf{ap}_{A,B}(;)(;v,w) \\
!v &\text{ for } !_A(v;)() \\
\mathsf{i}v &\text{ for } \mathsf{i}_A(;)(;v)
\end{aligned}
$$

We will write $\mathbb{O}^{\mathbb{H}}$ for the generalised signature $(\mathsf{M}^{\mathsf{H}}_I, \mathsf{M}^{\mathsf{H}}_L, \mathcal{O}^{\mathsf{H}}_I, \mathcal{O}^{\mathsf{H}}_L)$, where $\mathbb{H}$ is the higher-order signature $(\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$.

Now we can define the higher-order typing-system.

Definition 7.1.2 (Higher-Order Typing System)
The *higher-order typing system* over a higher-order signature $\mathbb{H}$, which we will write $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$, is just the generalised linear typing system $\mathsf{Lin}(\mathbb{O}^{\mathbb{H}})$.

Say that a *higher order axiom set* is just an axiom set $\mathbb{A}$ over the higher-order typing system $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$.

Definition 7.1.3 (Basic Higher-Order Axiom Set)
Define the *basic higher order axiom set* $\mathbb{A}^{\mathbb{H}}$ over a higher-order signature $\mathbb{H}$ to be the axiom set containing the equality judgements in the sets $\mathsf{ONat}(\mathbb{O}^{\mathbb{H}}, I^L)$ and $\mathsf{ONat}(\mathbb{O}^{\mathbb{H}}, \otimes^L_{A,B})$ for all $A, B \in \mathsf{M}^{\mathsf{H}}_L$, and every instance for types of $\mathbb{H}$ of the equality judgements:

β	η
$\Gamma;\Delta \vdash \mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ v = v{:}A$	$\Gamma;\Delta \vdash \mathsf{let}\ *\ \mathsf{be}\ v\ \mathsf{in}\ * = v{:}I$
$\Gamma;\Delta \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ v_1 \otimes v_2\ \mathsf{in}\ w = w\{v_1, v_2/x, y\}{:}A$	$\Gamma;\Delta \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ v\ \mathsf{in}\ x \otimes y{:}A \otimes B$
$\Gamma;\Delta \vdash (\lambda x.v)w = w\{v/x\}{:}A$	$\Gamma;\Delta \vdash \lambda x.(vx){:}A \multimap B$
$\Gamma;\_ \vdash \mathsf{i}(!v) = v{:}A$	$x :!A;\_ \vdash !(\mathsf{i}x) = x :!A$

We are now in a position to define the higher-order type-theory over a given higher-order signature and axiom set.

Definition 7.1.4 (Higher-Order Type-Theory)
Given a higher-order signature $\mathbb{H}$ and a higher-order axiom set $\mathbb{A}$, define the *higher-order type theory over* $\mathbb{H}$ *and* $\mathbb{A}$, written $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A})$, as the generalised linear type-theory $\mathsf{Lin}(\mathbb{O}^{\mathbb{H}}, \mathbb{A} \cup \mathbb{A}^{\mathbb{H}})$.

Derived Typing Rules

Given this definition, $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A})$ has the following typing rules for its higher-order operators:

$$\Gamma; \_ \vdash *{:}I \qquad \frac{\Gamma; \Delta_1 \vdash v{:}I \quad \Gamma; \Delta_2 \vdash w{:}A}{\Gamma; \Delta \vdash \mathsf{let}\ * \ \mathsf{be}\ v\ \mathsf{in}\ w{:}A}$$

$$\frac{\Gamma; \Delta_1 \vdash v{:}A \quad \Gamma; \Delta_2 \vdash w{:}B}{\Gamma; \Delta \vdash v \otimes w{:}A \otimes B} \qquad \frac{\Gamma; \Delta_1 \vdash v{:}A \otimes B \quad \Gamma; \Delta_2, x{:}A, y{:}B \vdash w{:}C}{\Gamma; \Delta \vdash \mathsf{let}\ x \otimes y\ \mathsf{be}\ v\ \mathsf{in}\ w{:}C}$$

$$\frac{\Gamma; \Delta', x{:}A \vdash v{:}B}{\Gamma; \Delta' \vdash \lambda x.v{:}A \multimap B} \qquad \frac{\Gamma; \Delta_1 \vdash v{:}A \multimap B \quad \Gamma; \Delta_2 \vdash w{:}A}{\Gamma; \Delta \vdash vw{:}B}$$

$$\frac{\Gamma; \_ \vdash v{:}A}{\Gamma; \_ \vdash !v :!A} \qquad \frac{\Gamma; \Delta' \vdash v :!A}{\Gamma; \Delta' \vdash \mathsf{¡}(v){:}A}$$

where $\Delta = \Delta_1 \# \Delta_2$.

The Higher-Order Extension of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$

Having defined higher-order type-theories, we can now show that we immediately have a family of such things. Firstly, note that given any generalised signature $\mathbb{O} = (\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$, we have another generalised signature $(\mathsf{M}_I^{\mathsf{H}}, \mathsf{M}_L^{\mathsf{H}}, \mathcal{O}_I, \mathcal{O}_L)$. Therefore, $\mathbb{O}$ is itself a higher-order signature. Further, the embedding on pre-terms in fact maps $\Gamma; \Delta$-terms of $\mathsf{Lin}(\mathbb{O})$ to $\Gamma; \Delta$-terms of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O})$.

Now, note that if $\mathbb{A}$ is an axiom set over the generalised linear typing system $\mathsf{Lin}(\mathbb{O})$, then it is also one over the generalised typing system $(\mathsf{M}_I^{\mathsf{H}}, \mathsf{M}_L^{\mathsf{H}}, \mathcal{O}_I, \mathcal{O}_L)$. Hence, it is itself a higher-order axiom set. We have, therefore, that $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O}, \mathbb{A})$ is a higher-order type-theory, and we call this higher-order type-theory the *higher-order extension* of the original type-theory. It follows easily that the embedding on terms $\mathsf{Lin}(\mathbb{O}, \mathbb{A}) \to \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}, \mathbb{A})$ is sound; later in this chapter, we will use semantic methods to demonstrate that it is also conservative.

Operators as Higher-Order Constants

One advantage of higher-order structure is that it allows us to code general operators as constants of higher types. Unfortunately, there is a slight problem with encoding operators in $\mathcal{O}_I$, which arises since we need to make the image of any instance of such an operator copyable in the higher-order type-theory. In general, there is no way to give a constant encoding of the operator which will achieve this without adding equalities to the system. However, we can prove a result:

Lemma 7.1.5
The arbitrary linear type-theory $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ is equivalent to the linear type theory $\mathsf{Lin}((\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L), \mathbb{A} \cup \bigcup_{O \in \mathcal{O}_I} \mathsf{Int}(\mathbb{O}, O))$, where $\mathsf{Int}(\mathbb{O}, O)$ is defined as the set of all equality judgements of the form

$$\begin{aligned} \Gamma; \Delta \vdash & \mathsf{let}\ x\ \mathsf{be}\ O((\vec{x}_1; \vec{y}_1)v_1, \ldots, (\vec{x}_r; \vec{y}_r)v_r;)()\ \mathsf{in}\ w \\ & = w\{O((\vec{x}_1; \vec{y}_1)v_1, \ldots, (\vec{x}_r; \vec{y}_r)v_r);)()/x\} \colon A \end{aligned}$$

in the typing system $\mathsf{Lin}(\mathbb{O})$.

Proof This can be seen using the translations which are the identity on types and terms in both directions. The only issue is whether these translations preserve provable equality, but this is easily seen since in one direction we map an instance of the F-rule for the operator O to an equality judgement in $\mathsf{Int}(\mathbb{O}, O)$, and in the other direction we map an equality judgement in a set $\mathsf{Int}(\mathbb{O}, O)$ back to an instance of the $F - \beta$-axiom for the intuitionistic operator O. $\square$

This result assures us that any linear type-theory is equivalent to one with no intuitionistic operators at the level of terms and provable equality.

Given a linear operator set $\mathcal{O}_L$, define the *operator constant set* corresponding to it, written $\mathcal{O}_L^{\mathsf{C}}$, to be the set which contains for each operator $O \in \mathcal{O}_L$ having arity:

$$\frac{(\vec{Q}_1; \vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r; \vec{A}_r)B_r \quad ; \quad (\vec{Q}'_1; \vec{A}'_1)B'_1, \quad \ldots, \quad (\vec{Q}'_s; \vec{A}'_s)B'_s}{(\vec{Q}''; \vec{A}'')B''}$$

a constant c_O with arity:

$$(\otimes_{i=1\ldots r} !(\otimes\vec{Q}_i\vec{A}_i \multimap B_i)) \otimes (\otimes_{j=1\ldots s}(\otimes\vec{Q}'_j\vec{A}'_j \multimap B'_j)) \multimap (\otimes\vec{Q}''\vec{A}'' \multimap B'')$$

where $\otimes\vec{A}$ is the standard left-bracketed tensor.

Now, given a higher-order signature $\mathbb{H} = (\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L)$, we can easily see that $(\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L^{\mathsf{C}})$ is another higher-order signature. Write $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{H})$ for the higher-order typing system $\mathsf{Lin}^{\mathsf{H}}(\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L^{\mathsf{C}})$.

Having made this definition, we can give a translation from $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$ to $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{H})$.

Definition 7.1.6 (The Translation $(\_)^\square$)
Define the translation $(\_)^\square$ from $\mathsf{Lin^H}(\mathbb{H})$ to $\mathsf{Lin^{HC}}(\mathbb{H})$ as the identity on types, and on pre-terms as follows:

$$(x)^\square = x$$
$$(\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w)^\square = \mathsf{let}\ x\ \mathsf{be}\ (v)^\square\ \mathsf{in}\ (w)^\square$$

$$\begin{aligned}(O((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(\vec{v}';\vec{w}'))^\square = \\ c_O(\otimes_{i=1\ldots r}!(\lambda z.\mathsf{let}\ \otimes\vec{x}''_i\vec{y}_i\ \mathsf{be}\ z\ \mathsf{in}\ \mathsf{let}\ \vec{x}_i\ \mathsf{be}\ \vec{x}''_i\ \mathsf{in}\ (v_i)^\square)) \\ \otimes(\otimes_{j=1\ldots s}(\lambda z.\mathsf{let}\ \otimes\vec{x}'''_j\vec{y}'_j\ \mathsf{be}\ z\ \mathsf{in}\ \mathsf{let}\ \vec{x}'_j\ \mathsf{be}\ \vec{x}'''_j\ \mathsf{in}\ (w_j)^\square)) \quad (\otimes(\vec{v}'\vec{w}')^\square)\end{aligned}$$

This translation can be shown to take $\Gamma;\Delta$-terms of $\mathsf{Lin^H}(\mathbb{H})$ to $\Gamma;\Delta$-terms of $\mathsf{Lin^{HC}}(\mathbb{H})$.

Now, given a higher-order signature $\mathbb{H} = (\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L)$, we can define the generalised linear type theory $\mathsf{Lin^{HC}}(\mathbb{H},\mathbb{A})$ to be the higher-order type-theory $\mathsf{Lin^H}((\mathsf{M}_I,\mathsf{M}_L,\emptyset,\mathcal{O}^{\mathsf{C}}_L),\mathbb{A}^\square)$. Given this definition, the translation $(\_)^\square$ is sound.

7.2 Higher-Order Semantics

Having given our definition of higher-order type theories, we now need to consider the semantics of such type-theories. Since every higher-order type theory is in fact an instance of a generalised linear type theory, we already have a definition of model for any higher-order signature. However, this inherited definition of model is not ideal, for example because it requires us to give a primitive interpretation of all the new higher-order types which exist in a general higher-order type-theory. We would much prefer that these, and indeed the operators which exists in every higher-order type theory, were interpreted into primitive structure already present in the models in a standard way.

We can make such a definition by considering the semantics of $\mathsf{DILL}(\mathbb{C})$. It is no coincidence that in fact many of the operators of the higher-order type theory have typing rules very similar to those of $\mathsf{DILL}(\mathbb{C})$, or that the models of $\mathsf{DILL}(\mathbb{C})$ which we defined were built on a carrier having a cartesian category with a strong monoidal functor to a symmetric monoidal category, augmented with extra structure. In fact, we will define models of higher-order type theories using most of the structure of $\mathsf{DILL}(\mathbb{C})$-models.

The *carrier* of a $\mathsf{Lin^H}(\mathbb{H},\mathbb{A})$-model is a quadruple $(\mathcal{C},\mathcal{S},F,G)$ such that $\mathcal{C}$ is a cartesian category, $\mathcal{S}$ is a symmetric monoidal closed category, $F:\mathcal{C}\to\mathcal{S}$ is strict monoidal and $G\vdash F$. Note that $\mathcal{C}$ need not necessarily be closed, and that neither $\mathcal{C}$ nor $\mathcal{S}$ need be strict.

Now, the carrier of a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A})$-model easily yields a carrier for a $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model, which we call the *underlying carrier*; simply take the triple $(\mathcal{C}, \mathcal{S}, F)$ with the *strict* cartesian structure on $\mathcal{C}$ given by the left-bracketed product, and the *strict* symmetric monoidal structure on $\mathcal{S}$ given by the left-bracketed tensor, and F as before.

Definition 7.2.1 (Higher-Order $\mathrm{Lin}^{\mathrm{H}}(\mathbb{H})$-interpretation)
A *higher-order interpretation* of the higher-order typing system $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$, which we write $\mathcal{H}$, is a carrier $(\mathcal{C}, \mathcal{S}, F, G)$ together with:

- primitive interpretation functions $[\![\_]\!]^{\mathcal{H}}_{\mathsf{M}_I} : \mathsf{M}_I \to \mathsf{obj}(\mathcal{C})$ and $[\![\_]\!]^{\mathcal{H}}_{\mathsf{M}_L} : \mathsf{M}_L \to \mathsf{obj}(\mathcal{S})$ such that for all $Q \in \mathsf{M}_I$, we have $[\![Q]\!]^{\mathcal{H}}_{\mathsf{M}_L} = F([\![Q]\!]^{\mathcal{H}}_{\mathsf{M}_I})$

- for each operator $O \in \mathcal{O}_L$ having arity

$$\frac{(\vec{Q}_1; \vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r; \vec{A}_r)B_r \quad ; \quad (\vec{Q}'_1; \vec{A}'_1)B'_1, \quad \ldots, \quad (\vec{Q}'_s; \vec{A}'_s)B'_s}{(\vec{Q}''; \vec{A}'')B''}$$

  a natural transformation

$$\begin{aligned}[\![O]\!]^{\mathcal{H}}_{\mathcal{O}_L} :& (\times_{i=1\ldots r}\mathcal{S}(F(_=) \otimes [\![\vec{Q}_i; \vec{A}_i]\!]^{\mathcal{G}}, [\![B_i]\!]^{\mathcal{G}}_{\mathsf{M}_L})) \times \\ & \quad (\times_{j=1\ldots s}\mathcal{S}(F(_=) \otimes [\![\vec{Q}'_j]\!] \otimes (\_{j}) \otimes [\![\vec{A}'_j]\!]^{\mathcal{G}}, [\![B'_j]\!]^{\mathcal{G}}_{\mathsf{M}_L})) \\ & \qquad \to \mathcal{S}(F(_=) \otimes [\![\vec{Q}'']\!] \otimes (\otimes_{j=1\ldots s}(\_{s})) \otimes [\![\vec{A}'']\!]^{\mathcal{G}}, [\![B'']\!]^{\mathcal{G}}_{\mathsf{M}_L})\end{aligned}$$

  which is natural independently in each of the $s+1$ arguments $(_=)$ and $(\_{1}), \ldots, (\_{s})$, and where the interpretation $[\![\_]\!]^{\mathcal{G}}$ is extended to arbitrary contexts in the obvious way, as given shortly,

- for each operator $O \in \mathcal{O}_I$ having arity

$$\frac{(\vec{Q}_1; \vec{A}_1)B_1, \quad \ldots, \quad (\vec{Q}_r; \vec{A}_r)B_r;}{(;)R}$$

  a natural transformation

$$\begin{aligned}[\![O]\!]^{\mathcal{H}}_{\mathcal{O}_I} : \times_{i=1\ldots r}\, \mathcal{S}(F(_=) \otimes [\![\vec{Q}_i; \vec{A}_i]\!]^{\mathcal{G}}, [\![B_i]\!]^{\mathcal{G}}_{\mathsf{M}_L}) \\ \to \mathcal{C}(_=, [\![Q'']\!]^{\mathcal{G}}_{\mathcal{C}})\end{aligned}$$

  where the interpretation is again given shortly, such that for all objects X of $\mathcal{C}$ and all arrows $f_i : F(X) \otimes [\![\vec{Q}_i; \vec{A}_i]\!] \to [\![B_i]\!]$ where $i = 1 \ldots r$ in $\mathcal{S}$,

$$[\![O]\!]^{\mathcal{G}}_{\mathcal{O}_L}(X)(f_1, \ldots, f_r) = F([\![O]\!]^{\mathcal{G}}_{\mathcal{O}_I}(X)(f_1, \ldots, f_r))$$

We can now easily extend the interpretation functions $[\![\_]\!]$ to higher order types:

$$
\begin{aligned}
[\![I]\!] &= I \\
[\![A \otimes B]\!] &= [\![A]\!] \otimes [\![B]\!] \\
[\![A \multimap B]\!] &= [\![A]\!] \multimap [\![B]\!] \\
[\![!A]\!] &= FG([\![A]\!])
\end{aligned}
$$

$$
[\![!A]\!]_{\mathcal{C}} = G([\![A]\!])
$$

and to sequences of such types using the left-bracketed product and tensor in the familiar way. Further, we can extend the interpretation of operators as follows:

$$
\begin{aligned}
[\![*]\!](X) &= F(\mathsf{d}_X) \\
[\![I^L_A]\!](X;Y)(f) &= \mathsf{li}_{F(=)\otimes\_}; f \\
[\![\otimes^R_{A,B}]\!](X;Y_1Y_2)(f_1f_2) &= \mathsf{admin}_1; (f_1 \otimes f_2) \\
[\![\otimes^L_{A,B-C}]\!](X;Y)(f) &= \mathsf{admin}_2; f \\
[\![\lambda_{A,B}]\!](X;Y)(f) &= \lambda f \\
[\![\mathsf{ap}_{A,B}]\!](X;) &= (F(\mathsf{d}_X) \otimes \mathsf{ap}); \mathsf{ri}_{[\![B]\!]} \\
[\![!_A]\!](X)(f) &= F(\mathsf{un}_X; G(f)) \\
[\![\mathsf{i}_A]\!](X) &= (F(\mathsf{d}_X) \otimes \mathsf{nu}_{[\![A]\!]}); \mathsf{ri}_{[\![A]\!]}
\end{aligned}
$$

$$
[\![!_A]\!]_{\mathcal{C}}(X)(f) = \mathsf{un}_X; G(f)
$$

where $\mathsf{admin}_!$ and admin_2 are the obvious morphisms in the s.m.c.:

$$
\begin{aligned}
\mathsf{admin}_1 :& (FX \otimes Y_1) \otimes Y_2 \to (FX \otimes Y_1) \otimes (FX \otimes Y_2) \\
\mathsf{admin}_2 :& ((FX \otimes [\![A]\!]) \otimes [\![B]\!]) \otimes Y \to (FX \otimes ([\![A]\!] \otimes [\![B]\!])) \otimes Y
\end{aligned}
$$

Now, we have a lemma:

Lemma 7.2.2
Given a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$-interpretation, its underlying carrier (with the strict left-bracketed product and tensor) and the extended interpretation functions defined above make up all the information required for a $\mathsf{Lin}(\mathbb{O}^{\mathbb{H}})$-interpretation.

We can therefore interpret terms of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$ into our model by the derived interpretation, and henceforth we will use this derived interpretation of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$ into $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$-interpretations. Given this derived interpretation, we have a lemma on the axiom set $\mathbb{A}^{\mathbb{H}}$.

Lemma 7.2.3
Any $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$-interpretation $\mathcal{H}$ interprets the equality judgements in the axiom set $\mathbb{A}^{\mathbb{H}}$ as categorical equalities in the s.m.c. part of $\mathcal{H}$.

This is easily shown by analogy with soundness for $\mathsf{DILL}(\mathbb{C})$ in $\mathsf{DILL}(\mathbb{C})$-models, since we have the same underlying categorical structure. With this lemma, we can now define $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-models.

Definition 7.2.4 ($\mathrm{Lin}^{\mathrm{H}}(\mathbb{H},\mathbb{A})$-model)
A $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-model is a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$-interpretation $\mathcal{H}$ such that the derived interpretation given by lemma 7.2.2 maps equality judgements of the higher-order axiom set $\mathbb{A}$ to equalities in the category $\mathcal{S}$ which is the s.m.c. part of the carrier of $\mathcal{H}$.

Soundness for these models is easy to prove:

Lemma 7.2.5 (Soundness)
Given a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-model $\mathcal{H}$, and equality judgement $\Gamma;\Delta \vdash v = w : A$ of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$, we have that:

$$[\![\Gamma;\Delta \vdash v{:}A]\!]^{\mathcal{H}} = [\![\Gamma;\Delta \vdash w{:}A]\!]^{\mathcal{H}}$$

Proof The equality rules are soundly interpreted since categorical equality is a congruence, the basic axioms of the generalised linear type-theory are soundly interpreted by virtue of the fact that the interpretation is derived from a generalised linear type-theory interpretation for which we already have soundness, the axioms in the set $\mathbb{A}^{\mathbb{H}}$ are soundly interpreted by lemma 7.2.3 and finally the axioms in the set $\mathbb{A}$ of the higher-order signature are soundly interpreted by definition. $\square$

We now consider completeness. As usual, we will proceed by defining a term model.

Definition 7.2.6 (The Linear Term Category)
We define the linear term category, $\mathcal{HS}_T$, as follows:

- The objects of $\mathcal{HS}_T$ are linear types of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$.
- $\mathcal{HS}_T(A,B) = \{[(x,v)^{A,B}] | \_; x:A \vdash v:B\}$, where we write $[(x,v)^{A,B}]$ as normal to denote the equivalence class of $(x,v)^{A,B}$ under the equivalence $\equiv''$ defined by

$$(x,v)^{A,B} \equiv'' (y,w)^{A,B} \text{ if } \_; x{:}A \vdash v = w\{x/y\}{:}A$$

As before, we omit type annotations where convenient and write $[(x, v)]$ as $[x, v]$.

Identities and composition are defined as for the term model of $\mathsf{DILL}(\mathbb{C})$, and further, the closed monoidal structure is also defined as for the term model of $\mathsf{DILL}(\mathbb{C})$.

Definition 7.2.7 (The Intuitionistic Term Category)
We define the intuitionistic term category, $\mathcal{HC}_T$, as follows:

- The objects of $\mathcal{HC}_T$ are sequences of intuitionistic types of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A})$.
- $\mathcal{HC}_T(\vec{A}, \vec{B}) = \{[(\vec{x}, \vec{v})^{\vec{A},\vec{B}}] | \vec{x} : \vec{A}; \_ \vdash v_i : \otimes B_i$ and each v_i is intuitionistic$\}$, where we write $[(\vec{x}, \vec{v})^{\vec{A},\vec{B}}]$ to denote the equivalence class of $(\vec{X}, \vec{V})^{\vec{A},\vec{B}}$ under the equivalence $\equiv''_C$ defined by:

  $$(\vec{x}, \vec{v})^{\vec{A},\vec{B}} \equiv''_C (\vec{y}, \vec{w})^{\vec{A},\vec{B}}$$

  if the sequences $\vec{v}$ and $\vec{w}$ are the same length, the sequence $\vec{x}$ and $\vec{y}$ are the same length, and for each i, we have $\vec{x}{:}\vec{A}; \_ \vdash v_i =_I w_i\{\vec{x}/\vec{y}\}{:}B_i$.

  As before, we omit type information where convenient, abbreviate $[(\vec{x}, \vec{v})]$ to $[\vec{x}, \vec{v}]$ and assume that in such an arrow, all the variables in $\vec{x}$ are distinct.

We define identities, composition and a strict cartesian structure over this just as for the term model of $\mathsf{DILL}(\mathbb{C})$.

Now we define F_T on objects of $\mathcal{HC}_T$ as the left-bracketed tensor, and on arrows as

$$F_T([\vec{x}, \vec{v}]) = [y, \mathsf{let}\ \otimes\vec{x}'\ \mathsf{be}\ y\ \mathsf{in}\ \mathsf{let}\ \vec{x}\ \mathsf{be}\ \vec{x}'\ \mathsf{in}\ \otimes\vec{v}]$$

This can be seen to be functorial. Define G_T on objects of $\mathcal{HS}_T$ to take A to the intuitionistic type $!A$. On objects, define

$$G_T([x, v]) = [y, !(v\{\mathrm{i}y/x\})]$$

Now we can define the term model:

Definition 7.2.8 (Higher-Order Term Model)
Define the *higher-order term model*, written $\mathcal{H}_T$, to have carrier $(\mathcal{HC}_T, \mathcal{HS}_T, F_T, G_T)$, with interpretation functions given in the obvious way.

It is a straightforward exercise to give these interpretation functions and show that the definitions given make $\mathcal{H}_T$ into a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A})$-model.

We can also easily show that

Lemma 7.2.9

Given a $\Gamma;\Delta$-term v of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$ having type A, we have:

$$[\![\Gamma;\Delta \vdash v{:}A]\!]^{\mathcal{H}_T} = [z, \mathsf{let}\ \otimes\vec{x}'\vec{y}\ \mathsf{be}\ z\ \mathsf{in}\ \mathsf{let}\ \vec{x}\ \mathsf{be}\ \vec{x}'\ \mathsf{in}\ v]$$

where z and $\vec{x}'$ are fresh, $\Gamma = \vec{x}{:}\vec{Q}$ and $\Delta = \vec{y}{:}\vec{A}$.

Now we can prove completeness by the standard method.

Lemma 7.2.10 (Completeness)

Given two $\Gamma;\Delta$-terms v and w of type A in $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$, we have a provable equality judgement $\Gamma;\Delta \vdash v = w{:}A$ iff in all $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-models, we have:

$$[\![\Gamma;\Delta \vdash v{:}A]\!] = [\![\Gamma;\Delta \vdash w{:}A]\!]$$

This is proved as normal.

We can define $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$-morphisms as follows:

Definition 7.2.11 ($\mathrm{Lin}^{\mathrm{H}}(\mathbb{H})$-morphisms)

A $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$-morphism is a pair of functors $(\mathcal{F}_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}', \mathcal{F}_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}')$ such that:

- $\mathcal{F}_{\mathcal{C}}$ is strict cartesian and $\mathcal{F}_{\mathcal{S}}$ is strict monoidal closed,
- the following diagrams commute:

$$\begin{tikzcd} \mathcal{C} \arrow[r, "F"] \arrow[d, "\mathcal{F}_{\mathcal{C}}"'] & \mathcal{S} \arrow[d, "\mathcal{F}_{\mathcal{S}}"] \\ \mathcal{C}' \arrow[r, "F'"] & \mathcal{S}' \end{tikzcd} \qquad \begin{tikzcd} \mathcal{C} \arrow[d, "\mathcal{F}_{\mathcal{C}}"'] & \mathcal{S} \arrow[l, "G"'] \arrow[d, "\mathcal{F}_{\mathcal{S}}"] \\ \mathcal{C}' & \mathcal{S}' \arrow[l, "G'"'] \end{tikzcd}$$

- $(\mathcal{F}_{\mathcal{C}}, \mathcal{F}_{\mathcal{S}})$ is a $\mathsf{Lin}(\mathbb{O}^{\mathbb{H}})$-morphism.

It is easy to see that the third of these conditions can be replaced by the weaker condition that the pair of functors preserve the primitive interpretations on M_I,M_L,$\mathcal{O}_I$ and $\mathcal{O}_L$, as the condition as stated can be deduced from the fact that they preserve the structure of the carrier of the $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$-interpretation.

Define $\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{H}}}(\mathbb{H},\mathbb{A})$ to be the category of small $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-models and morphisms. Further, define $\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{HC}}}(\mathbb{O},\mathbb{A})$ to be the category of small $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-models and morphisms

It is now possible to define a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-morphism from $\mathcal{H}_T$ to any arbitrary $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-model $\mathcal{H}$ using the interpretation.

Definition 7.2.12 (Initiality Morphism)
Given a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-model $\mathcal{H}$, we have a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-morphism $\mathcal{FT} = (\mathcal{FT}_{\mathcal{C}}, \mathcal{FT}_{\mathcal{S}}) : \mathcal{H}_T \to \mathcal{H}$. Define $\mathcal{FT}_{\mathcal{C}}$ on objects of $\mathcal{HC}_T$ as the left-bracketed product of the interpretations of the sequence of types, and on arrows $[\vec{x}, \vec{v}]$ as the pairing of the $[\![\vec{x}:\vec{A}; \_ \vdash v_i]\!]^{\mathcal{H}}_{\mathcal{C}}$.

Further, define $\mathcal{FT}_{\mathcal{C}}$ on objects just as their interpretation, and on arrows $[x, v]$ as the interpretation $[\![\_; x:A \vdash v:B]\!]^{\mathcal{H}}$.

Now we can show that this initiality morphism is a $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-morphism and is unique up to isomorphism.

It is now worth noting that since we know that every $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\mathbb{A})$-model gives rise to a $\mathsf{Lin}(\mathbb{O}^{\mathbb{H}}, \mathbb{A}\cup\mathbb{A}^{\mathbb{H}})$ model and similarly for morphisms, we have a functor from $\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{H}}}(\mathbb{H},\mathbb{A})$ to the category of $\mathsf{Lin}(\mathbb{O}^{\mathbb{H}}, \mathbb{A}\cup\mathbb{A}^{\mathbb{H}})$-models and morphisms.

7.3 Conservativity

In this section we will use the Yoneda Lemma to prove that the translation of the $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ into its higher-order extension $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$ is sound and conservative. Since we know by lemma 7.1.5 that every generalised linear type-theory is isomorphic to one without any intuitionistic operators, for the remainder of this section we only consider generalised signatures of this form.

Summary of the Proof To show conservativity of the translation, the basic procedure is as follows. Given the term model of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, we can construct via the Yoneda lemma a new $\hat{\mathcal{C}}, \hat{\mathcal{S}}$ and $\hat{F}$ derived from the carrier of the term model such that $\hat{\mathcal{C}}$ is a cartesian closed, $\hat{\mathcal{S}}$ is s.m. closed, and $\hat{F}$ has a right adjoint $G : \hat{\mathcal{S}} \to \hat{\mathcal{C}}$ which is symmetric monoidal. These elements can be used to construct a $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-model $\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A}))$, and further this $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-model induces a $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model $\kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))$. But the Yoneda lemma also tells us that there exist a fully faithful cartesian functor $Y_C : \mathcal{C} \to \hat{\mathcal{C}}$ and a fully faithful s.m. functor $Y_S : \mathcal{S} \to \hat{\mathcal{S}}$ such that $Y_C \circ \hat{F} \simeq F \circ Y_S$ up to monoidal natural isomorphism. These can be used to construct a $\mathsf{Lin}(\mathbb{O})$-morphism $\mathcal{G}_T(\mathbb{O},\mathbb{A}) \to \kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))$ which is fully faithful in its s.m. component.

Now, assume that the translations from $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ to $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$ of two $\Gamma;\Delta$-terms of type A are provably equal in $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$. Then by soundness their interpretations must be equal in $\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A}))$. Now we use the fact that their interpretations in the s.m.c. part of this model must be the same as the interpretation of the original $\Gamma;\Delta$ terms of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ in the s.m.c. part of the induced model $\kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))$. But by initiality we know that the fully faithful $\mathsf{Lin}(\mathbb{O})$-morphism we constructed must be the canonical $\mathsf{Lin}(\mathbb{O})$-morphism from the term category, which takes each term arrow to its interpretation. Hence the interpretations of our two terms in the induced $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model are the images under the morphism of their interpretations in the term model $\mathcal{G}_T(\mathbb{O},\mathbb{A})$, but since their interpretations are equal in the induced category by faithfulness their interpretations in the term model will be equal, and hence by completeness the terms themselves are equal.

The outline of the proof is simple, but there are a number of problems to be overcome which we have glossed over in this account under the phrase "can be used to construct". These include size problems and strictness issues.

Preliminaries

We now give some type-theoretic results which will be used in the proof of conservativity.

DEFINITION 7.3.1

We define an embedding of categories $\kappa : \mathsf{Cat}_{\mathsf{Lin}^{\mathsf{HC}}}(\mathbb{O},\mathbb{A}) \to \mathsf{Cat}_{\mathsf{Lin}}(\mathbb{O},\mathbb{A})$ as follows: Given a $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-model $\mathcal{G}$ with carrier $(\mathcal{C},\mathcal{S},F,G)$ and interpretation functions $[\![\_]\!]^{\mathcal{G}}$, let $\kappa(\mathcal{G})$ have carrier $(\mathcal{C},\mathcal{S},F)$ with the strict left-bracketed product and tensor, interpretation functions on types those of $\mathcal{G}$, and finally the interpretation on operators in $\mathcal{O}_L$ given via the interpretation of constants in $\mathcal{G}$ and the following isomorphism:

$$\begin{aligned}
&\mathcal{S}(I, [\![\bigotimes_{i=1\ldots r} !(\otimes\vec{Q}_i\vec{A}_i \multimap B_i) \otimes (\bigotimes_{j=1\ldots s} (\otimes\vec{Q}'_j\vec{A}'_j \multimap B'_j)) \multimap (\otimes\vec{Q}''\vec{A}'' \multimap B'')]\!])\\
\simeq &\mathsf{Nat}_{\mathcal{S},\ldots,\mathcal{S},\mathcal{C}}(\prod_{i=1\ldots r} \mathcal{S}(F(_)\otimes[\![\vec{Q}_i;\vec{A}_i]\!],[\![B_i]\!]) \times \prod_{j=1\ldots s} \mathcal{S}(F(_)\otimes _{j} \otimes [\![\vec{Q}'_j;\vec{A}'_j]\!],[\![B'_j]\!]),\\
&\qquad\qquad\qquad\qquad \mathcal{S}(F(_)\otimes _{1}\otimes\ldots\otimes _{s}\otimes[\![\vec{Q}'';\vec{A}'']\!],[\![B'']\!]))
\end{aligned}$$

Having defined this embedding, we give a lemma on interpretations in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-models.

LEMMA 7.3.2

For any $\Gamma;\Delta$-term v of type A in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, and any $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model $\mathcal{G}$, $[\![\Gamma;\Delta \vdash v{:}A]\!]^{\mathcal{G}} = \mathcal{FT}^{\mathcal{G}}([\![\Gamma;\Delta\vdash v{:}A]\!])$, where $\mathcal{FT}^{\mathcal{G}} : \mathcal{G}_T(\mathbb{O},\mathbb{A}) \to \mathcal{G}$ is the unique initiality map.

This is easily seen from the definition of the canonical morphism (definition 5.5.3, on page 111). We also need a lemma relating the interpretations of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ and $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$.

Lemma 7.3.3
Given a $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-model $\mathcal{G}$ and a $\Gamma;\Delta$-term v of type A in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, then $[\![\Gamma;\Delta \vdash v^{\Box}{:}A]\!]^{\mathcal{G}} = [\![\Gamma;\Delta \vdash v{:}A]\!]^{\kappa(\mathcal{G})}$ in the s.m. category which is common to $\mathcal{G}$ and $\kappa(\mathcal{G})$.

In order to see this, we need to note firstly that the structure used by both interpretations is all on the carrier and hence shared apart from the primitive interpretation of the operators, and it is straightforward isomorphism-chasing to show that the interpretation of any operator instance is the same when it is first mapped to $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$ as it is when interpreted directly from $\mathsf{Lin}(\mathbb{O},\mathbb{A})$ into $\kappa(\mathcal{G})$.

Given this last result, we can now show soundness of the translation $(\_)^{\Box}$ using our semantic results.

Lemma 7.3.4 (Soundness)
If $\Gamma;\Delta \vdash v = w{:}A$ in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, then $\Gamma;\Delta \vdash (v)^{\Box} = (w)^{\Box}{:}A$ in $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$.

Proof Assume that we have an arbitrary $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-model $\mathcal{G}$, and two terms $\Gamma;\Delta \vdash v = w{:}A$ in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$. Now by soundness of the $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-interpretation we must have that $[\![\Gamma;\Delta \vdash v : A]\!]^{\kappa(\mathcal{G})} = [\![\Gamma;\Delta \vdash w : A]\!]^{\kappa(\mathcal{G})}$ in the induced $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model, and also by our lemma $[\![\Gamma;\Delta \vdash v^{\Box}: A]\!]^{\mathcal{G}} = [\![\Gamma;\Delta \vdash w^{\Box}: A]\!]^{\mathcal{G}}$ under the interpretation of $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$ in the arbitrary $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-model. But then by completeness of $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$, we must have that $\Gamma;\Delta \vdash v^{\Box} = w^{\Box}{:}A$ as required. $\Box$

The Yoneda Construction

We now consider the details of the Yoneda construction which we will use to obtain a $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-model $\varphi(G)$ from any given $\mathsf{Lin}(\mathbb{O},\mathbb{A})$-model $\mathcal{G}$, and also to show that the unique $\mathsf{Lin}(\mathbb{O})$-morphism $\mathcal{FT}^{\kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))} : \mathcal{G}_T(\mathbb{O},\mathbb{A}) \to \kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))$ has a fully faithful s.m. part. As we noted at the beginning of this section, there are some technical problems which we must consider carefully. First we state the Yoneda lemma in the form that we will use it.

Lemma 7.3.5 (Yoneda)
Given a locally small category $\mathcal{C}$, define $\hat{\mathcal{C}}$ to be the functor category $\mathsf{Set}^{\mathcal{C}^{op}}$, and define $Y_{\mathcal{C}} : \mathcal{C} \to \hat{\mathcal{C}}$ to be the functor which takes any object of $\mathcal{C}$, X, to the hom-functor $\mathcal{C}(\_,X)$ and which takes any arrow of $\mathcal{C}$, $f : X \to Y$, to the natural transformation $\mathcal{C}(\_,f) : \mathcal{C}(\_,X) \to \mathcal{C}(\_,Y)$ with the obvious compositional action. This construction has the following properties [Day70a, Day70b, Day73]:

- If $\mathcal{C}$ is symmetric monoidal, then $\hat{\mathcal{C}}$ is symmetric monoidal closed, and $Y_{\mathcal{C}}$ is strong symmetric monoidal.

- If $\mathcal{C}$ is cartesian, then $\hat{\mathcal{C}}$ is cartesian closed, and $Y_{\mathcal{C}}$ is strong cartesian.
- Given a strong symmetric monoidal functor $F : \mathcal{C} \to \mathcal{S}$, we have a strong symmetric monoidal functor $\hat{F} : \hat{C} \to \hat{S}$ which is unique up to isomorphism, has a right adjoint and such that we have a symmetric monoidal natural isomorphism $\hat{F} \circ Y_{\mathcal{C}} \simeq Y_{\mathcal{S}} \circ F$.

As we have previously said, given this lemma there are some technical difficulties to overcome before we can achieve our aim, which is to find a small $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O}, \mathbb{A})$-model, based on the Yoneda construction over the term model, such that the initiality morphism to κ of this model in the category $\mathsf{Cat}_{\mathsf{Lin}}(\mathbb{O}, \mathbb{A})$ is faithful in its s.m. part.

The first problem is a size problem. The categories yielded by the Yoneda construction are not small in general, and hence no model we construct directly using them will be in any of our categories of models. However, we can avoid this by taking full subcategories of the categories $\hat{\mathcal{C}}$ containing (the image of) $\mathcal{C}$ and closed under the relevant operations.

The second difficulty is that although models of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ have strict tensor structure and strict cartesian structure, the Yoneda construction yields categories which have corresponding structure which may not be strict. We therefore present our first proposition:

Proposition 7.3.6 (Strictness)
Given a symmetric monoidal category $\mathcal{C}$, there exists a strict symmetric monoidal category $\mathcal{C}_s$ which is equivalent to $\mathcal{C}$. Similarly, given a cartesian category $\mathcal{C}$, there exists a strict cartesian category $\mathcal{C}_s$ which is equivalent to $\mathcal{C}$. Also, given a symmetric monoidal functor $F : \mathcal{C} \to \mathcal{C}'$, there exists a strict symmetric monoidal functor $F_s : \mathcal{C}_s \to \mathcal{C}'_s$ such that the following diagram commutes, where U_C and S_C are the fully faithful functors witnessing the equivalence:

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "F"] \arrow[d, shift right, "S_{\mathcal{C}}"'] & \mathcal{C}' \arrow[d, shift right, "S_{\mathcal{C}'}"'] \\
\mathcal{C}_s \arrow[r, "F_s"] \arrow[u, shift right, "U_{\mathcal{C}}"'] & \mathcal{C}'_s \arrow[u, shift right, "U_{\mathcal{C}'}"']
\end{tikzcd}$$

The final difficulty is that given a $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model $\mathcal{G}$ with carrier $(\mathcal{C}, \mathcal{S}, F)$, the candidate carrier for a $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O}, \mathbb{A})$-model, $(\hat{\mathcal{C}}_s, \hat{\mathcal{S}}_s, \hat{F}, G)$ has the property that $Y_{\mathcal{C}}; \hat{F} \simeq F; Y_{\mathcal{S}}$, where we will want to use the Yoneda functors $Y_{\mathcal{C}}$ and $Y_{\mathcal{S}}$ to construct a $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-map from $\mathcal{G}$ to the $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O}, \mathbb{A})$-model based on the Yoneda construction. Hence we recall the following proposition:

Proposition 7.3.7 (Skeleton)
Given a category $\mathcal{C}$, there exists a category $sk(\mathcal{C})$ which is equivalent to $\mathcal{C}$ such that any two objects of $sk(\mathcal{C})$ which are isomorphic are equal and any two arrows of $sk(\mathcal{C})$ which are isomorphic in the arrow category of $sk(\mathcal{C})$ are equal, where the arrow category of a category $\mathcal{S}$ has objects the arrows of $\mathcal{S}$ and as arrows $(f : X \to Y) \to (g : X' \to Y')$ pairs of arrows of $\mathcal{S}$, (h_1, h_2) where $h_1 : X \to X'$ and $h_2 : Y \to Y'$ and the obvious diagram commutes.

Note that the fully faithful functors $K : \mathcal{S} \to sk(\mathcal{S})$ and $I : sk(\mathcal{S}) \to \mathcal{S}$ given by this proposition are strict symmetric monoidal under the obvious (strict) s.m. structure over the skeleton. Given this machinery, we can now define a small $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O}, \mathbb{A})$-model using the Yoneda construction over a small $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model.

Lemma 7.3.8
Given a small $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-model $\mathcal{G}$, we can define a small $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O}, \mathbb{A})$-model $\varphi(\mathcal{G})$.

Proof For the purposes of this proof, we will assume that the categories $\hat{\mathcal{C}}$ and $\hat{\mathcal{S}}$ are the small categories obtained by taking full subcategories of the Yoneda constructions as mentioned earlier.

Firstly, we consider the carrier. Given that the carrier of $\mathcal{G}$ is $(\mathcal{C}, \mathcal{S}, F)$, we make the following definitions. Let $S_{\mathcal{C}} : \hat{\mathcal{C}} \to \hat{\mathcal{C}}_s$ and $U_{\mathcal{C}} : \hat{\mathcal{C}}_s \to \hat{\mathcal{C}}$ be the fully faithful cartesian functors given by the equivalence of $\hat{\mathcal{C}}$ and $\hat{\mathcal{C}}_s$, such that $S;U \simeq \_ : \hat{\mathcal{C}} \to \hat{\mathcal{C}}$. Also, let $S_{\mathcal{S}} : \hat{\mathcal{S}} \to \hat{\mathcal{S}}_s$and $U_{\mathcal{S}} : \hat{\mathcal{S}}_s \to \hat{\mathcal{S}}$ be the fully faithful symmetric monoidal functors given by the equivalence of $\hat{\mathcal{S}}$ and $\mathcal{S}$. Finally, let $K : \hat{\mathcal{S}}_s \to sk(\hat{\mathcal{S}}_s)$ and $I : sk(\hat{\mathcal{S}}_s) \to \hat{\mathcal{S}}_s$ be the fully faithful strict symmetric-monoidal functors given by the equivalence of $\hat{\mathcal{S}}_s$ and $sk(\hat{\mathcal{S}}_s)$.

Now, let the carrier of $\varphi(\mathcal{G})$ be $(\hat{\mathcal{C}}_s, sk(\hat{\mathcal{S}}_s), \hat{F}_s; K, I; G_s)$ where $\hat{F} \dashv G$. In order to show that this has the right form we need just to show that $\hat{F}_s; K \dashv I; G_s$, but this follows from the original adjunction via the diagram of proposition 7.3.6 and the properties of K and I.

We will now define $Y'_{\mathcal{C}} : \mathcal{C} \to \hat{\mathcal{C}}_s$ as $Y_{\mathcal{C}}; S_{\mathcal{C}}$ and $Y'_{\mathcal{S}} : \mathcal{S} \to sk(\hat{\mathcal{S}}_s)$ as $Y_{\mathcal{S}}; S_{\mathcal{S}}; K$.

Now we consider the interpretation functions. Firstly, define $[\![\_]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L}$ as follows:

$$
\begin{aligned}
[\![A \in \mathsf{M}_L]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} &= Y'_{\mathcal{S}}([\![A]\!]^{\mathcal{G}}_{\mathsf{M}_L}) \\
[\![I]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} &= I \\
[\![A \otimes B]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} &= [\![A]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} \otimes [\![B]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} \\
[\![A \multimap B]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} &= [\![A]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} \multimap [\![B]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} \\
[\![!A]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} &= \hat{F}_s G_s [\![A]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L}
\end{aligned}
$$

and define $[\![\_]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_I}$ as follows:

$$\begin{aligned}[\![Q \in \mathsf{M}_I]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_I} &= Y'_{\mathcal{C}}([\![Q]\!]^{\mathcal{G}}_{\mathsf{M}_I}) \\ [\![!A]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_I} &= G_s[\![A]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L}\end{aligned}$$

Now these two definitions satisfy the required property:

$$\hat{F}_s([\![\_]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_I}) = [\![\_]\!]^{\varphi(\mathcal{G})}_{\mathsf{M}^{\mathsf{H}}_L} : \mathsf{M}^{\mathsf{H}}_I \to \hat{\mathcal{S}}_s$$

by virtue of the natural isomorphism:

$$Y'_{\mathcal{C}}; \hat{F}_s; K \simeq Y_{\mathcal{C}}; S_{\mathcal{C}}; \hat{F}_s; K \simeq Y_{\mathcal{C}}; \hat{F}; S_{\mathcal{S}}; K \simeq F; Y_{\mathcal{S}}; S_{\mathcal{S}}; K \simeq F; Y'_{\mathcal{S}}$$

and the fact that since any two isomorphic objects or arrows in $sk(\hat{\mathcal{S}}_s)$ are equal, $Y'_{\mathcal{C}}; \hat{F}_s = F; Y'_{\mathcal{S}}$.

Finally, we need to give the operator interpretation on elements of $\mathcal{O}_L$. Since we know that the isomorphism

$$\begin{aligned}&sk(\hat{\mathcal{S}}_s)(I, [\![\bigotimes_{i=1\ldots r} !(\otimes\vec{Q}_i\vec{A}_i \multimap B_i) \otimes (\bigotimes_{j=1\ldots s} (\otimes\vec{Q}'_j\vec{A}'_j \multimap B'_j)) \multimap (\otimes\vec{Q}''\vec{A}'' \multimap B'')]\!]) \\ \simeq &\mathsf{Nat}_{sk(\hat{\mathcal{S}}_s),\ldots,sk(\hat{\mathcal{S}}_s),\hat{\mathcal{C}}_s}(\prod_{i=1\ldots r} sk(\hat{\mathcal{S}}_s)(F(=) \otimes [\![\vec{Q}_i; \vec{A}_i]\!], [\![B_i]\!]) \\ &\qquad \times \prod_{j=1\ldots s} sk(\hat{\mathcal{S}}_s)(F(=) \otimes \_j \otimes [\![\vec{Q}'_j; \vec{A}'_j]\!], [\![B'_j]\!]), \\ &\qquad\qquad sk(\hat{\mathcal{S}}_s)(F(=) \otimes \_1 \otimes \ldots \otimes \_s \otimes [\![\vec{Q}''; \vec{A}'']\!], [\![B'']\!]))\end{aligned}$$

holds given the interpretations we have already defined, it suffices to show that for each operator we have an instance of the second natural transformation. By consideration of the equivalences, though, these natural transformations arise from natural transformations:

$$\begin{aligned}&\mathsf{Nat}_{\hat{\mathcal{S}},\ldots,\hat{\mathcal{S}},\hat{\mathcal{C}}}(\prod_{i=1\ldots r} \hat{\mathcal{S}}(F(=) \otimes [\![\vec{Q}_i; \vec{A}_i]\!], [\![B_i]\!]) \times \prod_{j=1\ldots s} \hat{\mathcal{S}}(F(=) \otimes \_j \otimes [\![\vec{Q}'_j; \vec{A}'_j]\!], [\![B'_j]\!]), \\ &\qquad \hat{\mathcal{S}}(F(=) \otimes \_1 \otimes \ldots \otimes \_s \otimes [\![\vec{Q}''; \vec{A}'']\!], [\![B'']\!]))\end{aligned}$$

but this set is isomorphic by virtue of the Yoneda embedding to the set:

$$\begin{aligned}&\mathsf{Nat}_{\mathcal{S},\ldots,\mathcal{S},\mathcal{C}}(\prod_{i=1\ldots r} \mathcal{S}(F(=) \otimes [\![\vec{Q}_i; \vec{A}_i]\!], [\![B_i]\!]) \times \prod_{j=1\ldots s} \mathcal{S}(F(=) \otimes \_j \otimes [\![\vec{Q}'_j; \vec{A}'_j]\!], [\![B'_j]\!]), \\ &\qquad \mathcal{S}(F(=) \otimes \_1 \otimes \ldots \otimes \_s \otimes [\![\vec{Q}''; \vec{A}'']\!], [\![B'']\!]))\end{aligned}$$

Hence we have given the model $\varphi(\mathcal{G})$. □

Lemma 7.3.9 (Fully Faithful)
The initiality map $\mathcal{FT}^{\kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))} : \mathcal{G}_T(\mathbb{O},\mathbb{A}) \to \kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))$ is fully faithful in both its component functors.

Proof We can prove this by showing that the pair of the two fully faithful functors $Y'_{\mathcal{C}}$ and $Y'_{\mathcal{S}}$ is a $\mathsf{Lin}(\mathbb{O})$-morphism with the appropriate domain and codomain. This will then imply the result since the initiality map is unique and hence must be this map. In order to check that $(Y'_{\mathcal{C}}, Y'_{\mathcal{S}})$ is a $\mathsf{Lin}(\mathbb{O})$-morphism, we need simply to check that it has the right behaviour with respect to the primitive interpretation and the functor F of the model, both of which hold by virtue of the equality $Y'_{\mathcal{C}}; \hat{F}_s = F; Y'_{\mathcal{S}}$ for a general Yoneda construction, and with respect to the operator interpretation, the result is given by the isomorphisms of lemma 7.3.8. $\square$

Theorem 7 (Conservativity of $(\_)^{\square}$)
If, given two $\Gamma;\Delta$-terms v and w of type A in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, we can derive the equality judgement $\Gamma;\Delta \vdash v^{\square} = w^{\square}\colon A$ in $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$, then we must be able to derive the equality judgement $\Gamma;\Delta \vdash v = w\colon A$ in $\mathsf{Lin}(\mathbb{O},\mathbb{A})$.

Proof Consider the (small) $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$-model $\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A}))$. Given the derivable equality judgement $\Gamma;\Delta \vdash v^{\square} = w^{\square}\colon A$ in $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$, we know that

$$[\![\Gamma;\Delta \vdash v^{\square}\colon A]\!]^{\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A}))} = [\![\Gamma;\Delta \vdash w^{\square}\colon A]\!]^{\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A}))}$$

in $\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A}))$.

Now this means that $[\![\Gamma;\Delta \vdash v\colon A]\!]^{\kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))} = [\![\Gamma;\Delta \vdash w\colon A]\!]^{\kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))}$ in $\kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))$, by lemma 7.3.3. But since we know by lemma 7.3.9 that the s.m. part of the initiality $\mathsf{Lin}(\mathbb{O})$-morphism $\mathcal{FT}^{\kappa(\varphi(\mathcal{G}_T(\mathbb{O},\mathbb{A})))}$ is faithful, this implies that $[\![\Gamma;\Delta \vdash v\colon A]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})} = [\![\Gamma;\Delta \vdash w\colon A]\!]^{\mathcal{G}_T(\mathbb{O},\mathbb{A})}$. This then gives the result by completeness. $\square$

7.4 Corollaries

We note some simple consequences of this result in general. Firstly, we give a lemma:

Lemma 7.4.1
For any linear type theory $\mathsf{Lin}(\mathbb{O}',\mathbb{A}')$ such that the following diagram commutes:

$$\begin{tikzcd}
\mathsf{Lin}(\mathbb{O},\mathbb{A}) \arrow[rr, "(_)^{\square}"] \arrow[dr, "\theta_1"'] & & \mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A}) \\
& \mathsf{Lin}(\mathbb{O}',\mathbb{A}') \arrow[ur, "\theta_2"'] &
\end{tikzcd}$$

and θ_1 and θ_2 are sound, θ_1 is conservative.

Proof Given $\theta_1(\Gamma;\Delta) \vdash \theta_1(\Gamma;\Delta \vdash v{:}A) = \theta_1(\Gamma;\Delta \vdash w{:}A){:}\theta_1(A)$ in $\mathsf{Lin}(\mathbb{O}',\mathbb{A}')$, we have that

$$\theta_2(\theta_1(\Gamma;\Delta)) \vdash \theta_2(\theta_1(\Gamma;\Delta \vdash v{:}A)) = \theta_2(\theta_1(\Gamma;\Delta \vdash w{:}A)){:}\theta_2(\theta_1(A))$$

and hence $\Gamma;\Delta \vdash (v)^{\Box} = (w)^{\Box}{:}A$. But by the conservativity of $(\_)^{\Box}$, this means that $\Gamma;\Delta \vdash v = w{:}A$. $\Box$

This then gives us the most important corollary; that our conservativity result implies that for the more primitive embedding:

Corollary 7.1
The trivial embedding $\mathsf{Lin}(\mathbb{O},\mathbb{A}) \to \mathsf{Lin}^{\mathsf{H}}(\mathbb{O},\mathbb{A})$ is conservative *for any generalised signature* $\mathbb{O}$.

Proof First note that since the map $(\_)^{\Box} : \mathsf{Lin}^{\mathsf{H}}(\mathbb{O},\mathbb{A}) \to \mathsf{Lin}^{\mathsf{HC}}(\mathbb{O},\mathbb{A})$ is sound for signatures having no intuitionistic operators, by our lemma the trivial embedding $\mathsf{Lin}(\mathbb{O},\mathbb{A}) \to \mathsf{Lin}^{\mathsf{H}}(\mathbb{O},\mathbb{A})$ is conservative for such signatures. But now note that any generalised type theory is isomorphic to one having no intuitionistic operators, and furthermore that the following square commutes, where $\mathbb{O}'$ and $\mathbb{A}'$ are the signature with no intuitionistic operators and the associated axiom set:

$$\begin{tikzcd}
\mathsf{Lin}(\mathbb{O}',\mathbb{A}') \arrow[r, "emb"] \arrow[d, leftrightarrow] & \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}',\mathbb{A}') \arrow[d, leftrightarrow] \\
\mathsf{Lin}(\mathbb{O},\mathbb{A}) \arrow[r, "emb"'] & \mathsf{Lin}^{\mathsf{H}}(\mathbb{O},\mathbb{A})
\end{tikzcd}$$

This implies that the trivial embedding is also conservative in the general case where the signature may have intuitionistic operators, since isomorphisms are conservative. $\Box$

Output Natural Operators

We now use this result to show how we can represent any output-natural set of operators in a linear type-theory as a single constant in the higher-order extension of the type-theory. The idea behind this is best seen through an example. Consider the output-natural set of operators O with arity

$$\frac{(\vec{Q};\vec{A})B \quad ; \quad (\vec{Q}'_1;\vec{A}'_1)B'_1 \quad (\vec{Q}'_2;\vec{A}'_2)(\_)}{(\vec{Q}'';\vec{A}'')(\_)}$$

We claim that this can be represented in a higher-order setting by the constant c_{O} with arity:

$$!(\otimes\vec{Q}\vec{A} \multimap B) \otimes (\otimes\vec{Q}'_1\vec{A}'_1 \multimap B_1) \multimap (\otimes\vec{Q}''\vec{A}'' \multimap \otimes\vec{Q}'_2\vec{A}'_2)$$

In order to see how this works, consider the translation of a general operator instance:

$$\begin{aligned}
&\mathsf{O}_C((\vec{x};\vec{y})v; (\vec{x}'_1;\vec{y}'_1)w_1, (\vec{x}'_2;\vec{y}'_2)w_2)(\vec{v}'\vec{w}') \\
\mapsto &\mathsf{let}\ \otimes z\vec{y}'_2\ \mathsf{be}\ (c_{\mathsf{O}}!(\lambda z'.\mathsf{let}\ \otimes\vec{x}''\vec{y}\ \mathsf{be}\ z'\ \mathsf{in}\ \mathsf{let}\ \vec{x}\ \mathsf{be}\ \vec{x}''\ \mathsf{in}\ v) \otimes \\
&\qquad\qquad (\lambda z'.\mathsf{let}\ \otimes\vec{x}''\vec{y}'_1\ \mathsf{be}\ z'\ \mathsf{in}\ \mathsf{let}\ \vec{x}'_1\ \mathsf{be}\ \vec{x}''\ \mathsf{in}\ w_1)(\otimes\vec{v}'\vec{w}')) \\
&\quad \mathsf{in}\ w_2
\end{aligned}$$

We can clearly see that this term has output-natural behaviour inherited from that of the let -construct. Now consider the arity of the constant c_{O_C} which represents the operator O_C in the standard encoding given earlier in this chapter:

$$!(\otimes\vec{Q}\vec{A} \multimap B) \otimes (\otimes\vec{Q}'_1\vec{A}'_1 \multimap B_1) \otimes (\otimes\vec{Q}'_2\vec{A}'_2 \multimap B_2) \multimap (\otimes\vec{Q}''\vec{A}'' \multimap C)$$

We can now express c_n in terms of this set of constants as follows:

$$c_n \mapsto \lambda x.c_{\mathsf{O}_{\otimes\vec{Q}'_2\vec{A}'}}(x \otimes (\lambda y.y))$$

and also, when we compose the mappings taking the output-natural set of operators to the (system with the) constant c_n and taking the (system with the) constant c_n to the (standard higher-order system with) constants c_{O_C}, the results are the same up to provable equality, by a simple application of the output-naturality axioms.

To state this formally, given a higher-order signature $(\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L \cup \mathsf{O})$ where the set of operators O is output-parameterised *over the set* M_L and has arity:

$$\frac{(\vec{Q}_1;\vec{A}_1)B_1, \ldots, (\vec{Q}_r;\vec{A}_r)(B_r); (\vec{Q}'_1;\vec{A}'_1)B'_1, \ldots, (\vec{Q}'_{s-1};\vec{A}'_{s-1})B'_{s-1}, (\vec{Q}'_s;\vec{A}'_s)(\_)}{(\vec{Q}'';\vec{A}'')(\_)}$$

we say that the *output-natural-in-*O *operator constant set*, written $(\mathcal{O}_L \cup \mathsf{O})^{\mathsf{OC}}$ is defined as $\mathcal{O}_L^{\mathsf{C}} \cup \{c_{\mathsf{O}}\}$, where c_{O} is a constant with arity:

$$(\bigotimes_{i=1\ldots r} !(\otimes\vec{Q}_i\vec{A}_i \multimap B_i) \otimes \bigotimes_{j=1\ldots(s-1)} (\otimes\vec{Q}'_j\vec{A}'_j \multimap B'_j)) \multimap (\otimes\vec{Q}''\vec{A}'' \multimap \otimes\vec{Q}'_s\vec{A}'_s)$$

Now, we have as before that given a higher-order signature $\mathbb{H} = (\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L)$ such that $\mathcal{O}_L$ contains an output-parameterised-in-M_L set of operators,

$$\mathbb{H}^{\mathsf{OC}}_{\mathsf{O}}(\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}^{\mathsf{OC}}_L)$$

is another higher-order signature. Write $\mathsf{Lin}^{\mathsf{HC}}_{\mathsf{O}}(\mathbb{H})$ for the higher-order typing system over this higher-order signature $\mathbb{H}^{\mathsf{OC}}_{\mathsf{O}}$.

Definition 7.4.2 (The Translation $(\_)^{\blacksquare}$)
Define the translation $(\_)^{\blacksquare}$ from $\mathsf{Lin}^{\mathsf{H}}(\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L)$ to $\mathsf{Lin}^{\mathsf{H}}(\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}^{\mathsf{OC}}_L)$as the identity on types and on pre-terms as follows:

$$(x)^{\blacksquare} = x$$

$$(\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w)^{\blacksquare} = \mathsf{let}\ x\ \mathsf{be}\ (v)^{\blacksquare}\ \mathsf{in}\ (w)^{\blacksquare}$$

$$\begin{array}{l}
(O'((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(\vec{v}';\vec{w}'))^{\blacksquare} = \\
\qquad c_O(\otimes_{i=1\ldots r}!(\lambda z.\mathsf{let}\ \otimes\vec{x}''_i\vec{y}_i\ \mathsf{be}\ z\ \mathsf{in}\ \mathsf{let}\ \vec{x}_i\ \mathsf{be}\ \vec{x}''_i\ \mathsf{in}\ (v_i)^{\blacksquare})) \\
\qquad\quad \otimes(\otimes_{j=1\ldots s}(\lambda z.\mathsf{let}\ \otimes\vec{x}'''_j\vec{y}'_j\ \mathsf{be}\ z\ \mathsf{in}\ \mathsf{let}\ \vec{x}'_j\ \mathsf{be}\ \vec{x}'''_j\ \mathsf{in}\ (w_j)^{\blacksquare})) \quad (\otimes(\vec{v}'\vec{w}')^{\blacksquare}) \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{on operators } O' \notin \mathsf{O}
\end{array}$$

$$\begin{array}{l}
(\mathsf{O}_C((\vec{x}_1;\vec{y}_1)v_1,\ldots,(\vec{x}_r;\vec{y}_r)v_r;(\vec{x}'_1;\vec{y}'_1)w_1,\ldots,(\vec{x}'_s;\vec{y}'_s)w_s)(\vec{v}';\vec{w}'))^{\blacksquare} = \\
\quad \mathsf{let}\ \otimes\vec{z}\vec{y}'_s\ \mathsf{be}\ c_n(\otimes_{i=1\ldots r}!(\lambda z.\mathsf{let}\ \otimes\vec{x}''_i\vec{y}_i\ \mathsf{be}\ z\ \mathsf{in}\ \mathsf{let}\ \vec{x}_i\ \mathsf{be}\ \vec{x}''_i\ \mathsf{in}\ (v_i)^{\blacksquare})) \\
\qquad\qquad (\otimes_{j=1\ldots(s-1)}(\lambda z.\mathsf{let}\ \otimes\vec{x}'''_j\vec{y}'_j\ \mathsf{be}\ z\ \mathsf{in}\ \mathsf{let}\ \vec{x}'_j\ \mathsf{be}\ \vec{x}'''_j\ \mathsf{in}\ (w_j)^{\blacksquare})) \\
\qquad\qquad\qquad (\otimes(\vec{v}'\vec{w}')^{\blacksquare}) \\
\qquad \mathsf{in}\ \mathsf{let}\ \vec{x}'_s\ \mathsf{be}\ \vec{z}\ \mathsf{in}\ w_s
\end{array}$$

We can extend the typing system $\mathsf{Lin}^{\mathsf{HC}}_{\mathsf{O}}(\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L)$ to a type theory by letting $\mathsf{Lin}^{\mathsf{HC}}_{\mathsf{O}}((\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}_L), \mathbb{A})$ be defined as the higher-order type theory

$$\mathsf{Lin}^{\mathsf{H}}((\mathsf{M}_I, \mathsf{M}_L, \emptyset, \mathcal{O}^{\mathsf{OC}}_L), (\mathbb{A})^{\blacksquare})$$

This translation can be shown to be easily invertible.

7.5 Action Calculi

We now consider the action calculi we have defined in the light of our conservativity result. First we need to define another translation.

Definition 7.5.1 (The Translation ι_5)

Define the translation $\iota_5 : \mathsf{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K}) \to \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}})$ as follows:

$$
\begin{aligned}
\iota_5(\boxtimes m) &= !(\iota_5(m)) \\
\iota_5(l \multimap m) &= \iota_5(l) \multimap \iota_5(m) \\
\iota_5(m \otimes l) &= \iota_5(m) \otimes \iota_5(l)
\end{aligned}
$$

$$
\begin{aligned}
\iota_5(x) &= x \\
\iota_5(\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w) &= \mathsf{let}\ x\ \mathsf{be}\ \iota_5(v)\ \mathsf{in}\ \iota_5(w) \\
\iota_5(\otimes(\vec{v})) &= \otimes\iota_5(\vec{v}) \\
\iota_5(\mathsf{let}\ \otimes\vec{x}\ \mathsf{be}\ v\ \mathsf{in}\ w) &= \mathsf{let}\ \otimes\vec{x}\ \mathsf{be}\ \iota_5(v)\ \mathsf{in}\ \iota_5(w) \\
\iota_5(\mathsf{K}((\vec{x}_1)v_1, \dots, (\vec{x}_r)v_r, w_1 \dots w_s)) &= \mathsf{K}((\vec{x}_1)\iota_5(v_1), \dots, (\vec{x}_r)\iota_5(v_r), \iota_5(w_1) \dots \iota_5(w_s)) \\
\iota_5(\mathsf{code}(v)) &= !\iota_5(v) \\
\iota_5(\mathsf{decode}(v)) &= \mathsf{i}\iota_5(v) \\
\iota_5(\lambda x{:}l.v) &= \lambda x{:}l.\iota_5(v) \\
\iota_5(vw) &= \iota_5(v)\iota_5(w)
\end{aligned}
$$

This translation is easily shown to be sound. Further, we have the following translation lemma:

Lemma 7.5.2 (AC Translations)

The following diagram of translations commutes:

$$
\begin{tikzcd}
 & \mathsf{Lin}^{\mathsf{A}^{\boxtimes}}(\mathbb{K}) \arrow[dl, "\iota_3"'] \arrow[d, "\iota_2"] & \mathsf{Lin}^{\mathsf{A}}(\mathbb{K}) \arrow[l, "\iota_1"'] \arrow[d, "emb"] & =\mathsf{Lin}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}}) \\
\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K}) \arrow[r, "\iota_4"'] & \mathsf{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K}) \arrow[r, "\iota_5"'] & \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}}) &
\end{tikzcd}
$$

Now we can use our conservativity result as follows:

Corollary 7.2

The maps $\iota_1 : \mathsf{Lin}^{\mathsf{A}}(\mathbb{K}) \to \mathsf{Lin}^{\mathsf{A}^{\boxtimes}}(\mathbb{K})$, $\iota_1;\iota_2 : \mathsf{Lin}^{\mathsf{A}}(\mathbb{K}) \to \mathsf{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K})$ and $\iota_1;\iota_3 : \mathsf{Lin}^{\mathsf{A}}(\mathbb{K}) \to \mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$ are all conservative. Equivalently, the embeddings $\mathsf{AC}(\mathbb{K}) \to \mathsf{AC}_{\boxtimes}(\mathbb{K})$, $\mathsf{AC}(\mathbb{K}) \to \mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$ and $\mathsf{AC}(\mathbb{K}) \to \mathsf{AC}_{\Rightarrow}(\mathbb{K})$ are all conservative.

Proof The first statement is immediate by lemma 7.4.1, the diagram of the previous lemma and corollary 7.1 of the conservativity theorem. The second follows by the isomorphisms between the various action calculi and the corresponding linear type theories. $\square$

7.6 Linear Logic

We now show that we can present the type theory of linear logic, in the form of DILL, as an instance of our higher-order generalised linear type theory. This demonstrates that our notion of general linear type theory captures at least the notion of linearity found in the canonical example, linear logic.

Consider an arbitrary DILL-signature $\mathbb{C} = (\mathsf{P}_L, \mathcal{C})$. Let $\mathcal{O}^{\mathcal{C}}_L$ be the set of operators just consisting of the constants $\mathcal{C}$ and their types. Now define the higher-order signature $\mathbb{H}^{\mathbb{C}} = (\emptyset, \mathsf{P}_L, \emptyset, \mathcal{O}^{\mathcal{C}}_L)$. We claim that the higher-order type theory $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}^{\mathbb{C}}, \emptyset)$, which we will refer to as $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, is isomorphic to $\mathsf{DILL}(\mathbb{C})$. Having now defined the type-theory $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, we proceed to translate the type-theory $\mathsf{DILL}(\mathbb{C})$ into $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, and show the soundness of this translation.

Translating $\mathsf{DILL}(\mathbb{C})$ to $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$

We define a map $(\_)^\circ$ which will take a pair of a sequence of variables and a pre-term of $\mathsf{DILL}(\mathbb{C})$, and return a pre-term of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$. The intuition behind this translation is that whenever the pre-term t is a $\Gamma;\Delta$-term such that $\Gamma = \vec{y}\colon\vec{A}$, the translation on that pre-term and sequence $\vec{y}$ and the pre-term t will give a $\vec{y}\,{:}!\vec{A};\Delta$-term of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$.

Definition 7.6.1 (The Translation $(\_)^\circ$)
We define the translation $(\_)^\circ$, which takes pairs of a sequence of distinct variables and a pre-term of $\mathsf{DILL}(\mathbb{C})$, and returns a pre-term of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, as follows:

$$
\begin{aligned}
(\vec{y}x\vec{y}', x)^\circ &= \mathsf{i}x \\
(\vec{y}, x)^\circ &= x \text{ if } x \notin \vec{y} \\
(\vec{y}, c)^\circ &= c \\
(\vec{y}, *)^\circ &= * \\
(\vec{y}, \mathsf{let}\ * \ \mathsf{be}\ t\ \mathsf{in}\ u)^\circ &= \mathsf{let}\ *\ \mathsf{be}\ (\vec{y}, t)^\circ\ \mathsf{in}\ (\vec{y}, u)^\circ \\
(\vec{y}, t \otimes u)^\circ &= (\vec{y}, t)^\circ \otimes (\vec{y}, u)^\circ \\
(\vec{y}, \mathsf{let}\ x \otimes x'\ \mathsf{be}\ t\ \mathsf{in}\ u)^\circ &= \mathsf{let}\ x \otimes x'\ \mathsf{be}\ (\vec{y}, t)^\circ\ \mathsf{in}\ (\vec{y}, u)^\circ \\
(\vec{y}, \lambda x.t)^\circ &= \lambda x.(\vec{y}, t)^\circ \\
(\vec{y}, tu)^\circ &= (\vec{y}, t)^\circ(\vec{y}, u)^\circ \\
(\vec{y}, !t)^\circ &= !(\vec{y}, t)^\circ \\
(\vec{y}, \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ u)^\circ &= \mathsf{let}\ x\ \mathsf{be}\ (\vec{y}, t)^\circ\ \mathsf{in}\ (\vec{y}x, u)^\circ
\end{aligned}
$$

It is now easy to show that for terms $\vec{x}\colon\Gamma;\Delta \vdash t\colon A$ of $\mathsf{DILL}(\mathbb{C})$, we have the typing judgement $\vec{x}\,{:}!\Gamma;\Delta \vdash (\vec{x}, t)^\circ\colon A$ in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$. We give two lemmas to show the action of the translation on substitutions.

Lemma 7.6.2 (Intuitionistic Substitution)
Given a $\Gamma; \_$-term t of type A in $\mathsf{DILL}(\mathbb{C})$ and a $x\!:\!A, \Gamma; \Delta$-term u of type B in $\mathsf{DILL}(\mathbb{C})$, we have the following derivable equality judgement in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$:

$$\vec{y} :!\vec{A'}; \Delta \vdash (\vec{y}, u\{t/x\})^\circ = (\vec{y}x, u)^\circ\{!(\vec{y}, t)^\circ/x\}\!:\!B$$

where $\Gamma = \vec{y}\!:\!\vec{A'}$.

Proof This is proved by induction over the structure of the $x\!:\!A, \Gamma; \Delta$-term u of type B in $\mathsf{DILL}(\mathbb{C})$. We give the key cases.

Firstly assume that u is an intuitionistic variable x (and hence that $A = B$). Then $(\vec{y}x, x)^\circ = \text{¡}x$, and so

$$(\vec{y}x, x)^\circ\{!(\vec{y}, t)^\circ/x\} = \text{¡}!(\vec{y}, t)^\circ$$

and $\vec{y}\!:\!\vec{A'}; \_ \vdash (\vec{y}, t)^\circ = \text{¡}!(\vec{y}, t)^\circ :!A$, so that the result holds.

Secondly assume that u is a linear variable y'. Then $(\vec{y}x, y')^\circ = y'$, and so

$$(\vec{y}x, y')^\circ\{!(\vec{y}, t)^\circ/x\} = y'$$

and $\vec{y}\!:\!\vec{A'}; y'\!:\!C \vdash (\vec{y}, y'\{t/x\}) = y'\!:\!C$ so that we have the result. The inductive cases follow easily. □

Lemma 7.6.3 (Linear Substitution)
Given a $\Gamma; \Delta_1$-term t of type A in $\mathsf{DILL}(\mathbb{C})$ and a $\Gamma; \Delta_2, x\!:\!A$-term u of type B in $\mathsf{DILL}(\mathbb{C})$, we have the following derivable equality judgement in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$:

$$\vec{y} :!\vec{A}; \Delta \vdash (\vec{y}, u\{t/x\})^\circ = (\vec{y}, u)^\circ\{(\vec{y}, t)^\circ/x\}\!:\!B$$

where $\Gamma = \vec{y}\!:\!\vec{A}$ and $\Delta = \Delta_1 \# \Delta_2$.

The proof of this lemma is very similar to that of the last, but simpler. It again goes by induction over the term u.

We can now prove that the translation is sound:

Lemma 7.6.4 (Soundness of $(\_)^\circ$)
Given an equality judgement $\vec{y}\!:\!\Gamma; \Delta \vdash t = u\!:\!A$ of $\mathsf{DILL}(\mathbb{C})$, we have an equality judgement $\vec{y} :!\Gamma; \Delta \vdash (\vec{y}, t)^\circ = (\vec{y}, u)^\circ\!:\!A$ in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$.

Proof The proof is by considering the structure of equality judgements in $\mathsf{DILL}(\mathbb{C})$. Clearly the reflexivity, transitivity, symmetry and context rules are translated to the corresponding rules of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$. The $\beta\eta$-equalities of $\mathsf{DILL}(\mathbb{C})$ are translated to the corresponding equalities of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, which is clear except in the case of the !-$\beta\eta$ equalities; in the case of $! - \beta$, we have the typed equality in $\mathsf{DILL}(\mathbb{C})$:

$$\vec{y}{:}\Gamma;\Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ !t\ \mathsf{in}\ u = u\{t/x\}{:}B$$

In $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, we have $(\vec{y}, \mathsf{let}\ !x\ \mathsf{be}\ !t\ \mathsf{in}\ u)^\circ = \mathsf{let}\ x\ \mathsf{be}\ !(\vec{y},t)^\circ\ \mathsf{in}\ (\vec{y}x,u)^\circ$. If $(\vec{y},t)^\circ = v$ and $(\vec{y},u)^\circ = w$, then we can derive the equality judgement:

$$\vec{y}\ {:}!\Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ !v\ \mathsf{in}\ w = w\{!v/x\}{:}B$$

since ! is intuitionistic, but now by our lemma we have the equality judgement:

$$\vec{y}\ {:}!\Gamma;\Delta \vdash (\vec{y}, u\{t/x\})^\circ = w\{!v/x\}{:}B$$

In the case of $! - \eta$, we have the typed equality in $\mathsf{DILL}(\mathbb{C})$:

$$\vec{y}{:}\Gamma;\Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ !x = t\ {:}!A$$

In $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, we have $(\vec{y}, \mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ !x)^\circ = \mathsf{let}\ x\ \mathsf{be}\ (\vec{y},t)^\circ\ \mathsf{in}\ !\mathsf{i}x$. Now using the $! - \eta$ equality of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ and the let -rule, we have the equality judgement:

$$!\Gamma;\Delta \vdash \mathsf{let}\ x\ \mathsf{be}\ (\vec{y},t)^\circ\ \mathsf{in}\ !\mathsf{i}x = (\vec{y},t)^\circ\ {:}!A$$

as required.

Finally, we have to consider the commuting conversions. These are provable using output-naturality and lemma 7.6.3. $\square$

Translating $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ to $\mathsf{DILL}(\mathbb{C})$

We now define a map $(\_)^\bullet$ from the type-theory $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ into the type theory $\mathsf{DILL}(\mathbb{C})$, along exactly the same lines as in the previous subsection.

Definition 7.6.5 (The Translation $(\_)^\bullet$)
As before, we define the translation $(\_)^\bullet$, which takes a pair of a sequence of distinct variables and a pre-term of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, and gives a pre-term of $\mathsf{DILL}(\mathbb{C})$, as

follows:

$$
\begin{aligned}
(\vec{y}x\vec{y}', x)^\bullet &= !x \\
(\vec{y}, x)^\bullet &= x \text{ if } x \notin \vec{y} \\
(\vec{y}, \mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w)^\bullet &= \mathsf{let}\ !x\ \mathsf{be}\ (\vec{y}, v)^\bullet\ \mathsf{in}\ (\vec{y}x, w)^\bullet \\
(\vec{y}, c)^\bullet &= c \\
(\vec{y}, *)^\bullet &= * \\
(\vec{y}, \mathsf{let}\ *\ \mathsf{be}\ v\ \mathsf{in}\ w)^\bullet &= \mathsf{let}\ *\ \mathsf{be}\ (\vec{y}, v)^\bullet\ \mathsf{in}\ (\vec{y}, w)^\bullet \\
(\vec{y}, v \otimes w)^\bullet &= (\vec{y}, v)^\bullet \otimes (\vec{y}, w)^\bullet \\
(\vec{y}, \lambda x.v)^\bullet &= \lambda x.(\vec{y}, v)^\bullet \\
(\vec{y}, vw)^\bullet &= (\vec{y}, v)^\bullet (\vec{y}, w)^\bullet \\
(\vec{y}, !v)^\bullet &= !(\vec{y}, v)^\bullet \\
(\vec{y}, \text{¡}v)^\bullet &= \mathsf{let}\ !x\ \mathsf{be}\ (\vec{y}, v)^\bullet\ \mathsf{in}\ x
\end{aligned}
$$

We can now easily show that if $\vec{y} :!\Gamma; \Delta \vdash v : A$ in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, we have the typing judgement $\vec{y}:\Gamma; \Delta \vdash (\vec{y}, v)^\bullet : A$ in $\mathsf{DILL}(\mathbb{C})$.

We have the following lemmas on substitution:

Lemma 7.6.6 (Intuitionistic Substitution)
Given a $\Gamma; \_$-term v of type A in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ and a $\Gamma, x : A; \Delta$-term w of type B in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, we have the derivable equality judgement:

$$\vec{y}:\vec{A'}; \Delta \vdash (\vec{y}, w\{v/x\})^\bullet = (\vec{y}x, w)^\bullet \{(\vec{y}, v)^\bullet / !x\} : B$$

where $\Gamma = \vec{y}; !\vec{A'}$.

We can see from the definition of the translation that the variable x always occurs in a subterm of the form $!x$ in the pre-term $(\vec{y}x, v)^\bullet$.

Lemma 7.6.7 (Linear Substitution)
Given a $\Gamma; \Delta_1$-term v of type A in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ and a $\Gamma; \Delta_2, x : A$-term w of type B in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, we have the derivable equality judgement:

$$\vec{y}:\vec{A}; \Delta \vdash (\vec{y}, w\{v/x\})^\bullet = (\vec{y}, w)^\bullet \{(\vec{y}, v)^\bullet / x\} : B$$

where $\Delta = \Delta_1 \# \Delta_2$ and $\Gamma = \vec{y} :!\vec{A}$.

Now we can prove that the translation is sound.

Lemma 7.6.8 (Soundness of $(\_)^\bullet$)
Given an equality judgement $\vec{y} :!\Gamma; \Delta \vdash v = w : A$ in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, we have an equality judgement $\vec{y}:\Gamma; \Delta \vdash (\vec{y}, v)^\bullet = (\vec{y}, w)^\bullet$ in $\mathsf{DILL}(\mathbb{C})$.

Proof Again the proof is by considering the structure of equality judgements if $\mathsf{Lin^D}(\mathbb{C})$. Again, the symmetry, reflexivity, transitivity and context rules are easily seen to be sound under the translation. Further, the $F - \beta_v$ rule is sound since firstly, if the intuitionistic term is a variable, we have:

$$(\vec{y}y', \mathsf{let}\ x\ \mathsf{be}\ y'\ \mathsf{in}\ w)^\bullet = \mathsf{let}\ !x\ \mathsf{be}\ !y'\ \mathsf{in}\ (\vec{y}, w)^\bullet$$

and secondly, for the case where the intuitionistic term is an instance of !, we have that

$$(\vec{y}, \mathsf{let}\ x\ \mathsf{be}\ !v\ \mathsf{in}\ w)^\bullet = \mathsf{let}\ !x\ \mathsf{be}\ !(\vec{y}, v)^\bullet\ \mathsf{in}\ (\vec{y}x, w)^\bullet$$

But then we have that in $\mathsf{DILL}(\mathbb{C})$,

$$\Gamma; \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ !(\vec{y}, v)^\bullet\ \mathsf{in}\ (\vec{y}x, w)^\bullet = (\vec{y}x, w)^\bullet\{(\vec{y}, v)^\bullet/x\}$$

However, the image of the right-hand side of the σ-equality is $(\vec{y}x, w)^\bullet\{!(\vec{y}, v)^\bullet/!x\}$ and since x always occurs in a subterm of the form $!x$ in the image of $(\vec{y}x, w)$, this is equal to the translation of the left-hand side.

Now for the $F - \eta$ rule we have that

$$(\vec{y}, \mathsf{let}\ x\ \mathsf{be}\ w\ \mathsf{in}\ x)^\bullet = \mathsf{let}\ !x\ \mathsf{be}\ (\vec{y}, w)^\bullet\ \mathsf{in}\ !x$$

and from these definitions the required typed equality judgements can be shown. The other let -rules are instances of commuting conversions which hold of the $\mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ u$-construct in $\mathsf{DILL}(\mathbb{C})$.

As for the output naturality equalities for the operators I^L and $\otimes^L_{A,B}$, these are soundly mapped into $\mathsf{DILL}(\mathbb{C})$ by virtue of the commuting conversions of $\mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ u$ and $\mathsf{let}\ x \otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u$.

This leaves us just with the $\beta\eta$-equalities. Those for I, $\otimes$ and λ are translated directly to their counterparts in $\mathsf{DILL}(\mathbb{C})$. For the $! - \beta$ equality of $\mathsf{Lin^D}(\mathbb{C})$, we have:

$$\vec{y} :!\Gamma; \Delta \vdash \mathsf{i}!v = v{:}A$$

Now $(\vec{y}, \mathsf{i}!v)^\bullet = \mathsf{let}\ !x\ \mathsf{be}\ !(\vec{y}, v)^\bullet\ \mathsf{in}\ x$ and so we have the required equality judgement:

$$\vec{y}{:}\Gamma; \Delta \vdash \mathsf{let}\ !x\ \mathsf{be}\ !(\vec{y}, v)^\bullet\ \mathsf{in}\ x = (\vec{y}, v)^\bullet{:}A$$

by virtue of the $! - \beta$ equality of $\mathsf{DILL}(\mathbb{C})$. For the $! - \eta$ equality of $\mathsf{Lin^D}(\mathbb{C})$, we have:

$$x :!A; \_ \vdash !(\mathsf{i}x) = x :!A$$

Now $(x, !(\mathsf{i}x))^\bullet = !(\mathsf{let}\ !x\ \mathsf{be}\ !x\ \mathsf{in}\ x)$ and so we have the required equality judgement:

$$x{:}A; \_ \vdash !(\mathsf{let}\ !x\ \mathsf{be}\ !x\ \mathsf{in}\ x) = !x :!A$$

again by virtue of the $! - \beta$ equality of $\mathsf{DILL}(\mathbb{C})$. $\square$

The Translations are Inverse

Having given translations relating $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ and $\mathsf{DILL}(\mathbb{C})$, we now show that they form an inverse pair.

Lemma 7.6.9 (Inverse Pair)
The translations $(\_)^\circ$ and $(\_)^\bullet$ are inverse up to provable equality, in the sense that for any term of $\mathsf{DILL}(\mathbb{C})$ $\vec{y}{:}\Gamma;\Delta \vdash t{:}A$, we have the equality judgement:

$$\vec{y} : \Gamma;\Delta \vdash (\vec{y},(\vec{y},t)^\circ)^\bullet = t{:}A$$

and for any term of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ $\vec{y} :!\Gamma;\Delta \vdash v{:}A$ we have the equality judgement:

$$\vec{y} :!\Gamma;\Delta \vdash (\vec{y},(\vec{y},v)^\bullet)^\circ = v{:}A$$

Proof We prove this by considering the structure of the pre-terms of the source type theory. Considering firstly $(\vec{y},(\vec{y},t)^\circ)^\bullet$, we have:

$$\begin{array}{ccccc}
(\vec{y}x\vec{y}',(\vec{y}x\vec{y}',x)^\circ)^\bullet & = & (\vec{y}x\vec{y}',\text{¡}x)^\bullet & = & \mathsf{let}\ !x\ \mathsf{be}\ !x\ \mathsf{in}\ x \\
(\vec{y},(\vec{y},x)^\circ)^\bullet & = & (\vec{y},x)^\bullet & = & x \\
(\vec{y},(\vec{y},c)^\circ)^\bullet & = & (\vec{y},c)^\bullet & = & c \\
(\vec{y},(\vec{y},*)^\circ)^\bullet & = & (\vec{y},*)^\bullet & = & * \\
(\vec{y},(\vec{y},\mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ u)^\circ)^\bullet & = & (\vec{y},\mathsf{let}\ *\ \mathsf{be}\ v\ \mathsf{in}\ w)^\bullet & = & \mathsf{let}\ *\ \mathsf{be}\ t'\ \mathsf{in}\ u' \\
(\vec{y},(\vec{y},t\otimes u)^\circ)^\bullet & = & (\vec{y},v\otimes w)^\bullet & = & t'\otimes u' \\
(\vec{y},(\vec{y},\mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u)^\circ)^\bullet & = & (\vec{y},\mathsf{let}\ x\otimes y\ \mathsf{be}\ v\ \mathsf{in}\ w)^\bullet & = & \mathsf{let}\ x\otimes y\ \mathsf{be}\ t'\ \mathsf{in}\ u' \\
(\vec{y},(\vec{y},\lambda x.t)^\circ)^\bullet & = & (\vec{y},\lambda x.v)^\bullet & = & \lambda x.t' \\
(\vec{y},(\vec{y},tu)^\circ)^\bullet & = & (\vec{y},vw)^\bullet & = & t'u' \\
(\vec{y},(\vec{y},!t)^\circ)^\bullet & = & (\vec{y},!v)^\bullet & = & !t' \\
(\vec{y},(\vec{y},\mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ u)^\circ)^\bullet & = & (\vec{y},\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ (\vec{y}x,u)^\circ)^\bullet & = & \mathsf{let}\ !x\ \mathsf{be}\ t'\ \mathsf{in}\ (\vec{y}x,(\vec{y}x,u)^\circ)^\bullet
\end{array}$$

where $v = (\vec{y},t)^\circ$, $w = (\vec{y},u)^\circ$, $t' = (\vec{y},v)^\bullet$ and $u' = (\vec{y},v)^\bullet$. Now we can easily prove the result by induction over the unique derivation of the typing judgement $\vec{y}{:}\Gamma;\Delta \vdash t{:}A$ of $\mathsf{DILL}(\mathbb{C})$, using the $! - \beta$ rule in the case of an intuitionistic axiom, and reflexivity in all other cases.

Now considering $(\vec{y},(\vec{y},v)^\bullet)^\circ$, we have:

$$\begin{array}{ccccc}
(\vec{y}x\vec{y}',(\vec{y}x\vec{y}',x)^\bullet)^\circ & = & (\vec{y}x\vec{y}',!x)^\circ & = & \text{¡}!x \\
(\vec{y},(\vec{y},x)^\bullet)^\circ & = & (\vec{y},x)^\circ & = & x \\
(\vec{y},(\vec{y},\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w)^\bullet)^\circ & = & (\vec{y},\mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ (\vec{y}x,w)^\bullet)^\circ & = & \mathsf{let}\ x\ \mathsf{be}\ v'\ \mathsf{in}\ (\vec{y}x,(\vec{y}x,w)^\bullet)^\circ \\
(\vec{y},(\vec{y},c)^\bullet)^\circ & = & (\vec{y},c)^\circ & = & c \\
(\vec{y},(\vec{y},*)^\bullet)^\circ & = & (\vec{y},*)^\circ & = & * \\
(\vec{y},(\vec{y},\mathsf{let}\ *\ \mathsf{be}\ v\ \mathsf{in}\ w)^\bullet)^\circ & = & (\vec{y},\mathsf{let}\ *\ \mathsf{be}\ t\ \mathsf{in}\ u)^\circ & = & \mathsf{let}\ *\ \mathsf{be}\ v'\ \mathsf{in}\ w' \\
(\vec{y},(\vec{y},v\otimes w)^\bullet)^\circ & = & (\vec{y},t\otimes u)^\circ & = & v'\otimes w' \\
(\vec{y},(\vec{y},\mathsf{let}\ x\otimes y\ \mathsf{be}\ v\ \mathsf{in}\ w)^\bullet)^\circ & = & (\vec{y},\mathsf{let}\ x\otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u)^\circ & = & \mathsf{let}\ x\otimes y\ \mathsf{be}\ v'\ \mathsf{in}\ w' \\
(\vec{y},(\vec{y},\lambda x.v)^\bullet)^\circ & = & (\vec{y},\lambda x.t)^\circ & = & \lambda x.v' \\
(\vec{y},(\vec{y},vw)^\bullet)^\circ & = & (\vec{y},tu)^\circ & = & v'w' \\
(\vec{y},(\vec{y},!v)^\bullet)^\circ & = & (\vec{y},!t)^\circ & = & !v' \\
(\vec{y},(\vec{y},\text{¡}v)^\bullet)^\circ & = & (\vec{y},\mathsf{let}\ !x\ \mathsf{be}\ t\ \mathsf{in}\ x)^\circ & = & \mathsf{let}\ x\ \mathsf{be}\ v'\ \mathsf{in}\ \text{¡}x
\end{array}$$

where $v' = (\vec{y}, t)^\circ$, $w' = (\vec{y}, u)^\circ$, $t = (\vec{y}, v)^\bullet$ and $u = (\vec{y}, v)^\bullet$. Now again we can prove the result by induction over the unique derivation of the typing judgement $\vec{y} :!\Gamma; \Delta \vdash v{:}A$ in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, using the $! - \beta$ rule for the intuitionistic axiom case and using two let -rules in the iv case. $\square$

Induced Semantics

Since we have given semantics for both $\mathsf{DILL}(\mathbb{C})$ and $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, it is natural to ask how these may be related. Since $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ is a higher-order theory, its models are closely related to those of $\mathsf{DILL}(\mathbb{C})$. In fact, it is a trivial observation that;

LEMMA 7.6.10
Any $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-model $\mathcal{H}$ yields a $\mathsf{DILL}(\mathbb{C})$-model.

Proof Clearly we can take the carrier of the $\mathsf{DILL}(\mathbb{C})$-model to be the carrier of the $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-model. Further, the interpretation function on the primitive types P_L is given directly by $[\![\_]\!]_{\mathsf{M}_L}$. Finally, the operator interpretation of the $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-model gives the interpretation of the constants. $\square$

Before extending this to morphisms, we need to define the concept of morphism between two $\mathsf{DILL}(\mathbb{C})$-models.

DEFINITION 7.6.11 (DILL($\mathbb{C}$)-MORPHISM)
A $\mathsf{DILL}(\mathbb{C})$-morphism $\mathcal{F} : \mathcal{L} \to \mathcal{L}'$ between two $\mathsf{DILL}(\mathbb{C})$-models is a pair of functors $(\mathcal{F}_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}', \mathcal{F}_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}')$ such that:

- $\mathcal{F}_{\mathcal{C}}$ is strict cartesian and $\mathcal{F}_{\mathcal{S}}$ is strict monoidal closed,
- the following diagrams commute:

$$\begin{tikzcd} \mathcal{C} \arrow[r, "F"] \arrow[d, "\mathcal{F}_{\mathcal{C}}"'] & \mathcal{S} \arrow[d, "\mathcal{F}_{\mathcal{S}}"] \\ \mathcal{C}' \arrow[r, "F'"] & \mathcal{S}' \end{tikzcd} \qquad \begin{tikzcd} \mathcal{C} \arrow[d, "\mathcal{F}_{\mathcal{C}}"'] & \mathcal{S} \arrow[l, "G"'] \arrow[d, "\mathcal{F}_{\mathcal{S}}"] \\ \mathcal{C}' & \mathcal{S}' \arrow[l, "G'"'] \end{tikzcd}$$

- $\mathcal{F}_{\mathcal{S}}([\![\_]\!]^{\mathcal{L}}_{\mathsf{P}_I}) = [\![\_]\!]^{\mathcal{L}'}_{\mathsf{P}_I} : \mathsf{P}_I \to \mathsf{obj}(\mathcal{S}')$,
- $\mathcal{F}_{\mathcal{S}}([\![c]\!]^{\mathcal{L}}_{\mathsf{C}}) = [\![c]\!]^{\mathcal{L}'}_{\mathsf{C}}$ for each $c{:}A$ in C.

We now call the category of $\mathsf{DILL}(\mathbb{C})$-models and morphisms $\mathsf{Cat}_{\mathsf{DILL}}(\mathbb{C})$.

Now we can extend our previous result to morphisms:

LEMMA 7.6.12
Any $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-morphism $\mathcal{F}$ between two $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-models yields a $\mathsf{DILL}(\mathbb{C})$-morphism between the resultant $\mathsf{DILL}(\mathbb{C})$-models.

Proof Again, it is clear that the $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-morphism has the right form to be a $\mathsf{DILL}(\mathbb{C})$-morphism, and it is only necessary to show that it preserves the interpretation functions, which it must do by virtue of their derivation from the $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-interpretations. $\square$

Now, call the category of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-models and morphisms $\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{D}}}(\mathbb{C})$. In fact, we can show also that any $\mathsf{DILL}(\mathbb{C})$-morphism between two $\mathsf{DILL}(\mathbb{C})$-models arising from $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-models itself arises from a $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-morphism, which implies that we have a functor $\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{D}}}(\mathbb{C}) \to \mathsf{Cat}_{\mathsf{DILL}}(\mathbb{C})$ which is an embedding.

Further, using our translation results we can see that the term $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-model yields the term $\mathsf{DILL}(\mathbb{C})$-model defined in chapter 3, and hence that the term $\mathsf{DILL}(\mathbb{C})$-model is initial up to isomorphism in the full subcategory of $\mathsf{DILL}(\mathbb{C})$-models which arise from $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-models.

Chapter 8

Normal Forms for $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$

Having introduced a general linear type-theory, given its semantics and shown how it has interesting systems as instances, we are now motivated to look more closely at the properties we can prove of the framework in general. In our conservativity result of the previous chapter, we have made one step in this direction, and in this chapter and the next we want to discuss another, larger issue.

The aim of our general linear type-theory is to provide a framework in which the 'programs' (or processes, or functions, etc) of a particular language can be written and equipped with as much typing information as required, and also in which definitional equality between programs can be proved. What we mean by 'definitional equality' is very basic, for example the associativity of sequential composition would be a definitional equality, whereas any kind of β-evaluation of a function would not. Given that the idea of definitional equality is so basic, it seems imperative that we be able to decide whether two terms are definitionally equal or not, and yet with the machinery we have so far this is certainly not generally provable.

We present in this chapter and the next a system of normal forms for the terms of the type-theory $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ which will decide the equality of the system in the particular case when the axiom set $\mathbb{A}$ is empty. We will further extend the system to show that the equality of any system of the form $\mathsf{Lin}(\mathbb{O}, \mathsf{ONat}(\mathbb{O}, \mathsf{O}))$ for some $\mathbb{O}$ can be decided, as can the higher-order system $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \emptyset)$.

These results then have as simple corollaries the decidability of $\mathsf{DILL}(\mathbb{C})$ and of the higher-order action calculi.

8.1 Proof Nets

One of the innovations associated with the introduction of linear logic by Girard [Gir87] was the definition of proof nets, which have since become objects of study in their own right. One key property of proof nets is that two proofs which are equivalent by commuting conversions, which we have already seen in the various linear type-theories we have introduced, are mapped to equal proof nets. Further, many familiar equalities in type-theories such as the $\beta\eta$-equality of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ can be presented as the the symmetric transitive closure of a set of local rewrites, which can then be shown to be confluent.

In this chapter we will define relations, which are a syntactic presentation of proof-nets. Proof-nets have shown themselves to be a very natural language for the proofs of linear logic in particular, as mentioned in the introduction. However, as they are a graphical syntax, it is difficult to formulate and prove rigidly the delicate lemmas involved in proving normalisation and confluence, as we wish to do. Hence rather than working directly with proof-nets, we will briefly present the theory of proof-nets, but present our development in the language of *relations*, which are a non-graphical syntax for proof-nets.

First we outline the theory of nets for the particular case of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, which raises most of the issues.

Fundamentals

Proof-nets for the type-theory $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ differ from those for $\mathsf{ILL}(\mathbb{C})$, just as the type-theories do. The most significant difference is that to reflect the distinction between the linear context and the intuitionistic context, we will define proof-nets for $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ using two "colours" of wire, linear and intuitionistic, which in this monochrome environment will be represented by full and dashed lines respectively.

As a convention, where we want to indicate an arbitrary number of wires at some point in a net, we will use a short dash across the line, as in figure 8.1, which represents a number of intuitionistic wires.

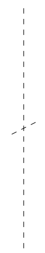
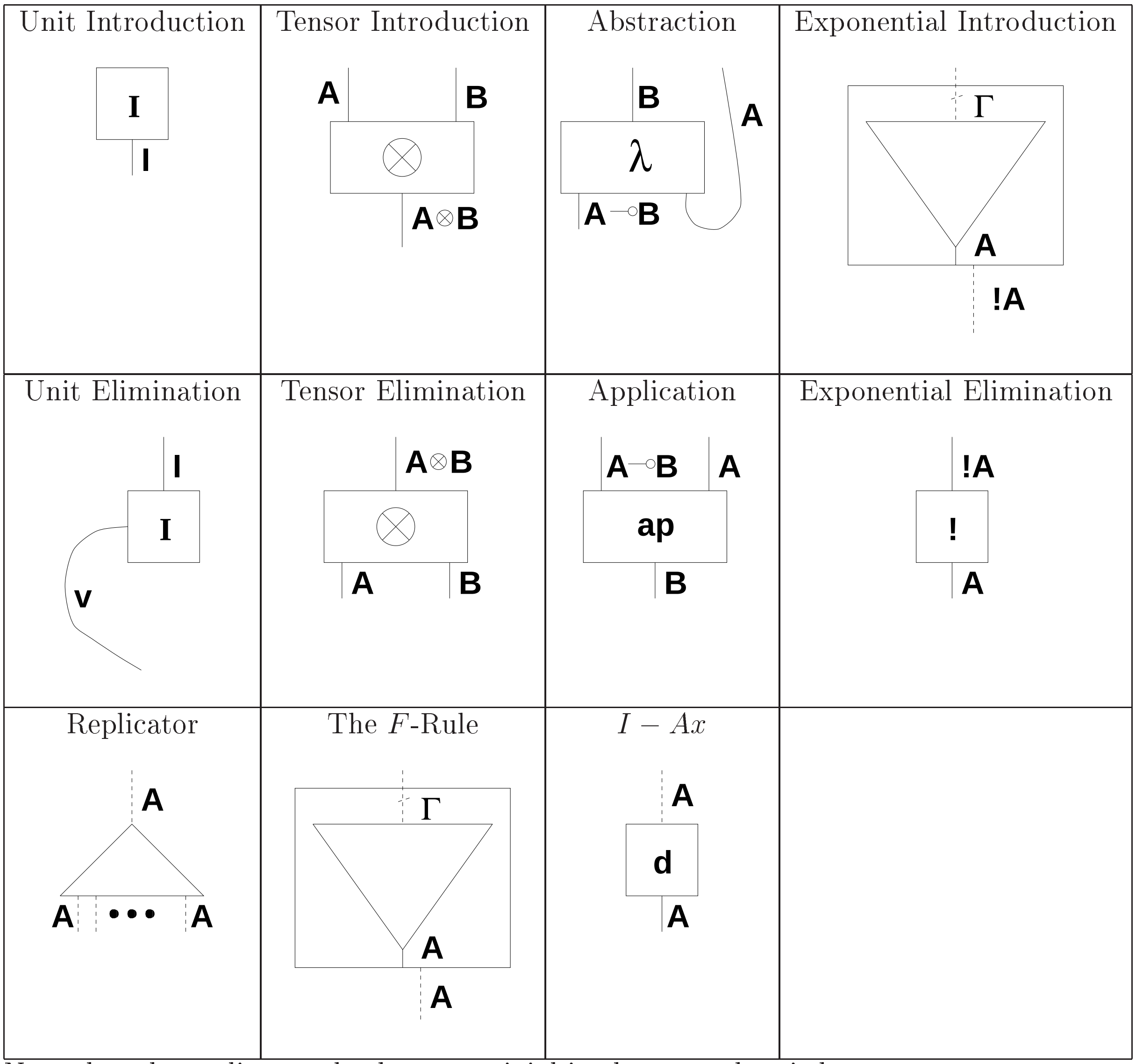


Figure 8.1: Multiple Wires

Now, proof-nets are built out of basic components, which in the case of the type-theory $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ are as given in figure 8.2. Note that each input and output of a component is typed with a type of $\mathsf{DILL}(\mathbb{C})$, except for the unit elimination which has a possibly unexpected output labelled v. This can be seen as a kind of zero width multiple wire; we call it a vestigial, and its rôle will be discussed shortly.



Note that the replicator also has a vestigial in the case when it has no outputs.

Figure 8.2: The Elementary Components

A *net* for $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ is a finite set of components which are connected with wires. Nets correspond to pre-terms in the sense that so far we have imposed no restrictions on the typings of wires, or their colours. Now, a *path* in a net is simply a sequence of components each of which is connected to the next by a wire or a vestigial which is not a *binding* wire (one which is connected to the output port labelled A of an abstraction $A \multimap B$). A *maximal* path is one which cannot be extended, and the *length* of the path is the number of wires it contains. Now, a *proof-net* for $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ is a net for $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ which satisfies a number of conditions:

Colouring Wires must be well-coloured, ie they must be connected to inputs and outputs of the same colour.

Linearity Wires must be linear, ie they be must connected to at most one output and at most one input. If a wire is connected to one output only, then we say it is a free output, and if it is connected to one input only, we say it is a free input.

Typing Wires must be well-typed, ie they must connect inputs and outputs of the same type.

Intuitionistic The net must have only one free output wire, which must be linear.

Boxing Any collection of components occuring inside a box must be a proof-net having no linear free inputs.

Acyclicity There must not exists two components c and c' in the net such that there exist a path from c to c' and a non-zero length path from c' to c.

Abstraction For each abstraction component in the net, all maximal paths starting from the binding output of the abstraction must include the input wire of the same abstraction component.

Most of these conditions are self-explanatory, and we will consider a couple of examples of proof-net constructions to demonstrate their motivation. We assume a function Φ which maps terms of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ to proof-nets for $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$. Firstly consider the proof-net corresponding to an application $\Gamma; \Delta \vdash vw : B$, where we have $\Gamma; \Delta_1 \vdash v : A \multimap B$ and $\Gamma; \Delta_2 \vdash w : A$, and $\Delta = \Delta_1 \# \Delta_2$. The proof net corresponding to the application is seen in figure 8.3.

The key elements are common to many of the constructions. We use a replicator to copy the intuitionistic context for both sub-nets corresponding to v and w, and bind the output wires of both subnets, creating a new free output wire which has the type of the application instance.

Now we can consider the proof-net corresponding to the abstraction $\Gamma; \Delta \vdash \lambda x.v : A \multimap B$ with the obvious derivation, which is seen in figure 8.4.

In this case, the interesting feature is that we have bound the input labelled x of the subnet corresponding to v using an output of the λ-component. This output is called a *binding* output because it must always be used in this way,

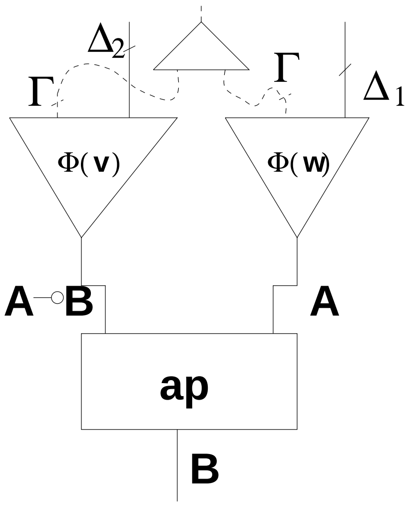


Figure 8.3: Application in Proof Nets

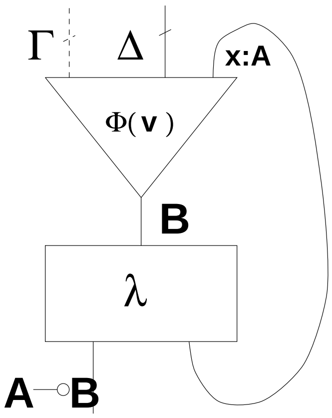


Figure 8.4: Abstraction in Proof Nets

and the acyclicity condition on proof-nets is not violated since such outputs are specifically excluded from being part of paths. Notice that the linear naturality of the λ-operator in $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ is achieved simply by not binding the other linear inputs of the subnet corresponding to v.

Finally, consider the net corresponding to the derivation $\Gamma; \_ \vdash !v :!A$, which can be seen in figure 8.5. The key points about this construction are firstly that we enclose the entire subnet corresponding to v in a 'box' since the !-operator of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$ is not linearly natural with respect to this argument, and secondly that we represent the fact that the !-operator is intuitionistic by allowing it to return an intuitionistic wire as its result. We then need to regain intuitionistic naturality by allowing certain components (intuitionistic operator instances and replicators) to permeate into the box.

The general pattern for the implementation of an arbitrary operator is that we

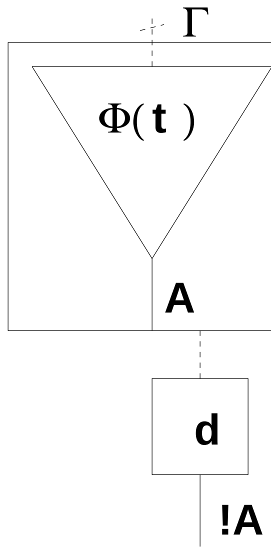


Figure 8.5: !-Introduction in Proof-Nets

enclose all the arguments in which it is not linearly natural in a box, as motivated in the above example, and bind only the variables in the bound positions of those arguments which it is natural in. Finally, if the operator is intuitionistic, then we let the box which necessarily exists have an intuitionistic output.

Vestigials

Having given a brief account of proof nets for this specific case, we have still to explain the role of vestigials. Vestigials are wires which do not correspond to assumptions and have no types, but which are necessary to establish the scoping of elimination constructs having no other outputs. Consider the two derivations $\Gamma; \Delta, x : I \vdash \lambda y : A.(\mathsf{let} * \mathsf{be}\ x\ \mathsf{in}\ v) : A \multimap B$ and $\Gamma; \Delta, x : I \vdash \mathsf{let} * \mathsf{be}\ x\ \mathsf{in}\ (\lambda y : A.v) : A \multimap B$. If we used no vestigials, both of these terms would map to the proof-net in figure 8.6, and hence the scope of the unit-elimination construct would be lost.

Now this particular case is not a problem because in fact the two derivations we have given are equal up to commuting conversion equality. However, there are cases in which this loss of scope causes two terms which are not equal by commuting conversions to have equal images under Φ. One example, which we present, is derived from the coherence problem for symmetric monoidal closed categories, and is due to Kelly and MacLane.

In [KM72] a diagram of natural transformations is presented which does not commute in every symmetric monoidal category. Hence, by virtue of our completeness result for $\mathsf{DILL}(\mathbb{C})$, we would expect that the the diagram does not com-

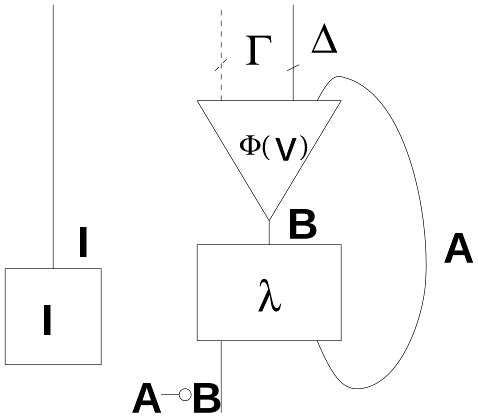


Figure 8.6: Abstraction without vestigials

mute in the term-model, and in particular that the two terms which correspond to the two possible unequal morphisms are not made equal by the type-theory $\mathsf{DILL}(\mathbb{C})$. The first of these two terms is (where we abbreviate $B \multimap I$ as B^I):

$$\_; x{:}A^{I^{I^I}} \vdash \lambda y_1{:}A^{I^I}.\mathsf{let} * \mathsf{be}\ x(\lambda y_2{:}A^I.\mathsf{let} * \mathsf{be}\ y_1(\lambda y_3{:}A.\mathsf{let} * \mathsf{be}\ y_2 y_3\ \mathsf{in} *)){:}A^{I^{I^I}}$$
$$\mathsf{in} *$$
$$\mathsf{in} *$$

The second of the two terms is:

$$\_; x{:}A^{I^{I^I}} \vdash \lambda y_1{:}A^{I^I}.\mathsf{let} * \mathsf{be}\ y_1(\lambda y_3{:}A.\mathsf{let} * \mathsf{be}\ x(\lambda y_2{:}A^I.\mathsf{let} * \mathsf{be}\ y_2 y_3\ \mathsf{in} *)){:}A^{I^{I^I}}$$
$$\mathsf{in} *$$
$$\mathsf{in} *$$

On inspection, it is clear that the first of these two terms equal to the term $\_; x{:}A^{I^{I^I}} \vdash x{:}A^{I^{I^I}}$, as we would expect since one of the morphisms in the Kelly-MacLane diagram is the identity.

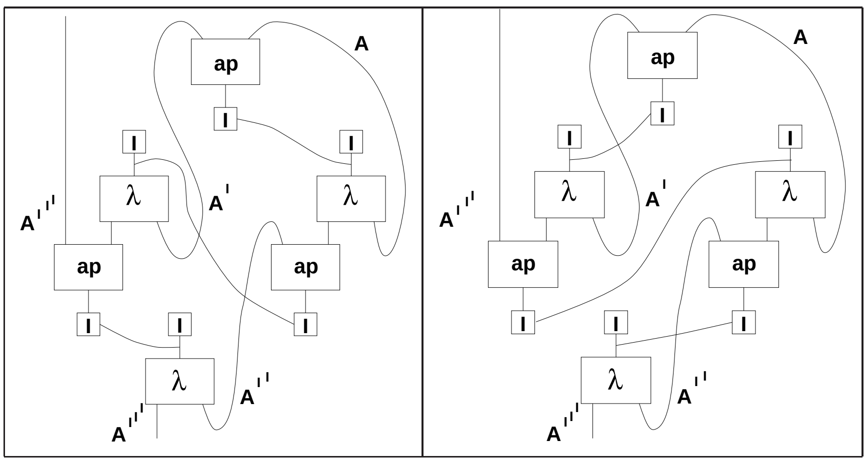


Figure 8.7: Kelly and MacLane's Counterexample

Now, the left-hand proof-net of figure 8.7 represents the first term, and the right-hand one the second. They are not equal as they stand, with vestigials, but we can see that if we were to erase the vestigials from each of the I-elimination boxes, they would be identical. Hence we can see that vestigials are necessary in this framework, to distinguish the proof-nets of unequal proofs.

This result shows that we need vestigials in order to distinguish separate nets. However, one problem with vestigials is that proof-nets with different vestigial positionings (such as the two in figure 8.7) *may* represent terms which are equal via commuting conversions. Hence we must reintroduce a *vestigial equality* on nets. This will be decidable.

Proof-Net Rewrites

One of the key properties of proof-nets is that we can give local rewrites over them which will be confluent and normalising. We show as an example in figure 8.8 the two rewrites which describe the β and η-rules of the arrow type $\multimap$.

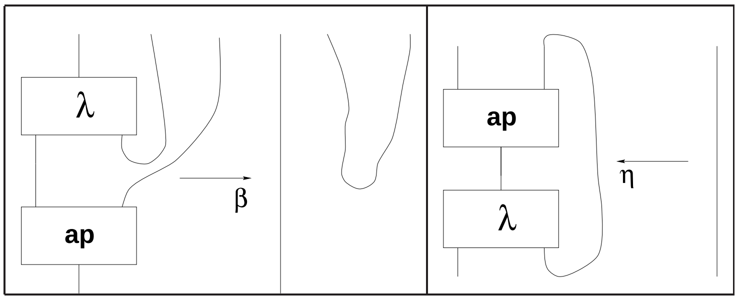


Figure 8.8: The λ-β and η-Rewrites

An interesting point here is that we use an expansionary η-rewrite, following Ghani [Gha95]. If we use η-reductions, we cannot obtain confluence due to a problem with the unit rewrites.

8.2 Relations

We now introduce relations, which are non-graphical analogues of proof-nets for a general linear typing system $\mathsf{Lin}(\mathbb{O})$. Although proof-nets are a very intuitive language for proofs, and capture very effectively the intuitions involved especially in proof equivalence, to prove delicate lemmas relating proof-nets to terms is clumsy, because it involves extended case analysis of graphical situations. In particular, it is easy to overlook unfamiliar cases. Hence, we have chosen to use a non-graphical syntax which, however, closely mirrors the behaviour of proof-nets.

First consider some simple examples. The proof net (for $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$) is given in figure 8.2.

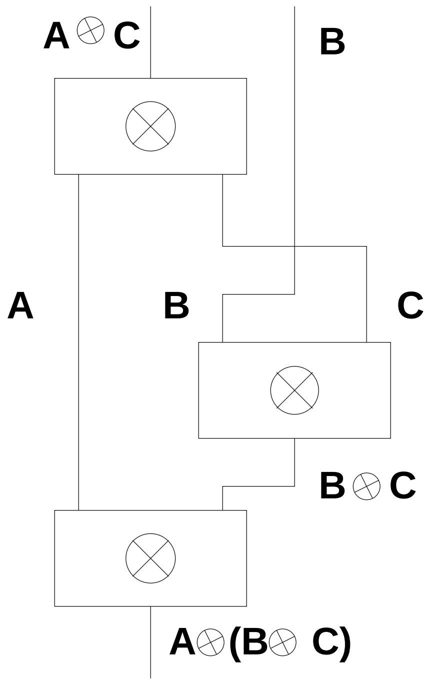


Figure 8.9:

This corresponds to the term

$$\_; x' : A \otimes C, y : B \vdash \mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ x_1 \otimes (y \otimes x_2) : A \otimes (B \otimes C)$$

and the relation

$$R(x', y, z) = \exists x_1 : A, x_2 : C, y' : B \otimes C.(z = x_1 \otimes y') \wedge (y' = y \otimes x_2) \wedge (x_1 \otimes x_2 = x')$$

where by convention we let the (unique) output variable be the last parameter of the relation, in this case z.

As we can see, each rule instance in the proof is reflected in the proof net by a component, in the term by a constructor, and in the relation by a clause. Intuitively, we obtain relations from nets by giving a clause for each component of the net, conjoining them and binding the variables corresponding to internal wires. Now consider the term

$$\_; x' : A \otimes C \vdash \lambda y : B.\mathsf{let}\ x_1 \otimes x_2\ \mathsf{be}\ x'\ \mathsf{in}\ x_1 \otimes (y \otimes x_2) : B \multimap (A \otimes (B \otimes C))$$

which has the relation

$$S(x', z') = \forall y : B. \exists z : (A \otimes (B \otimes C)).(z' y = z) \wedge R(x', y, z)$$

The only extension to our intuition is that in this case the binding variable of the term is bound in the relation using a universal quantifier rather than the existential. This will hold for all binding variables of binding operators.

Now, we can make some simplifications to our syntax. Firstly, note that because of the associativity of the $\wedge$-operator on relations, and the familiar prenex normal forms of the expressions we have written, in general a relation can be written as a sequence of quantifiers followed by the conjunction of a set of clauses:

$$Qx_1.\ldots Qx_r.c_1 \wedge \ldots \wedge c_s$$

Clearly, not all relations of this form represent valid proof-nets. We can exactly characterise those that do, however, as the relations satisfying a certain set of conditions based on the net conditions of page 181.

Note in particular that all relations corresponding to proof nets will mention a variable twice exactly iff it is bound, and once exactly iff it is free. Hence we can decide whether a variable is bound, and since precisely the binding variables of operator clauses are universally quantified, we can reconstruct the sequence of quantifiers from the set of clauses. This procedure may seem to introduce scoping concerns, but we will show by translating our relations soundly to terms (up to commuting conversion equality) that suitable scopes for all binders can be inferred from the clause set.

Therefore, from now on we will give relations as sets of clauses. We will work with the variable set X as usual. Firstly, we construct pre-relations as follows:

Definition 8.2.1 (Pre-Relations)
We define *pre-relations* and *clauses* over a generalised signature $\mathbb{O} = (\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$ mutually inductively as follows:

- Pre-relations, ranged over by $D, E\ldots$ are *finite* sets of clauses, and
- clauses, ranged over by $d, e\ldots$ have one of the following forms, where M is a nonempty finite set of variables and D is a pre-relation:

$$\begin{aligned} d ::=& (\_ =^{x:A} x{:}Q) \mid (M{:}Q = x{:}Q) \mid (x{:}Q = F(x{:}Q)) \mid (F(x{:}Q) = x{:}Q) \\ & \mid x{:}A = (\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})O((\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})D(x{:}A),\ldots,(\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})D(x{:}A); \\ & \qquad (\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})(x{:}A),\ldots,(\vec{x}{:}\vec{Q};\vec{x}{:}\vec{A})(x{:}A)) \end{aligned}$$

where we allow the operator clauses respectively for any operator $O \in \mathcal{O}_L$. We will refer to the variable x in the clause $(\_ =^{x:A} y{:}Q)$ as a *vestigial*.

In this definition, the form $(\vec{x}:\vec{Q};\vec{y}:\vec{A})D(z:B)$ binds the inputs $\vec{x}$ and $\vec{y}$ and the output z in the pre-relation D. The form $(\vec{x}:\vec{Q};\vec{y}:\vec{A})(z:B)$ is similar except that the variables are bound not in a particular pre-relation but in the whole pre-relation containing the clause. To further clarify the *operator clause*, which is the most complex of the definition, we use two familiar examples. First consider the case where we have an operator ! of arity:

$$\frac{(;\_)A \quad ;}{(\_;\_)!A}$$

The operator clause in this case is $(x : !A = ()!(()D(y : A);)$. In this, we take a pre-relation D with output $y{:}A$ and apply the operator to it to give the pre-relation with the same intuitionistic inputs and the one output $x{:}A$. Now consider the operator $\otimes^R$ having arity:

$$\frac{; \quad (\_;\_)A \quad (\_;\_)B}{(\_;\_)A \otimes B}$$

The operator clause here is $(x{:}A \otimes B = () \otimes (;()(y_1{:}A),()(y_2{:}B)))$, and in this case because the operator is linearly natural in the arguments, we do not 'box' them in the syntax but instead, as in proof-nets, allow the operator clause to bind variables elsewhere in the body of the pre-relation it occurs in.

We henceforth omit type information except where necessary, in the familiar way, and further where it is convenient we will omit some details of a general operator clause for brevity; for example, we might write

$$(x = (\vec{y}_1;\vec{y}_2')O(\ldots,(\vec{x}_1';\vec{x}_2')D(x''),\ldots;\ldots))$$

to indicate a general operator clause containing the argument $(\vec{x}_1';\vec{x}_2')D(x'')$.

Clauses are equipped with inputs and outputs (which we will call *polarities*) and typings similarly to the components of proof nets. Each variable in a clause will either occur as an input, an output, a binding occurrence or a vestigial, and will have an associated type (except in the case of vestigials). Intuitively, clauses are analogous to term constructors or equivalently to rules. The first and second clauses in the inductive definition are exceptions to this, in that they correspond to the admissible weakening and contraction rules respectively. We will discuss the need for such explicit syntax later. The third and fourth clauses in the definition correspond respectively to the intuitionistic axiom and I-L rule, and the fifth corresponds to the general operator rule.

Preliminary Definitions

First we define what it means for a pre-relation D to contain a clause.

Definition 8.2.2

We say that a pre-relation D *contains* a clause c (*at depth* r) if $c \in D$ (and $r = 0$) or if for some operator clause:

$$(y = (\vec{y}_1'; \vec{y}_2')O(\ldots(\vec{x}_1'; \vec{x}_2')D'(x'')\ldots;\ldots)) \in D$$

where D' contains c (at depth $r - 1$).

Note that for any pre-relation there is an upper bound on the depth of clauses contained within it.

Having referred informally to clauses having inputs and outputs, we now need to define these notions.

Definition 8.2.3 (Input)

A variable x is an *input* of the clause c if

- either c is $(\_ =^z x)$, $(M = x)$, $(y = F(x))$ or $(F(y) = x)$,
- or c is $(y = (\vec{x}_1; \vec{x}_2)O(\ldots))$ where $x \in \vec{x}_1\vec{x}_2$,
- or c is $(y = (\vec{y}_1'; \vec{y}_2')O(\ldots;\ldots(\vec{x}_1'; \vec{x}_2')(x)\ldots))$
- or c is $(y = (\vec{y}_1'; \vec{y}_2')O(\ldots(\vec{x}_1'; \vec{x}_2')D'(x)\ldots;\ldots))$.

Further, we will say that x *occurs as an input* in a pre-relation D if D contains a clause c of which x is an input.

Definition 8.2.4 (Output)

A variable x is an *output* of the clause c if

- either c is $(M = y)$ where $x \in M$, is $(F(x) = y)$, or is $(x = F(y))$,
- or c is $(x = (\vec{x}_1; \vec{x}_2)O(\ldots;\ldots))$,

Similarly, we will say that x *occurs as an output* in the pre-relation D if a clause c having x as an output is contained in D.

We also define the notion of binding occurrence:

Definition 8.2.5 (Binder)

A variable x is a *binder* in a clause c if

- c is $(x = (\vec{x}_1'; \vec{x}_2')O(\ldots;\ldots(\vec{y}_1; \vec{y}_2)(x'')\ldots))$ where $x \in \vec{y}_1\vec{y}_2$,
- or c is $(y = (\vec{y}_1'; \vec{y}_2')O(\ldots(\vec{x}_1'; \vec{x}_2')D'(x'')\ldots;\ldots))$ where $x \in \vec{x}_1'\vec{x}_2'$.

Again, a variable x *occurs as a binder* in a pre-relation D if there is a clause contained in D which has x as a binder.

A variable x occurs as a vestigial in a pre-relation D if D contains a clause $(\_ =^x y)$, and we may say that the clause *has vestigial* x. A *child* of a variable x is a clause which has an input occurrence of x. A clause c has a *dependent* x if c has vestigial x or c has an output x. Note that every clause has exactly one dependent.

Definition 8.2.6 (Connectedness)
Two clauses c and c' are *connected* by a variable x in a pre-relation D if x is a dependent of c and has child c'.

We must now define a notion of path over a pre-relation, which will be central to our development. Intuitively, a path through a pre-relation is analogous to a path through the corresponding net. For the following definitions, we fix a particular pre-relation D.

Definition 8.2.7 (Path)
A path between two clauses c and c' in a pre-relation D is a pair of sequences $(c_1 \dots c_r, x_1, \dots x_{r-1})$ such that $c_1 = c$, $c_r = c'$ and x_i connects c_i and c_{i+1} for $i = 1 \dots r-1$.

Paths may be concatenated in the obvious way, so that a path from c to c' and one from c' to c'' yield a path from c to c''. A *complete* path from a clause c to another c' in a pre-relation D is a path such that c' has a dependent which occurs only as an output. Complete paths cannot be extended by post-concatenation.

Now we need to define linear and intuitionistic occurrences.

Definition 8.2.8 (Intuitionistic Occurrence)
A variable x occurs *intuitionistically* in a pre-relation D if

- D contains a clause $(\_ =^z x)$ or a clause $(M = y)$ where $x \in M$ or $y = x$,
- or D contains a clause:
  - $(y = F(x))$,
  - $(F(x) = y)$,
  - $(y = (\vec{y}_1; \vec{y}_2)O(\dots(\vec{x}; \vec{y}')D'(x')\dots;\dots))$ where $x \in \vec{x}$,
  - $(y = (\vec{y}_1; \vec{y}_2)O(\dots;\dots(\vec{x}; \vec{y}')(x')\dots))$ where $x \in \vec{x}$,
  - $(x = (;)O((\vec{x}_1; \vec{y}_1)v_1 \dots (\vec{x}_r; \vec{y}_r)v_r;))$ where $O \in \mathcal{O}_I$,

We now say that a variable occurs *linearly* in a pre-relation if it has an occurrence in a clause which is not intuitionistic.

Definition 8.2.9 (Free Variable)
A variable x is a *free input* of a pre-relation D if it occurs as an input in D but does not occur as an output or a binder. Similarly, a variable x is a *free output* of the pre-relation if it occurs as an output in D but does not occur as an input.

With this grounding, we can now define relations. Note that rather than being a construction-based definition like the definition of terms (a term is a pre-term that can be typed using these rules) it is a definition based on global properties.

Definition 8.2.10 (Relation)
Given a generalised typing context $\Gamma; \Delta$ and a type $A \in \mathsf{M}_L$, a $\Gamma; \Delta$-*relation* D of type A over $\mathbb{O}$ is a pre-relation over $\mathbb{O}$ having the following properties:

Linearity Each variable must occur in D at most once as an input, and at most once as an output or binder.

Colouring Each variable occurring twice must do so both times intuitionistically or both times linearly.

Typing Each variable occurring twice must do so both times with the same type annotation.

Boxing Each pre-relation D that occurs in a clause of D

$$(y = (\vec{x}'_1; \vec{x}'_2)O(\dots(\Gamma'; \Delta')D'(y' : A') \dots ; \dots))$$

must be a $\Gamma'; \Delta'$-relation of type A' with result variable y'.

Output There must be at most one variable x which occurs in D only as an output. If such a variable exists, it must occur linearly and with type A. If not, there must exist a unique variable x of type A in Δ but not occurring as an input, output or binder in D. We refer to the unique variable picked out by this requirement as the *result* variable of D for the typing context $\Gamma; \Delta$.

Inputs Γ must type at least the intuitionistically-occurring free inputs with the types they are annotated with in their occurrences. Also, if the result variable x is not a free output of D, then Δ must type exactly each free input of D with the type annotating it in D and $x : A$. If x is a free output of D, then Δ must type exactly each free input of D with the type annotating it in D.

Binding For any variable y which occurs as a binder in D in a clause c of the form

$$(y' = (\vec{x}'_1; \vec{x}'_2)O(\dots; \dots(\vec{y}'_1; \vec{y}'_2)(x'') \dots))$$

where $y \in \vec{y}'_1\vec{y}'_2$, either $y = x''$, or any complete path from a child of y must contain the clause c as its $i+1$th with the ith wire being x''.

Acyclicity No clause in D may have a path starting and ending at itself which is not the trivial path (c, ε).

Vestigial If D contains a clause of the form $(\_ =^y x')$, then y must be the result variable of D or must occur linearly as an input in D.

Although this definition seems circular, because of the references to relations occuring within the intuitionistic arguments of an operator clause, it is easy to decide whether any given pre-relation is a relation. This can be done by induction on the depth of operator clauses, since for any pre-relation there must be a bound on the depth of operator clauses contained in it.

Having defined relations over a generalised signature, we will let the set of all relations over the generalised signature $\mathbb{O}$ be denoted by $\mathsf{Rel}(\mathbb{O})$.

Basic Results

We now prove some useful lemmas about relations.

LEMMA 8.2.11 (PROPERTIES OF RELATIONS)

Children In a $\Gamma; \Delta$ relation D of type A, any variable has precisely one child except the result variable, which is childless.

Complete Paths I The last clause in any complete path from any clause c must have the result variable as a dependent.

Complete Paths II In a $\Gamma; \Delta$ relation D of type A, for any clause c there is at least one complete path from c to some other clause.

Unique Typing Given a generalised typing context $\Gamma; \Delta$ and a pre-relation D, there is at most one type A such that D is a $\Gamma; \Delta$-relation of type A.

Result Variable Given a $\Gamma; \Delta$-relation D of type A, the result variable can be determined effectively.

Proof

Children To prove this, note that the output property specifies that every variable except the result variable x must not be a free output, and hence must occur as an output and also as an input. Therefore any such variable must occur as the input to some clause which is then its child. However, such a variable must only occur once as an input, so there must be precisely one such clause. Since x is a free output, it cannot be the input of any clause, and hence it must be childless.

Complete Paths I This is obvious since the result variable is the only possible variable only occurring as an output.

Complete Paths II To show this, we need to demonstrate that every path which is not complete can be extended. Note first that every clause has at least one dependent. Now given a path from c to c' in a relation D, if c' has as a dependent the result variable we are done. If not, take a dependent of c' and its unique child; these extend the path.

Unique Typing Consider the two possibilities in the output condition of the definition of relations. If the result variable occurs as an output in D, then it must occur with a typing and for every typing context $\Gamma; \Delta$ it must have the same typing given as an annotation in D. If on the other hand the result variable does not occur in D but is typed in Δ, then it clearly follows that given the typing context there can only be one type assigned to the result variable.

Result Variable There are efficient ways to do this, but to see that it is true simply take the finite number of variables occurring linearly in D or typed in Δ, and for each test whether it occurs as an input in D, which is effective. Precisely one will not and this is the result variable.

We will now define a useful abbreviation. Abbreviate the pre-relation D containing the clauses $(M_1 = x_1), \ldots, (M_r = x_r)$ by removing these clauses and replacing them with the form $(M_1 \ldots M_r = x_1 \ldots x_r)$ which we will henceforth call a clause, overloading notation insignificantly. Further, we will in the same way abbreviate a pre-relation D containing the clauses $(\_ =^y x_1), \ldots, (\_ =^y x_r)$ by removing them and replacing them with the form $(\_ =^y x_1 \ldots x_r)$, which we will again call a clause.

Now define the *family* of a variable x occuring as a dependent in a relation D to be the set of clauses c such that every complete path in D from c contains the variable x. We will write this $\mathsf{fam}(x, D)$, omitting the relation D where it is obvious. Intuitively the family of a variable in a relation corresponds its maximal sub-relation which has x as its result variable.

We now prove two results on families.

LEMMA 8.2.12
Given a relation D, if a clause c is in two families $\mathsf{fam}(x, D)$ and $\mathsf{fam}(y, D)$, then either the variable x must have a child in $\mathsf{fam}(y, D)$, or y must have a child in $\mathsf{fam}(x, D)$.

Proof We know if a clause c is in two families $\mathsf{fam}(x, D)$ and $\mathsf{fam}(y, D)$, then we have that every complete path in D from c contains both x and y. Note that both x and y must occur as inputs and as dependents since otherwise they could not occur in a path. Now it must be the case that the unique children c_x and c_y respectively of x and y must both occur in every complete path from c in D. Now assume that in some complete paths from c in D the clause c_x occurs before c_y and in others c_y occurs before c_x. Then, by taking appropriate sub-paths of these paths we can construct a cyclic path from c_x to itself in D, which is a contradiction. Therefore, one occurs first in every complete path from c in D. Say w.l.o.g. that it is c_x. Then every complete path from c_x in D must be part of a complete path from c in D, and further it occurs in that path before c_y, so that y is in any complete path from c_x in D. Therefore, c_x is in $\mathsf{fam}(y, D)$. □

LEMMA 8.2.13
Given a $\Gamma; \Delta$-relation D and a variable x which occurs linearly in D, $\mathsf{fam}(x, D)$ is a $\Gamma'; \Delta'$-relation for some $\Gamma'; \Delta'$.

Proof We can prove this by considering each clause in turn. Clearly linearity, colouring, typing, boxing and acyclicity are inherited from D.

Of those that are left, first consider the output condition. Note that either x does not occur as an output in $\mathsf{fam}(x, D)$ because it is the empty relation, or because it occurs only as an output. This is trivial since if it were to occur as an input it would imply that the child of x, c_x, had a complete path in D which contained x, which would induce a cycle in D. Further, if any clause c is contained in $\mathsf{fam}(x, D)$ then a clause having x as dependent must, since each complete path in D from c must contain x as a connection between a dependent of x and its child. Now, having established that x is an appropriate candidate for the result variable, we must show that there can never be another variable y occuring only as an output in $\mathsf{fam}(x, D)$. For a contradiction, assume there is such a y. Then

y must occur as the output of a clause c' in $\mathsf{fam}(x, D)$. Now if y has a child c_y in D then there must be a complete path from c' in D containing y, which is the completion of the path (cc_y, y). But there is some complete path from c_y in D not containing x, or c_y would be in $\mathsf{fam}(x, D)$ and y would occur also as an input. Hence we can construct a complete path from c' in D which does not contain x. So y must be childless in D, and hence the result variable of D. But then the empty path from c' by definition must contain x, which is a contradiction. Hence the result variable of D cannot occur only as an output in $\mathsf{fam}(x, D)$, and x is the only such possible variable.

Considering the input condition, we can take Γ' to be the typing sequence that types all the intuitionistically occuring variables with the types they are annotated with, and Δ' to be the typing sequence which types all the linearly occuring ones with the types they are annotated with, and types the result variable x with any type A if it does not occur in $\mathsf{fam}(x, D)$. This will make $\mathsf{fam}(x, D)$ a $\Gamma'; \Delta'$-relation of some type.

Considering the binding condition, we can see that if an operator clause c with a binder y occurs in $\mathsf{fam}(x, D)$, then any complete path in D from a child of y must contain the clause c, and must therefore contain a complete path from c in D as a final segment. But this final segment must contain x, and so every complete path from the child of the binder must contain x and the child of the binder must be in $\mathsf{fam}(x, D)$, and the condition must be satisfied.

Considering the vestigial condition, assume we have a clause $(\_ =^y y')$ such that y is not the result variable of $\mathsf{fam}(x, D)$ and does not occur as an input in it. But if this clause occurs in $\mathsf{fam}(x, D)$, then it must be the case that there is a complete path from the clause in D, and such a path must contain x, implying that the child of y is in $\mathsf{fam}(x, D)$, which is a contradiction. $\square$

Now define a $\Gamma'; \Delta'$ sub-relation E of a $\Gamma; \Delta$-relation D to be a subset of D which is a $\Gamma'; \Delta'$ relation. A fundamental property of subrelations which we will need is the following:

Lemma 8.2.14

Given a relation R with a subrelation $R' \subseteq R$, any complete path in R from a clause $c \in R'$ must pass through a clause c' having as dependent the result variable of R'.

Proof Assume not. Now take the maximal non-zero initial segment of the path which is within R'. The last clause in this segment must have a dependent which is not the input of a clause in R'. If the dependent is a vestigial, the vestigial condition on R' implies that the dependent is the result variable of R', which is a contradiction. If it is not a vestigial, then it is a variable which does not occur as an input in R' and does occur as an output (of the last clause in the initial segment) and hence it must be the output variable of R'. $\square$

8.3 Relational Equality

We now define the various parts of the equality over relations. Since we are interested in proving that term equality is decidable via a translation into the full relational equality, we will need to make sure that each part of the equality is decidable. This will often be achieved by making a part of the equality the transitive reflexive closure of a confluent terminating rewrite.

Vestigial Moves

First, we introduce the concept of *vestigial moves* on a relation. As we shall see, the vestigial x in a clause of the form $(\_ =^x y)$ gives us essential information about scoping in the derivations corresponding to the relation. If we do not have this information, then two relations representing derivations not of provable equal terms may be the same. However, there is redundancy in that several relations with different vestigials may correspond to several provably equal terms. Hence we need to give an equality between such relations. We call this equality $=_{vm}$, and we make a definition:

Definition 8.3.1 (Vestigial Moves)
The $\Gamma;\Delta$-relations $D \cup \{(\_ =^x y)\}$ and $D \cup \{(\_ =^{x'} y)\}$ of type A are one-step equal *via vestigial moves*, if

1. There exists a path in $D \cup \{(\_ =^x y)\}$ from the child of x to a clause having dependent x'. Note that such a path cannot contain the clause $(\_ =^x y)$ or a cycle would result.

2. For any clause c of the form

$$(y' = (\vec{x}'_1; \vec{x}'_2)O(\ldots; \ldots (\vec{y}'_1; \vec{y}'_2)(x'') \ldots))$$

with $z \in \vec{y}_1\vec{y}_2$ such that there is a path from the child of z to $(\_ =^x y)$ in $D \cup \{(\_ =^x y)\}$, all complete paths from the dependent of x' must contain the clause c as their $i+1$th and the variable x'' as their ith.

We now say that $D =_{vm} D'$ if these two relations are shown to be equal by a finite sequence of one-step vestigial moves.

Having given this definition, it is imperative that we show that it is decidable.

LEMMA 8.3.2 (DECIDABILITY)
Vestigial equality is decidable.

Proof We can prove this by considering the following effective procedure:

> Given any finite set of relations, pick any relation, pick any vestigial and make any valid vestigial move. If the result is not in the set, add it and continue until no relation in the set has a vestigial move resulting in a new relation.

First we show that this is effective; note that determining whether a vestigial move is valid is a matter of considering a finite number of paths in a relation, and hence is effective. Since we start out with a finite set of relations, each of which have a finite number of vestigials, it is effectively possible to check whether any relation has a vestigial move on a particular vestigial.

Now, given this effective procedure, start it with the singleton set containing any relation. Since there are only a finite number of variables which may be used as each vestigial (by the vestigial condition, since the variable must occur elsewhere in the body of the relation not as a vestigial), there are at most a finite number of relations which may be equal by a vestigial move to any given relation. Since there must be a finite sequence of one-step vestigial moves relating any two relations which are equal via vestigial moves (since an infinite sequence would of necessity repeat in the finite set of vestigially-equal relations), this procedure must result in the finite set of all relations vestigially equal to the relation we initially considered. Since the set is finite, we can now simply check to see if the relation is vestigially equal to another relation by checking to see if it is in the finite set resulting from the procedure. $\square$

The Δ-rewrite

We now recall that our system includes clauses corresponding not only to the rules of our logic, but also to its admissible weakening and contraction rules. This is to ensure that we can express the equality of the system using local rewrites. Consider $\mathsf{DILL}(\mathbb{C})$. Recall from page 41 that in the presence of admissible cut rules for equality, the equality of $\mathsf{DILL}(\mathbb{C})$ is given by axioms which do not involve substitutions (the substitutions which appear in the η-rules can be equivalently rewritten as side-conditions on the equalities). However, there is still an infinite set of these axiomatic equalities parameterised over terms, for example as in the equality $\Gamma;\Delta \vdash \mathsf{let} * \mathsf{be}\ x\ \mathsf{in}\ t\{*/x\} = t\!:\!A$. It is possible in the presence of the commuting conversions to derive these equalities from a finite set of axioms not parameterised over terms. Our rewrites on the system of relations for $\mathsf{DILL}(\mathbb{C})$ will be suitable orientations of these.

However, this still leaves us with the problem of giving rewrites verifying the the admissible cut rules for equality. To do this in a local way for intuitionistic substitution we need to give a low-level implementation for the copying and discarding of intuitionistic relations, which is provided by the rewrite $\rightarrow_\Delta$.

Definition 8.3.3 (The Rewrite $\rightarrow_\Delta$)
We define the rewrite $\rightarrow_\Delta$ firstly over pre-relations as the transitive closure of the following one-step rewrites, where M and M' may be any non-empty sets:

$$D \cup \{(\_ =^x y'), (M \cup \{z', y'\} = z)\} \rightarrow_\Delta D \cup \{(M \cup \{z'\} = z)\}$$
$$D \cup \{(\{x\} = y)\} \rightarrow_\Delta D\{y/x\}$$
$$D \cup \{(M' = y), (M \cup \{z', y\} = x)\} \rightarrow_\Delta D \cup \{(M \cup \{z'\} \cup M' = x)\}$$

We call the clauses which are explicitly mentioned in the redex the *key* clauses of the rewrite.

Now it is easy to show that if $D \rightarrow_\Delta D'$, where D is a $\Gamma;\Delta$-relation of type A, then D' is a $\Gamma;\Delta$-relation of type A. We note here that $\rightarrow_\Delta$ is clearly terminating since in each one-step rewrite the number of clauses is reduced. Given this result, we now also need to prove the confluence of $\rightarrow_\Delta$ over $=_{vm}$.

Lemma 8.3.4 (Confluence)
The rewrite $\rightarrow_\Delta$ is confluent over the equality $=_{vm}$ on $\mathsf{Rel}(\mathbb{O})$.

Proof First we note that if $D_1 \rightarrow_\Delta D_2$ by a one-step Δ-rewrite and $D_1' =_{vm} D_1$, then $D_1' \rightarrow_\Delta D_2'$ by a one-step Δ-rewrite of the same kind, and $D_2' =_{vm} D_2$. We can see this in the case where the vestigial moves are one-step by noticing that since no linear variables are involved in any Δ-rewrite (as all the variables in the key clauses are occurring intuitionistically), any such rewrite is independent of the vestigial move. It is also trivial to see that the effect of the Δ-rewrite on paths cannot disrupt the vestigial move.

Given this, in order to prove the result it suffices to show that if one relation D has two Δ-rewrites to D_1 and D_2 respectively, that there exist Δ-rewrites $D_1 \rightarrow_\Delta D_1'$ and $D_2 \rightarrow_\Delta D_2'$ such that D_1' and D_2' are equal via vestigial moves. This is trivial by inspection except in the case where D is

$$D' \cup \{(\_ =^x y_1), (\_ =^{x'} y_2), (\{y_1, y_2\} = y)\}$$

In this case D_1 is $D' \cup \{(\_ =^{x} y)\}$ and D_2 is $D' \cup \{(\_ =^{x'} y)\}$. Hence we need to show that these two are equal via vestigial moves. Consider an arbitrary binding clause in $D' \cup \{(\_ =^{x} y_1), (\_ =^{x'} y_2), (\{y_1, y_2\} = y)\}$ having a binding variable z with a path from the child of z to the clause $(\_ =^{x'} y_2)$. Clearly there must also exist a path from the child of z to the clause $(\_ =^{x} y_2)$. Hence any binding clause must contain both discard clauses in its scope or either. Now, by virtue of this any complete path from either of the vestigial clauses must contain all the binding clauses that bind them both, and furthermore in each path they must occur in the same order (or else we could construct a path which omitted one). Take the binding clause which occurs first in an arbitrary complete path from either clause, and call it c. We know further that there is an input to c, say x'', which equally must be in every complete path. Now we claim that D_1 and D_2 are both equal by a one-step vestigial move to $D' \cup \{(\_ =^{x''} y)\}$. To show this note that the second part of the requirement is satisfied by virtue of our earlier discussion. The first part, that there exists a path from a child of x or x' respectively to a clause having x'' as dependent in both D_1 and D_2, can be shown since in each case, any complete path from the vestigial clause, including the one passing through the child of x or x', must contain c and x''. $\square$

Having defined these two equalities, we will refer to the system of relations quotiented by the equality generated by the transitive closure of the rewrite $\to_{\Delta}$ over the equality $=_{vm}$ as $\mathsf{Rel}(\mathbb{O}, =_{vm,\Delta})$.

8.4 Terms to Relations

We now map $\mathsf{Lin}(\mathbb{O})$ to $\mathsf{Rel}(\mathbb{O}, =_{vm,\Delta})$ in such a way that when we quotient $\mathsf{Lin}(\mathbb{O})$ by the subset of the provable equality judgements constructed using only the cc-axioms, it will be isomorphic to $\mathsf{Rel}(\mathbb{O}, =_{vm,\Delta})$.

Definition 8.4.1 (The Translation ρ)
The translation ρ takes $\Gamma;\Delta$-terms of type A in $\mathsf{Lin}(\mathbb{O})$ to $\Gamma;\Delta$-relations of type A in $\mathsf{Rel}(\mathbb{O})$ as follows, where $\Gamma = y_1{:}R_1 \dots y_{r''}{:}R_{r''}$:

$$\begin{aligned}
\rho_\Gamma(x) &= \{(_ =^x \vec{y})\} \qquad\qquad \text{(where } x \text{ is not typed in } \Gamma)\\
\rho_{\Gamma,x:Q}(x) &= \{(x' = F(x)), (_ =^{x'} \vec{y})\}\\
\rho_\Gamma(\mathsf{let}\ x{:}Q\ \mathsf{be}\ v\ \mathsf{in}\ w) &= \{(F(x) = z_v), (\{y'_1, y''_1\} \dots \{y'_{r''}, y''_{r''}\} = \vec{y})\}\\
&\quad \cup \rho_{\vec{y}:\vec{R}}(v)\{\vec{y}'/\vec{y}\} \cup \rho_{\vec{y}:\vec{R},x:Q}(w)\{\vec{y}''/\vec{y}\}
\end{aligned}$$

$$\begin{aligned}
(\text{On } O \in \mathcal{O}_I)\ \rho_\Gamma(op) &= \{(M_1 \dots M_{r''} = \vec{y}), (x' = F(x''))\} \cup\\
&\quad \{(x'' = (;)O((\vec{y}^{v_1}\vec{x}_1; \vec{y}_1)D_1, \dots, (\vec{y}^{v_r}\vec{x}_r; \vec{y}_r)D_r)\}\\
(\text{On } O \in \mathcal{O}_L - \mathcal{O}_I)\ \rho_\Gamma(op') &= (\bigcup_{i=1\dots s} E_i) \cup (\bigcup_{j=1\dots r'} D'_j) \cup (\bigcup_{i'=1\dots s'} E'_{i'}) \cup \{(M'_1, \dots M'_{r''} = \vec{y})\} \cup
\end{aligned}$$

$$\{(x'' = (\vec{z}'; \vec{z}'')O((\vec{y}^{v_1}\vec{x}_1; \vec{y}_1)D_1, \dots, (\vec{y}^{v_r}\vec{x}_r; \vec{y}_r)D_r; (\vec{x}'_1; \vec{y}'_1)(z_{w_1}), \dots, (\vec{x}'_s; \vec{y}'_s)(z_{w_s})))\}$$

where x', x'' and all the $\vec{y}'$, $\vec{y}''$ and $\vec{y}^v$ for terms v are fresh variables and:

$$\begin{aligned}
E_i &= \rho_{\Gamma,\vec{x}'_i:\vec{Q}'_i}(w_i)\{\vec{y}^{w_i}/\vec{y}\}\\
D'_j &= \rho_\Gamma(v'_j)\{\vec{y}^{v'_j}/\vec{y}\}\\
E'_{i'} &= \rho_\Gamma(w'_{i'})\{\vec{y}^{w'_{i'}}/\vec{y}\}\\
D_{j'} &= \rho_{\Gamma,\vec{x}_{j'}:\vec{Q}_{j'}}(v_{j'})\\
\vec{z}' &= z_{v'_1} \dots z_{v'_{r'}}\\
\vec{z}'' &= z_{w'_1} \dots z_{w'_{s'}}\\
M_i &= \{y_1^{v_1}, \dots, y_1^{v_r}\}\\
M'_i &= \{y_1^{v_1}, \dots, y_1^{v_r}, y_1^{w_1}, \dots, y_1^{w_s}, y_1^{v'_1}, \dots, y_1^{v'_{r'}}, y_1^{w'_1}, \dots, y_1^{w'_{s'}}\}
\end{aligned}$$

$$\begin{aligned}
op &= O((\vec{x}_1{:}\vec{Q}_1; \vec{y}_1{:}\vec{A}_1)v_1, \dots, (\vec{x}_r{:}\vec{Q}_r; \vec{y}_r{:}\vec{A}_r)v_r;)()\\
op' &= O((\vec{x}_1{:}\vec{Q}_1; \vec{y}_1{:}\vec{A}_1)v_1, \dots, (\vec{x}_r{:}\vec{Q}_r; \vec{y}_r{:}\vec{A}_r)v_r;\\
&\qquad (\vec{x}'_1{:}\vec{Q}'; \vec{y}'_1{:}\vec{A}')w_1, \dots, (\vec{x}'_s{:}\vec{Q}'; \vec{y}'_s{:}\vec{A}')w_s)(v'_1, \dots, v'_{r'}; w'_1, \dots, w'_{s'})
\end{aligned}$$

and where we write z_v for the result variable of $\rho_\Gamma(v)$ and similarly for the terms w, $\vec{v}$, $\vec{w}$, $\vec{v}'$ and $\vec{w}'$.

We need to show that definition is well-formed, ie that ρ applied to a term is always a relation. We can do this by induction over derivations of terms. Consider the conditions on relations in turn.

Linearity To show that each variable only occurs at most once as an input and at most once as an output or binder, first consider the axiom clauses. The result obviously follows in this case since x and $\vec{y}$ must be distinct, and x' is fresh. Now consider the inductive clauses. In each of these, when we refer inductively to another instance of the translation $\rho_\Gamma(v)$, we only introduce another occurrence of free variables in the translation instance, and in fact we only introduce new occurrences of variables which are free inputs of $\rho_\Gamma(v)$ as outputs and vice versa. Hence this property is satisfied.

Colouring We can show in a very similar way that the translation preserves colouring.

Typing Again, we need only to show firstly that the base clauses of the definition satisfy the typing property, which is obvious since in each case there is only ever one reference to each variable, which has an obvious type, and secondly by showing that in the inductive clauses we only ever refer twice to variables which are free in an instance $\rho_\Gamma(v)$ of the translation, and in this case since all the free variables in the instance of the translation are typed in the typing context $\Gamma; \Delta$ of the term v, we can see that they are assigned the correct types in their other occurrence.

Boxing This clearly follows by induction, since we only ever use instances of the translation in boxed clauses.

Output Consider the base cases. In these, the unique result variable is clear; it is x in the first and x' in the second. We can then show that the result variables of the two inductive cases are z_w and x'' respectively.

Inputs This condition is easy to establish in the base cases, and in the inductive cases it can be shown by observing that we bind precisely the correct variables of the operator arguments using the binding occurrences of the operator clause.

Binding In order to show this, we consider the only situation in which we introduce a binder, which is in the operator clause of the definition. But we can easily see that we only introduce binders which are bound variables in the ith argument of an operator term. Now we know by induction that in the image of a term under ρ, which is a relation, there must exist a complete path from any child of a free input. Also, because we know that no variable in the image of the argument apart from its free variables is referred to elsewhere in the translation of the operator instance, the only paths from a child of a binder must contain as initial sequences complete paths of the translation of the argument. But we can easily show that every complete path of the translation can only be extended in the translation of the operator term by adding the ith output variable and the operator clause itself, and further that each complete path through the translation of the argument can be extended in this way in the translation of the operator term.

Acyclicity This is again seen by induction; in the base cases is is clear that there is no cyclic path, in the let case if there is no cycle in the inductive instances of the translation then we can show by inspecting the form of the definition that there is no cycle possible in the image, and finally in the operator clause we can do the same thing.

Vestigial This is easy to see by considering the base case, which is the only point at which we add a vestigial. Clearly in that case it is the result variable of the image of the translation. □

Now we can show that the fragment of the provable equality defined by the c-axioms and the rules of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ is mapped soundly by this translation into relational equality up to $=_\Delta$.

LEMMA 8.4.2 (SOUNDNESS)
If two terms are provably equal $\Gamma; \Delta \vdash v = w : A$ in the fragment of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$ using just the cc axioms and the symmetry, reflexivity, transitivity and congruence rules, then their images $\rho_\Gamma(v)$ and $\rho_\Gamma(w)$ are equal up to $=_{vm,\Delta}$ on relations.

Proof We prove this by considering the structure of a derivation of an equality judgement in this fragment of the theory. Since the rewrite is congruent, and we are taking the transitive reflexive closure of it, we have that the equality $=_\Delta$ is a congruence. Hence we need only check that the cc equalities are soundly mapped to relations. We consider some sample cases.

$cc - 1$) In this case, the image of the left hand side easily reduces using the free variable condition to the relation

$$\rho_{\Gamma,y:Q}(w')\{\vec{y}'/\vec{x}'\} \cup \rho_{\Gamma,x:R}(v)\{\vec{y}''/\vec{x}'\} \cup \rho_\Gamma(w)\{\vec{y}'''/\vec{x}'\} \\ \cup \{(\{y_1', y_1'''\}, \ldots \{y_r', y_r'''\} = \vec{x}''), (\{y_1'', x_1''\}, \ldots \{y_r'', x_r''\} = \vec{x}')\}$$

where $\Gamma = x_1' {:} Q_1' \ldots x_r' {:} Q_r'$. But this reduces to

$$\rho_{\Gamma,y:Q}(w')\{\vec{y}'/\vec{x}'\} \cup \rho_{\Gamma,x:R}(v)\{\vec{y}''/\vec{x}'\} \cup \rho_\Gamma(w)\{\vec{y}'''/\vec{x}'\} \\ \cup \{(\{y_1'', y_1', y_1'''\}, \ldots \{y_r'', y_r', y_r'''\} = \vec{x}')\}$$

and the image of the right-hand side easily reduces using the free variable condition to the relation

$$\rho_{\Gamma,y Q}(w')\{\vec{y}'/\vec{x}'\} \cup \rho_{\Gamma,xR}(v)\{\vec{y}''/\vec{x}'\} \cup \rho_\Gamma(w)\{\vec{y}'''/\vec{x}'\}$$
$$\cup \{(\{y_1'', y_1'''\}, \ldots \{y_r'', y_r'''\} = \vec{x}''), (\{y_1', x_1''\}, \ldots \{y_r', x_r''\} = \vec{x}')\}$$

which has the same reduct as the left hand side.

$cc-2$) This is almost identical to the previous case.

$cc-3$) **and** $cc-4$) These cases follow the same principle as that of $cc-1$ since we have a union of the sets $\rho_\Gamma(v)$ and $\rho_\Gamma(v_i')$.

$cc-5$) This is the crucial case, since it shows that our relational equality handles linear naturality correctly. In order to make the process clearer, we only prove this in the case we there are no intuitionistically natural arguments (since these are peripheral to the equality) and precisely two linearly natural arguments. The more general case is proved using exactly the same techniques. In the restricted case, the equality is as follows:

$$\Gamma;\Delta \vdash \mathsf{let}\ x'\ \mathsf{be}\ v\ \mathsf{in}\ O(;(\vec{x}_1;\vec{y}_1)w_1,(\vec{x}_2;\vec{y}_2)w_2)(\vec{v}';\vec{w}')$$
$$= O(;(\vec{x}_1;\vec{y}_1)w_1,(\vec{x}_2;\vec{y}_2)(\mathsf{let}\ x'\ \mathsf{be}\ v\ \mathsf{in}\ w_2))(\vec{v}';\vec{w}')$$

The image of this easily reduces under the free variable condition to the relation

$$(\cup_{i=1\ldots r'}\rho_\Gamma(v_i')\{\vec{y}^{v_i'}/\vec{y}\}) \cup (\cup_{j=1\ldots s'}\rho_\Gamma(w_j')\{\vec{y}^{w_j'}/\vec{y}\}) \cup \rho_{\Gamma,\vec{x}_1:\vec{Q}_1}(w_1)\{\vec{y}^{w_1}/\vec{y}\}$$
$$\cup\, \rho_{\Gamma,x':Q',\vec{x}_2:\vec{Q}_2}(w_2)\{\vec{y}^{w_2}/\vec{y}\} \cup \rho_\Gamma(v)\{\vec{y}^{v}/\vec{y}\} \cup \{(\{y_1', y_1^v\}, \ldots, \{y_r', y_r^v\} = \vec{y}),$$
$$(\{y_1^{v_1'}, \ldots, y_1^{v_{r'}'}, y_1^{w_1'}, \ldots, y_1^{w_{s'}'}, y_1^{w_1}, y_1^{w_2}\}, \ldots, \{y_r^{v_1'}, \ldots, y_r^{v_{r'}'}, y_r^{w_1'}, \ldots, y_r^{w_{s'}'}, y_r^{w_1}, y_r^{w_2}\} = \vec{y}')\}$$

where $\Gamma = y_1{:}R_1 \ldots y_r{:}R_r$. But this has reduct

$$(\cup_{i=1\ldots r'}\rho_\Gamma(v_i')\{\vec{y}^{v_i'}/\vec{y}\}) \cup (\cup_{j=1\ldots s'}\rho_\Gamma(w_j')\{\vec{y}^{w_j'}/\vec{y}\}) \cup \rho_{\Gamma,\vec{x}_1:\vec{Q}_1}(w_1)\{\vec{y}^{w_1}/\vec{y}\}$$
$$\cup\, \rho_{\Gamma,x':Q',\vec{x}_2:\vec{Q}_2}(w_2)\{\vec{y}^{w_2}/\vec{y}\} \cup \rho_\Gamma(v)\{\vec{y}^{v}/\vec{y}\} \cup$$
$$\{(\{y_1^{v_1'}, \ldots, y_1^{v_{r'}'}, y_1^{w_1'}, \ldots, y_1^{w_{s'}'}, y_1^{w_1}, y_1^{w_2}, y_1^{v}\}, \ldots,$$
$$\{y_r^{v_1'}, \ldots, y_r^{v_{r'}'}, y_r^{w_1'}, \ldots, y_r^{w_{s'}'}, y_r^{w_1}, y_r^{w_2}, y_r^{v}\} = \vec{y})\}$$

But in a similar way the right-hand side has precisely this reduct. $\square$

8.5 Relations to Terms

Having given the sound translation from terms to relations, we now need to give a translation in the reverse direction. In fact, this is a complex procedure, largely because relations are not inherently a sequential syntax, whereas terms are. Hence the problem of inverting ρ is a version of what is historically known as the sequentialisation problem, first stated for proof-nets. As we already know, there is no one canonical proof derivation associated with a particular relation, although each relation does have an associated equivalence class of derivations (under the fragment of equality we are considering).

It is not clear how to give a direct translation from relations to terms, as we have no direct inductive characterisation of relations, and although we do for pre-relations, not all pre-relations can be mapped to terms in a uniform way. Hence, the way we will define the inverse translation is non-standard. First we define a class of *intermediates*, which are as the name suggests half-way houses between relations and terms, and then we define a rewrite on these which is confluent and terminating up to an equality on these intermediates. Such a rewrite defines a function on intermediates, and we use it in conjunction with a map from relations to intermediates and a map from intermediates to terms to give a function from relations to terms.

First we define intermediates.

Definition 8.5.1 (Intermediates)
A $\Gamma'; \Delta'$-*intermediate of type A over a signature* $\mathbb{O}$ is a triple $(v, D, (\Gamma; \Delta))$ such that v is a $\Gamma; \Delta$-term of type A, D is a finite set of clauses, (the clause set), $\rho_\Gamma(v) \cup D$ is a $\Gamma'\Delta'$ relation of type A, all the variables occuring only as outputs in D are typed in $\Gamma; \Delta$ and there are no clauses in D having dependents variables occuring as outputs in $\rho_\gamma(v)$.

We will write the set of intermediates over a signature $\mathbb{O}$ as $\mathsf{Int}(\mathbb{O})$, ranged over by $I \ldots$, and identify intermediates up to permutation on the linear and intuitionistic parts of the typing context.

Now we define a rewrite on intermediates, $\to_S$:

Definition 8.5.2 (The Sequentialisation Rewrite)
We define the one-step *sequentialisation rewrite* $\to_S$ on intermediates as follows:

$$\begin{array}{lrl}
1) & (v, D \cup \{(_ =^x y{:}Q)\}, (\Gamma; \Delta, x{:}A)) \to_S & (v, D, (\Gamma, y{:}Q; \Delta, x{:}A)) \\
2) & (v, D \cup \{(M = x{:}Q)\}, (\Gamma, M{:}Q; \Delta)) \to_S & (v\{x/M\}, D, (\Gamma, x{:}Q; \Delta)) \\
3) & (v, D \cup \{(x = F(y){:}Q)\}, (\Gamma; \Delta, x{:}Q)) \to_S & (v\{y/x\}, D, (\Gamma, y{:}Q; \Delta)) \\
4) & (v, D \cup \{(F(x) = y{:}Q)\}, (\Gamma, x{:}Q; \Delta)) \to_S & (\mathsf{let}\ x\ \mathsf{be}\ y\ \mathsf{in}\ v, D, (\Gamma; \Delta, y{:}Q)) \\
5a) & (v, D \cup \{op\}, (\Gamma, x{:}Q; \Delta)) \to_S & (v\{w/x\}, D, (\Gamma, \Gamma''; \Delta) \\
5b) & (v, D \cup D_1' \cup \ldots \cup D_s' \cup \{op'\}, (\Gamma; \Delta, y{:}B'')) \to_S & (v\{w'/y\}, D, (\Gamma, \Gamma'; \Delta, \Delta'))
\end{array}$$

where this last rewrite holds if D has no clauses $(\_ =^{x''} y'')$ for x'' occurring as an output in $D_1' \ldots D_s'$, for all $i = 1 \ldots r$ and all $j = 1 \ldots s$, we have:

$$(z_{v_i}, D_i, (\_; z_{v_i}{:}B_i)) \to_S (v_i, \emptyset, (\Gamma^{v_i}, \vec{x}_i{:}\vec{Q}_i; \vec{y}_i{:}\vec{A}_i))$$
$$(z_{w_j}, D_j', (\_; z_{w_j}{:}B_j')) \to_S (w_j, \emptyset, (\Gamma^{w_j}, \vec{x}_j'{:}\vec{Q}_j'; \Delta^{w_j}, \vec{y}_j'{:}\vec{A}_j'))$$

and given $\Gamma^{v_i} = \vec{x}^{v_i}{:}\vec{A}^{v_i}$ for $i = 1 \ldots r$, we abbreviate:

$$\begin{aligned}
op &= (y = (;)O((\vec{x}^{v_1}\vec{x}_1; \vec{y}_1)D_1, \ldots, (\vec{x}^{v_r}\vec{x}_r; \vec{y}_r)D_r;)) \\
op' &= (y = (\vec{x}''; \vec{y}'')O((\vec{x}^{v_1}\vec{x}_1; \vec{y}_1)D_1, \ldots, (\vec{x}^{v_r}\vec{x}_r; \vec{y}_r)D_r; (\vec{x}_1'; \vec{y}_1')z_{w_1}, \ldots, (\vec{x}_s'; \vec{y}_s')z_{w_s})) \\
w &= O((\vec{x}_1; \vec{y}_1)v_1, \ldots (\vec{x}_r; \vec{y}_r)v_r;)(;) \\
w' &= O((\vec{x}_1; \vec{y}_1)v_1, \ldots (\vec{x}_r; \vec{y}_r)v_r; (\vec{x}_1'; \vec{y}_1')w_1, \ldots (\vec{x}_s'; \vec{y}_s')w_s)(\vec{x}''; \vec{y}'') \\
\Gamma'' &= \Gamma^{v_1}, \ldots, \Gamma^{v_r} \\
\Gamma' &= \Gamma^{v_1}, \ldots, \Gamma^{v_r}, \Gamma^{w_1} \ldots, \Gamma^{w_s} \\
\Delta' &= \Delta, \vec{x}''{:}\vec{Q}'', \vec{y}''{:}\vec{A}'', \Delta^{w_1}, \ldots, \Delta^{w_s}
\end{aligned}$$

We specify further that a one-step rewrite of type 3 or $5b$ removing a variable x from the linear typing context here can only occur when there is no clause $(\_ =^x y)$ for some y which could be rewritten by a rewrite of type 1.

We need to check that this definition is well-formed, ie that the reduct of an intermediate is still an intermediate. This can be seen using simple properties of relations and of the translation ρ.

Intuitively, a sequentialisation rewrite on $\Gamma''; \Delta''$-A-intermediates

$$(v, D, (\Gamma; \Delta)) \to_S (v', D', (\Gamma'; \Delta'))$$

moves structure from the relation part of the intermediate to the term part in a sound way. By this we mean that the $\Gamma''; \Delta''$-A-relations $\rho_\Gamma(v) \cup D$ and $\rho_{\Gamma'}(v') \cup D'$ will be equal up to $=_{\Delta,vm}$.

We can now start to prove the results which we need about this rewrite. First note that each rewrite reduces the clause set by at least one clause. In the case of rewrites $1-4$ this is the obvious one, in $5a$ it is op, and in $5b$ it is op'. We call this clause the *active* clause in a rewrite. Further, note that any rewrite may only occur if all the output variables of its active clause are present in the typing context. This is not a sufficient condition because of the 1st rewrite and the restriction on rewrite order. We will refer to the rewrites which are given as conditions for the $5b$th rewrite as *subsidiary* rewrites of that main rewrite. Similarly, we will call the intermediates which are rewritten by the subsidiary rewrites the subsidiary intermediates.

Now since $\rho_\Gamma(v) \cup D$ must be a $\Gamma';\Delta'$-relation of type A, we know that no variable in D can occur more than once as an output, and hence we know that it may only be the output variable of one clause. This implies that any two instances of rewrites with a shared output variable must have the same active clause. In the cases of rewrites $2-4$ this implies that they are the same. We also note that variables are only added to the typing context when they are the inputs of active clauses in rewrites. This is clear except in the case of rewrite $5b$, but can be seen by induction even in that case.

We can also easily see that the only way of removing a clause from the clause set is via a rewrite of which it is an active clause, or equivalently via a rewrite which removes its output variables from the typing context. We will say that an intermediate rewrites totally if it rewrites to an intermediate with an empty clause set.

We firstly now show that every intermediate rewrites totally. This will require two lemmas.

Lemma 8.5.3
Under the assumption:

> For all D containing fewer than n clauses, if $(v, D, (\vec{x}:\vec{Q};\vec{y}:\vec{A}))$ has a clause c with all its dependents in $\vec{x}\vec{y}$, then it has a sequence of one-step rewrites whose last rewrite has active clause c.

we can show that any intermediate with a clause set having less than n elements rewrites totally.

Proof To show this, we prove that any intermediate with a non-empty clause set has at least one rewrite. Then note that given any intermediate since the clause set must be finite, the result follows. Firstly, take any intermediate with clause set having less than n elements. If it has no elements then we are done. If it has at least one then take one arbitrarily. This clause then must have at least one dependent, and these dependents are either all free output variables, in which case they are all in $\vec{x}\vec{y}$ and by the assumption there exists a rewrite, or there is one dependent of the clause which is the input of another clause c'. Using this clause we can then either find a rewrite or a third clause, and this procedure if continued will give us a rewrite with active clause the ith clause we select, or it will give us an infinite sequence of clauses such that each has as input a dependent of the previous one. However, since the clause set must be finite, such a sequence of clauses must contain a cycle, and it is easy to see that from the clauses and the dependents we can therefore construct a cyclic path in the clause set, which is a contradiction. $\square$

Having proved this conditional result, we now go on to prove the main lemma, which is also the assumption of the conditional in the general case:

Lemma 8.5.4
If $(v, D, (\vec{x}:\vec{Q}; \vec{y}:\vec{A}))$ has a clause c with all its dependents in $\vec{x}\vec{y}$, then it has a sequence of one-step rewrites whose last rewrite has active clause c.

Proof We will prove this by induction over the size of the clause set. If the clause set D is empty the lemma is vacuous. If the clause set has $n+1$ elements, then assume it has a clause c satisfying the premise of the lemma. There are now three possible cases. Firstly we may have that this clause has the form $(\_ =^x y)$, in which case the assumption implies that the rewrite (of type 1) can proceed. Secondly we may have that this clause is not an operator clause and is not a clause $(\_ =^x y)$, in which case we know that the assumption must allow the rewrite unless there is a possible rewrite of type 1 which may take precedence. Note however that if there are such rewrites, there can only be finitely many of them, and note also that they merely increase the size of the typing context, and therefore do not block the rewrite removing the clause c. Hence we can perform the sequence of rewrites of type 1 followed by the rewrite having active clause c. The third and most complex case is when the clause is an operator clause. In this case the assumption is that the sole output variable of the operator is in $\vec{x}\vec{y}$, but we also need to know that there exists suitable subsidiary intermediates and rewrites on them. However, since such intermediate must have smaller clause sets than D, we know that any subsidiary intermediate of the appropriate form must rewrite totally. Hence it suffices to show that suitable subsidiary intermediates exist. However, since we know that the family of any variable is a relation, we can simply take as subsidiary intermediates $(z_{w_j}, \mathsf{fam}(z_{w_j}, D \cup \rho_{\Gamma'}(v)), (\_; z_{w_j}))$. These are disjoint since we know that no clause can be in the families of two distinct variables unless one is in the family of the other, and this is clearly impossible since the active clause of the operator rewrite cannot be in any family of the z_{w_j}. Further, they are all subsets of D since any complete path in $D \cup \rho_{\Gamma'}(v)$ from a clause in $\rho_{\Gamma'}(v)$ cannot contain z_{w_j} or it would induce a cycle. Hence we have appropriate subsidiary intermediates and the rewrite can occur, and so the result follows by induction. □

Now using our previous lemma, we can see that we have the consequence:

Proposition 8.5.5
Any intermediate rewrites totally.

This is clear since we know that for any intermediate with a clause set having n elements, it follows from the result we have just proved, for intermediates with clause sets of n or fewer elements. However, we have just proved this last result for all n, and hence the proposition holds.

We now proceed to prove confluence of $\rightarrow_S$ up to the cc-fragment of provable equality on the term part of intermediates.

Lemma 8.5.6 (Relative Confluence)
If an intermediate I has two rewrites, $I \rightarrow_S I_1$ and $I \rightarrow_S I_2$, then I_1 and I_2 respectively have rewrites $I_1 \rightarrow_S I_1'$ and $I_2 \rightarrow I_2'$ such that I_1' and I_2' differ only on their term part, and if I_1' has term part v_1 and I_2' has term part v_2, these two terms are provably equal in the fragment of the equality theory having only the cc axioms and the rules.

Proof We prove this by considering all possible pairs of one-step rewrites in turn. Note that some combinations are prohibited by the restriction on the order of rewrites.

1 **and** 1 If we have two different rewrites of type 1, then clearly the resultant intermediate forms are the same up to exchange on the intuitionistic part of the typing context, and hence we have the result.

Any choice of two from $\{2, 3, 5a\}$ In these cases the result follows again by exchange on the typing context and by commutation of independent substitutions.

2 **and** 4**,** 3 **and** 4 In these cases the result follows by exchange and since the substitutions go through the let -construct.

4 **and** 4 In this case the result follows by the commuting conversion which allows two let -constructs to commute under appropriate free-variable restrictions.

5b and 2, 5b and 3, 5b and 4 Say that the rewrite 2, 3 or 4 is $I \rightarrow_S I_1$, called rewrite a, and that the other rewrite $I \rightarrow_S I_2$ of type 5b is called rewrite b. Now, we know that the active clause of rewrite a is the only clause containing any output variables of the clause, because any other occurrence in D would have to be an input occurrence, which is not possible by our earlier discussion. Now if we consider the subsidiary rewrites of rewrite b, we can see that the output variable of the active clause of the rewrite a can never be introduced into the typing context, and hence the rewrite a can never take place as part of a subsidiary rewrite of rewrite b. This then means that after rewrite b, rewrite a is still possible since its output variable is still in the typing context and the clause is still in the clause set. On the other hand, if we first do rewrite a, clearly we can still do rewrite b since the active clause of rewrite a was not involved, and we have merely altered the typing context in a way which will not affect rewrite b. Having established that the rewrites can be done in either order, the equality of the resultant intermediate forms is given by the commutativity of substitution.

5b and 5b This is the most complex case, since there are two subcases. The first is the case in which the two rewrites of type 5b have different active clauses, and the second is the case in which the active clauses are the same but the sets D'_i are different. First consider the case in which the active clauses are different.

In this case we first note that no clause in one of the sets D_i for one rewrite can be in any of the corresponding set for the other rewrite. This is because any two such sets are sub-relations of the relation $D \cup \rho_\Gamma(v)$, and hence by lemma 8.2.14 any maximal path in $D \cup \rho_\Gamma(v)$ from such a shared clause must pass through one clause having dependent the result variable of the first subrelation and another having dependent the result variable of the second subrelation. If these clauses are the same, then both active clauses, which are the children of the respective result variables, are the same, which is a contradiction. Otherwise, we must have that there is a path from the clause having dependent the result variable of (say) the first subrelation to that having dependent the result variable of the second subrelation. But this then implies that there exists a path from the child of the result variable of the first subrelation to the child of the result variable of the second relation, or in other words a path from the active clause of the first rewrite to that of the second rewrite. This then implies that the first rewrite cannot occur until the second has, which is again a contradiction. Thus it follows that the two rewrites are independent as they do not affect any of the same clauses or variables, and so they can be performed in either order. The resulting intermediates in each case are the same, by virtue of the commutativity of substitution.

Now consider the second case, in which the active clauses of the two rewrites are the same. In order for the two rewrites to be distinct, it must be the case that one of the sets D'_j differs from its counterpart E'_j. Note that for this to be the case, at least one of the sets must contain a clause having dependent z_{w_j} or else both sets would be empty and hence equal. Now we can see that both the sets D'_j and E'_j must be subsets of $\mathsf{fam}(z_{w_j})$ since if not, we would have that some clause having a complete path from itself not including z_{w_j} would be rewritten in the subsidiary rewrite of the 5b-rewrite. This is impossible since we know that for a rewrite to occur all its dependents must be in the typing context, and inductively this is only possible for variables whose children are in the family of the initial variable. Given this and the fact that since $\mathsf{fam}(z_{w_j})$, it suffices to show that the two rewrites using D'_j and using $\mathsf{fam}(z_{w_j})$ respectively have a common reduct. This follows by observations on substitution, however, since any clause which is not in D'_j but in $\mathsf{fam}(z_{w_j})$ must be rewriteable after the 5b-rewrite using D'_j because we simply add the typing context present after the subsidiary rewrite to the main typing context. $\square$

Now we can define a function based on the rewrite $\to_S$. Note that for any $\Gamma;\Delta$-relation D of type A with result variable x, $(x, D, (\_; x : A))$ is an intermediate and hence rewrites totally.

Definition 8.5.7 (The Translation σ)
Given a $\Gamma;\Delta$-relation D such that

$$(x, D, (\_; x{:}A)) \to_S (v, \emptyset, \Gamma'; \Delta')$$

define $\sigma_{\Gamma;\Delta}(D) = v$.

This is a total function by the earlier remarks. Further, we can easily show that v is a $\Gamma;\Delta$-term of type A by considering the structure of the sequentialisation rewrite.

We now need to show that this translation is sound with respect to the equality $=_{vm,\Delta}$. In order to do this we will first prove a lemma:

Lemma 8.5.8
Given an intermediate $(v, D \cup D', (\Gamma, \vec{x} : \vec{Q}; \vec{y} : \vec{A}, \Delta))$ which has a one-step sequentialisation rewrite to another intermediate $(f(v), D, (\Gamma, \vec{x}' : \vec{Q}'; \vec{y}' : \vec{A}', \Delta))$, we have for any other intermediate $(w, E \cup D', (\Gamma', \vec{x} : \vec{Q}; \vec{y} : \vec{A}, \Delta'))$ a rewrite:

$$(w, E \cup D', (\Gamma', \vec{x} : \vec{Q}; \vec{y} : \vec{A}, \Delta')) \rightarrow_S (f(w), E, (\Gamma', \vec{x}' : \vec{Q}'; \vec{y}' : \vec{A}', \Delta'))$$

if E has no clause with a vestigial occurrence of any variable in $\vec{y}$, where f is a function constructed from substitutions of terms for variables and the primitive function which takes v to $\mathsf{let}\ x\ \mathsf{be}\ y\ \mathsf{in}\ v$.

Proof We can prove this by simple consideration of the form of the one-step rewrite. In the case of the first to 5ath rewrite, the set D' must consist of the single clause which is the active clause of the relevant rewrite. Further, the typing sequences $\vec{x} : \vec{Q}$ and $\vec{y} : \vec{A}$ must be precisely the output variables of the active clause. Now given that the intermediate $(w, E \cup D', (\Gamma', \vec{x} : \vec{Q}; \vec{y} : \vec{A}, \Delta'))$ also has the active clause and the output variables of the active clause in the typing context, the only reason the required rewrite might not take place is because of the restriction that a rewrite of type 1 must take precedence over a rewrite of type 3 or 5a on the same output variable. But since the $\vec{y}$ are the linear output variables of the active clause, the condition on the lemma makes sure this cannot happen. □

Lemma 8.5.9 (Soundness)
If two $\Gamma; \Delta$-relations of type A, D_1 and D_2 are equal under the equality $=_{vm,\Delta}$, then $\Gamma; \Delta \vdash \sigma_{\Gamma;\Delta}(D_1) = \sigma_{\Gamma;\Delta}(D_2) : A$ using just the cc-fragment of the provable equality.

Proof In order to show this there are two proof obligations, firstly to show that the rewrite $\rightarrow_S$ is mapped to a provable equality and secondly to show that the equality $=_{vm}$ is preserved by $\sigma_{\Gamma;\Delta}$.

For the first, taking each rewrite in turn, we first take the starting intermediate of a relation, $(x, D, (\Gamma; \Delta))$, and rewrite until the first dependent of a key clause of the rewrite appears in the typing context. Now the result of rewriting the redex and the result of rewriting the reduct at this point are the same, modulo the α-conversion in one case. Therefore the result holds in this case.

The second is much more complex. Imagine that two $\Gamma; \Delta$-relations of type A, $D \cup \{\_ =^{x} y)\}$ and $D \cup \{(\_ =^{x'} y)\}$ are equal via a one-step vestigial move. Without loss of generality, suppose that the path that exists by the first clause of the definition of vestigial move is from the child of x to a clause having dependent x'. Now consider the $\Gamma;\Delta$-intermediates $(z, D \cup \{\_ =^x y)\}, (\_; z{:}A))$ and $(z, D \cup \{(\_ =^{x'} y)\}, (\_; z : A))$ of type A, where z is the result variable of both the relations. Taking the second of these, we first claim that there must exist a sequence of rewrites not containing any rewrite with an active clause having output variable x and not containing any rewrite with active clause $(\_ =^{x'} y)$ as follows:

$$(z, D \cup \{(\_ =^{x'} y)\}, (\_; z{:}A)) \rightarrow_S^* (f(z), D' \cup \{(\_ =^{x'} y)\}, (\Gamma'; x'{:}A', \Delta'))$$

We first note that since the vestigial clause $(\_ =^{x'} y)$ cannot rewrite unless the variable x' is present in the typing context, any rewrite sequence to an intermediate with empty clause set must pass through an intermediate with the variable x' present in the typing context and the vestigial clause $(\_ =^{x'} y)$ in the clause set.

We can now prove by induction over the size of the part of the clause set D of the intermediate that given a path from a child of x to a clause having x' as a dependent, there must either be a rewrite of the intermediate not having active clause with output x, or the typing context contains x'.

Firstly, if the set D is empty, then since any intermediate with non-empty clause set has a rewrite and the only possible rewrite is the one having active clause the only clause in the clause set $(\_ =^{x'} y)$, and this means that x' must be in the typing context, or the rewrite would not be possible.

Secondly, if the set D has $r+1$ elements, it suffices to find one rewrite which satisfies the criteria, as then the resulting intermediate must have a smaller set D. Now since there exists a rewrite by the termination of $\rightarrow_S$, assume that this rewrite has active clause with output x. Then x must occur in the typing context of the intermediate. But we can see from this that if the intermediate is $(v, D \cup \{(\_ =^{x'} y)\}, (\Gamma; x : B, \Delta'))$ every path from x in $\rho_{\Gamma';x:B,\Delta'}(v) \cup D \cup (\_ =^{x'} y)\}$ must be entirely within $\rho_{\Gamma';x:B,\Delta'}(v)$ and hence cannot contain x', which is a contradiction.

But now we need only check the vestigial condition of the previous lemma to see that by repeated applications of it, we have that there exist rewrites:

$$(z, D \cup \{\_ =^x y)\}, (\_; z{:}A)) \rightarrow_S^* (f(z), D' \cup \{\_ =^x y)\}, (\Gamma'; x'{:}A', \Delta'))$$

The vestigiality condition is guaranteed by the existence of the rewrites on the intermediate $(z, D \cup \{\_ =^{x'} y)\}, (\_; z{:}A))$ for all variables except x. So we need to check that at no point in the sequence of rewrites does a rewrite occur which has an active clause with the output variable x. But we already know this.

Now we have a one-step rewrite:

$$(f(z), D' \cup \{(\_ =^{x'} y)\}, (\Gamma'; x' : A', \Delta')) \to (f(z), D', (\Gamma'; x' : A', \Delta'))$$

Also, we have a sequence of rewrites

$$(f(z), D', (\Gamma'; x' : A', \Delta')) \to_S (f'(f(x)), D'', (\Gamma''; x : B, \Delta''))$$

and hence by our previous lemma we also have a sequence of rewrites:

$$(f(z), D' \cup \{\_ =^{x} y)\}, (\Gamma'; x' : A', \Delta')) \to_S (f'(f(x)), D'' \cup \{\_ =^{x} y)\}, (\Gamma''; x : B, \Delta''))$$

The vestigial condition on that lemma is satisfied obviously. Now we have a rewrite

$$(f'(f(x)), D'' \cup \{\_ =^{x} y)\}, (\Gamma''; x : B, \Delta'')) \to_S (f'(f(x)), D'', (\Gamma''; x : B, \Delta''))$$

and therefore we have that both $(z, D \cup \{\_ =^{x} y)\}, (\_; z : A))$ and $(z, D \cup \{(\_ =^{x'} y)\}, (\_; z : A))$ rewrite to $(f'(f(x)), D'', (\Gamma''; x : B, \Delta''))$ so that they must rewrite by confluence to the same intermediate with empty clause set, which then implies that σ is sound with respect to vestigial moves. $\square$

We can now prove that $\sigma_{\Gamma;\Delta}(\rho_\Gamma(v)) = v$ by straightforward calculation on each term construct.

LEMMA 8.5.10 (INVERSION)
Given a $\Gamma; \Delta$-term v of type A, $\sigma_{\Gamma;\Delta}(\rho_\Gamma(v)) = v$.

This then shows by standard arguments that:

LEMMA 8.5.11
$\Gamma; \Delta \vdash v = w : A$ using the cc-fragment of provable equality if and only if $\rho_\Gamma(v) =_\Delta \rho_\Gamma(w)$

We can go further and show that relations under the equality $=_{vm,\Delta}$ are exactly equivalent to terms quotiented by the provable cc-equality. First we prove a lemma:

LEMMA 8.5.12 (INVARIANT)
Given a $\Gamma; \Delta$-intermediate of type A, $(v, D, (\Gamma'; \Delta'))$, such that $(v, D, (\Gamma'; \Delta')) \to_S (w, E, (\Gamma''; \Delta''))$, we have that:

$$\rho_{\Gamma';\Delta'}(v) \cup D =_{vm,\Delta} \rho_{\Gamma'';\Delta''}(w) \cup E$$

as $\Gamma; \Delta$-relations of type A.

This is easily proved by considering each sequentialisation rewrite in turn.

This then allows us to prove the following result:

LEMMA 8.5.13 (INVERSION II)
Given a $\Gamma; \Delta$-relation D of type A, $\rho_{\Gamma;\Delta}(\sigma_{\Gamma;\Delta}(D)) =_{vm,\Delta} D$.

Proof First note that we have a rewrite:

$$(x, D, (\_; x\!:\!A)) \to_S (\sigma_{\Gamma;\Delta}(D), \emptyset, (\Gamma; \Delta))$$

where D has the result variable x in the context $\Gamma; \Delta$. But then by the invariant lemma, we have that

$$\rho_{\_;xA}(x) \cup D =_{vm,\Delta} \rho_{\Gamma;\Delta}(\sigma_{\Gamma;\Delta}(D)) \cup \emptyset$$

which immediately gives us the required result. $\square$

Hence we have that $\mathsf{Rel}(\mathbb{O}, =_{vm,\Delta})$ is isomorphism to $\mathsf{Lin}(\mathbb{O})$ quotiented by the provable equality built from the *cc* axioms and the rules. In the next chapter we consider rewrites on relations which will correspond to the $F - \beta_V\eta$ equality judgements on $\mathsf{Lin}(\mathbb{O})$-terms.

Chapter 9

Normal Forms and Decidability

In this chapter we continue the study of relations as normal forms for linear type-theories, showing that we can extend the results proved in the previous chapter to allow us to decide the linear typing system with $F-\beta_V\eta$-axioms and further to decide the system with output-naturality equalities $\mathsf{ONat}(\mathbb{O},\mathsf{O})$ for some signature $\mathbb{O}$ having an output-parameterised set of operators O. However, to do so will involve some delicate proofs.

9.1 Proof-Nets

As in the previous chapter, we motivate our definitions and proofs using proof-nets, in which we can raise all the problems we will face in a more understandable way.

In modelling not only the provable equality built over $\mathsf{Lin}(\mathbb{O})$ from the commuting conversions, but the full provable equality of $\mathsf{Lin}(\mathbb{O},\mathbb{A})$, we need to consider the $F-\beta_V$ and $F-\eta$ axiomatic equalities. However, there are three issues which are immediately raised.

Firstly, we have the issue of orientation of the rewrites that we shall give. It is by no means certain that the conventional reductive rewrite orientation is the correct one in general, and in fact we will be using a rewrite in which the β_v-rewrites are taken as reductions and the η-rewrites as expansions (see Ghani's thesis [Gha95]). Secondly, the $F-\beta_V$-equality contains within its right-hand side an intuitionistic substitution of an intuitionistic term for a variable, and in order for our relational system to model this, we will need to augment the rewrites of the replicators to copy intuitionistic operators, amongst other administration. The techniques we will use to do this are based in a number of proof–net-like systems, notably those inherited from nets for the λ-calculus.

Thirdly, and perhaps most substantially, we need to take account of the output-naturality equalities which hold of the $\otimes^L$ and I^L operators. In order to do this we adopt a different approach, which is based on the observation that in modelling the linear naturality of operators in proof-nets, we simply leave wires which are not bound by the operator. The intuition behind output-natural operators is that the output wire (or result) is not bound by the operator application, and so our new proof nets proceed by not binding them; for example, a representation of the $\mathsf{DILL}(\mathbb{C})$-term $\mathsf{let}\ x \otimes y\ \mathsf{be}\ t\ \mathsf{in}\ u$ which does not bind the result wire is seen in figure 9.1.

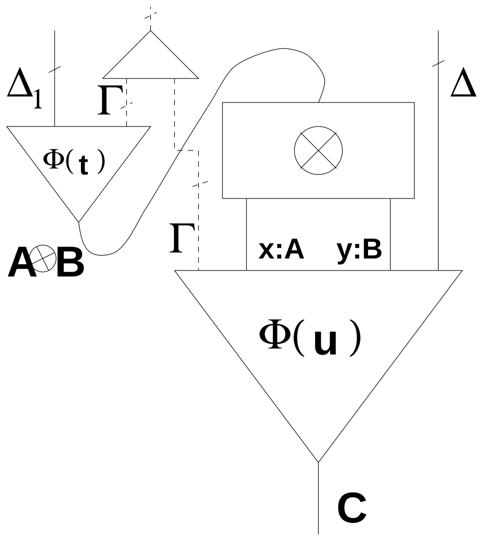


Figure 9.1: Proof-Nets for Output Naturality

Given this approach, output-naturality equalities map to identities on proof-nets, which is one of their major virtues. However, there is a slight complication in the treatment of operators with no binding behaviour, as for example in I^L. As explained in section 8.1 on page 183, such a non-binding operator requires a vestigial, since otherwise too many proof-nets would be identified.

9.2 The F-Equalities

We now introduce an additional rewrite on relations, the F-rewrite, which models the effect of the F-equalities of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$. We proceed to extend the result of the previous section to show that the derivable equality judgement of $\mathsf{Lin}(\mathbb{O}, \emptyset)$ soundly and completely maps to relations under the equality $=_{vm\Delta F}$ generated in the obvious way from the rewrites $\rightarrow_F$ and $\rightarrow_\Delta$ and the equality $=_{vm}$. First we make a definition:

Definition 9.2.1 (Resolved Variable)
Say a variable x of type $Q \in \mathsf{M}_I$ is *resolved* in the relation D if it is linear and is not the input of a clause $(F(z) = x)$ or the output of a clause $(x = F(z))$. Conversely, if a variable is linear and is not resolved, call it unresolved.

We must define the concept of common variable. A common variable for a clause in a relation is a linear variable which occurs in any complete path from the clause. Each clause has at least one common variable unless it has dependent the result variable, because the result variable will be a common variable. Further, for any clause having an input which occurs as a vestigial elsewhere in the relation, we can always find a common variable which is a valid vestigial move away. This holds simply by taking the common variable which occurs first in each complete path (it must always be the same one since if not a cycle would occur).

Now we define the rewrite on relations which we will use:

Definition 9.2.2 (The Rewrite $\to_F$)
Define the one-step rewrite $\to_F$ on relations using the following rules, where the $F - \eta$-rule is only permitted on unresolved variables x:

$$D \cup \{(F(x') = y), (y = F(x))\} \to_{F-\beta} D\{x/x', z/y\}$$
$$D\{x/x'\} \to_{F-\eta} D \cup \{(F(y) = x), (x' = F(y))\}$$

where in the second of these, y is fresh, and in the first, z is the first common variable in every complete path from $(F(x') = y)$ in the relation $D \cup \{(F(x') = y), (y = F(x))\}$.

Now we need to extend the rewrite Δ in order to deal with intuitionistic operators. We will overload notation to refer to the extended rewrite also as $\to_\Delta$. We now introduce some substitution notation; for any relation D let $D\sigma(1)$ be D with all its variables replaced by fresh variables, such that if $i \neq j$ then $D\sigma(i)$ and $D\sigma(j)$ share no variables. We extend this to intuitionistic operator clauses, meaning that each relation within the operator instance has the substitution applied to it, with corresponding α-conversion for the binding instances.

Definition 9.2.3 (The Rewrite $\rightarrow_\Delta$)
Let the one-step rewrite $\rightarrow_\Delta$ be given by the following cases, extending the rewrite $\rightarrow_\Delta$ of the previous chapter:

$$D \cup \{(\_ =^x y), (M \cup \{z'\} \cup \{y'\} = z)\} \rightarrow_{\Delta-1} D \cup \{(M \cup \{z'\} = z)\}$$
$$D \cup \{(\{x\} = y)\} \rightarrow_{\Delta-2} D\{y/x\}$$
$$D \cup \{(M' = y), (M \cup \{z'\} \cup \{y\} = x)\} \rightarrow_{\Delta-3} D \cup \{(M \cup \{z'\} \cup M' = x)\}$$

$$D \cup \{(z = (\vec{x}; \vec{y})O(\ldots(\vec{x}'; \vec{y}')D' \cup \{(M = y'')\}(x'') \ldots ; \ldots))\} \rightarrow_{\Delta-4}$$
$$D \cup \{(z = (\vec{x}; \vec{y})O(\ldots(\vec{x}'; \vec{y}')D'(x'') \ldots ; \ldots)), (M = y'')\}$$

$$D \cup \{(z = (\vec{x}; \vec{y})O(\ldots(\vec{x}'; \vec{y}')D'(x'') \ldots ; \ldots)), (z' = (;)O'(\ldots ;))\} \rightarrow_{\Delta-5}$$
$$D \cup \{(z = (\vec{x}; \vec{y})O(\ldots(\vec{x}'; \vec{y}')D' \cup \{(z' = (;)O(\ldots ;))\}(x'') \ldots ; \ldots))\}$$
where $O' \in \mathcal{O}_I$ and z' occurs as an input only in D'

$$D \cup \{(M = z), (z = (;)O'(\ldots ;))\} \rightarrow_{\Delta-6}$$
$$D \cup \{(z'_1 = (;)O'(\ldots ;)\sigma(1)), \ldots, (z'_r = (;)O'(\ldots ;)\sigma(n)), (M'_1 \ldots M'_r = \vec{x})\}$$

where in this last rewrite, the free intuitionistic variables occurring as inputs in $(;)O'(\ldots ;)$ are $\vec{x}$, we have $M'_i = \vec{x}\sigma(i)$ and $M = \{z'_1, \ldots, z'_r\}$.

We will now define the equality $=_{vm\Delta F}$ in the obvious way as the reflexive transitive closure of the union of the one-step rewrites $\rightarrow_\Delta$ and $\rightarrow_F$ over the equality $=_{vm}$.

We can now start to extend the results of the previous chapter to these new rewrites.

9.3 Deciding the Equality

Confluence

It is straightforward to check confluence for our rewrite by considering all possible critical pairs.

Lemma 9.3.1 (Confluence)
The transitive closure $\rightarrow_{\Delta F}$ of the union of the rewrites $\rightarrow_F$ and $\rightarrow_\Delta$ is confluent over the equality $=_{vm}$.

Proof As before, the key to the proof is the locality of the rewrites. However, since we now have rewrites which manipulate operator clauses, we have to also allow for the possibility that redexes may occur inside the relations which are in the operator clause. We consider pairs of rewrites case by case.

Any of $\{\Delta - 1, 2, 3\}$ with $F - \beta$ In this case we can clearly see that there is no possibility of these two rewrites sharing a clause. Hence the rewrites are independent.

Any of $\{\Delta - 1, 2, 3\}$ with $F - \eta$ In this case we can see that an $F - \eta$-rewrite can never occur on any variable in a $\Delta - 1, 2, 3$-redex, since $F - \eta$ can only occur on linear variables and all those occurring in the rewrites $\Delta - 1, 2, 3$ are intuitionistic. Hence the rewrites are independent.

Either of $\Delta - 4$ or $\Delta - 5$ with either of $F - \beta$ or $F - \eta$ In this case we can see that the only interaction between these two redexes can be if the F-redex occurs in an operator instance occurring in the Δ-redex. But no F-clause or linear variable in such an operator instance is affected, and so the rewrites are independent.

The $\Delta - 6$-rewrite and either the $F - \beta$ or $F - \eta$-rewrite In both these cases we can see that the F-redex may be copied any number of times (including none, when it is deleted). The result of doing the F-rewrite and then the $\Delta - 6$ rewrite is the same as that of doing the $\Delta - 6$-rewrite and then some number of copies of the F-rewrite.

Any combination of two from $\{\Delta - 1, 2, 3\}$ This case was dealt with in the previous proof.

$\Delta - 4$ and itself In this case clearly the redexes are non-overlapping unless the two duplicators are leaving the same operator construct, in which case they are still independent.

$\Delta - 4$ and any of $\{\Delta - 1, 2, 3\}$ This case is easily seen, with the rewrites only interacting if the duplicator which leaves the operator construct in the $\Delta - 4$-rewrite is also involved in the $\Delta - 1, 2, 3$-rewrite. If the rewrites do interact, the reducts will themselves have a common reduct using more instances of the same rewrites.

$\Delta - 6$ and and any of $\{\Delta - 1, 2, 3\}$ This is very similar to the previous case, we the added possibility that the $\Delta - 1, 2, 3$-rewrite might be duplicated.

$\Delta - 6$ and $\Delta - 5$ These rewrites interact only if the copied operator construct is the operator construct which is the active one in the $\Delta - 5$ rewrite, or in the case where the $\Delta - 6$-rewrite copies the $\Delta - 5$ redex. In the second case the two reducts can easily be shown to have a common reduct by repeated $\Delta - 5$ rewrites, and in the first case another instance of $\Delta - 6$ and a number of instances of $\Delta - 5$ are needed.

$\Delta - 6$ and $F - \beta\eta$ These rewrites can only interact if the $F - \beta\eta$-redex occurs in the operator construct duplicated by the $\Delta - 6$ rewrite, and we can easily rewrite the reducts to a common form using a number of $F - \beta\eta$-rewrites and a$\Delta - 6$-rewrite.

Any two of $F - \beta\eta$ These rewrites may share a clause, but if they do the result of performing either rewrite first is the same form.

$\Delta - 5$ and $\Delta - 1, 2, 3$ These rewrites must be independent, as they share no clause and no rewrite in the body of the operator clauses is affected by the $\Delta - 5$ rewrite.

$\Delta - 5$ and $\Delta - 4$ In this case the rewrites may interact in two interesting ways, one for each operator construct in the $\Delta - 5$ rewrite. In the case where the $\Delta - 4$-rewrite places a replicator between the two operator clauses of the $\Delta - 5$rewrite, the two reducts have a common reduct using a $\Delta - 6$-rewrite and possibly several $\Delta - 5$-rewrites. In the other case, the required rewrites to form a common reduct are simply two $\Delta - 5$ rewrites.

$\Delta - 5$ and $\Delta - 5$ In this case, the only interesting interaction between the two instances of the rewrite can be resolved simply using more instances of $\Delta - 5$.

Normalisation

Having established the confluence of this rewrite, we need to show that it is normalising, which is to say that every one-step rewrite sequence is finite. Although this is not as hard as it is for the higher-order case, because of the presence of the expansionary rewrites $F - \eta$ and $\Delta - 6$ it is not trivial. However, it can be shown by defining a measure on terms which we will now present.

First consider the action of our rewrites on an arbitrary replicator in a relation. In each of the rewrites $\Delta - 1, 2, 3$ and $\Delta - 6$ the tendency is for a replicator to move further away from the result variable of a relation, or to disappear altogether. The problem is that in the case of the $\Delta - 6$-rewrite, this is accomplished at the cost of increasing the number of clauses, which in the previous chapter was the measure which ensured termination. This is the key to the measure we will define, but there are complexities due to the $\Delta - 4$ and $\Delta - 5$ rewrites.

In the case of the $F - \eta$-rewrite, we notice that the rewrite cannot occur unless the variable is unresolved. Since all the variables introduced by a $F - \eta$-rewrite are resolved, we can make such rewrites reduce the measure by weighting each unresolved variable more heavily than the subnet which replaces it after the rewrite.

First we introduce some theory. We will be using polynomials of a single variable, which we will write $p(x), q(x) \ldots$ for the variable x; polynomials will only be used in this section, so this slight overloading of notation will be transparent. A general polynomial in a single variable has the form $\sum_{i=-r\ldots s} c_i x^i$ where the c_i are (possibly negative) integer coefficients. We can then define addition and subtraction (and the product of two polynomials) in the obvious way We can also define an ordering on polynomials of a single variable by saying $p(x) > q(x)$ iff the coefficient of the largest power of x in $p(x) - q(x)$ is strictly positive.

We now need to make some definitions formalising our earlier discussion of the measure. Recall that a path from c to c' in a relation D is a pair of sequences $(c_1 \ldots c_r, x_1 \ldots x_{r-1})$ such that $c_1 = c$, $c_r = c'$ and $_i$ connects c_i and c_{i+1} for all $i = 1 \ldots r - 1$. Now say that a *total path* in a relation D is a path which cannot be extended by prepending or appending a pair (c, x) to it. Intuitively, the total paths go from free inputs or inputs bound by an operator clause to a clause having dependent the result variable of either the relation D, or some relation D' in an operator clause contained in D. Write the set of all total paths in a relation D as $\mathsf{TPath}(D)$, and let $\phi \ldots$ range over paths for this chapter only. Clearly there exists a total path containing any clause contained in D.

Definition 9.3.2 (Complexity I)
Given a path ϕ, define the *first complexity* of it, written $\mathsf{Com}^1(\phi)$ inductively:

$$\mathsf{Com}^1(c, \varepsilon) = 1$$

$$\mathsf{Com}^1(\vec{c}c', \vec{x}x') = \begin{cases} 2\mathsf{Com}^1(\vec{c}, \vec{x}) & \text{if } c' \text{ is a replicator clause and } x' \text{ is resolved} \\ 2\mathsf{Com}^1(\vec{c}, \vec{x}) + 4 & \text{if } c' \text{ is a replicator clause and } x' \text{ is unresolved} \\ \mathsf{Com}^1(\vec{c}, \vec{x}) + 1 & \text{if } c' \text{ is not a replicator clause} \\ & \text{and } x' \text{ is resolved} \\ \mathsf{Com}^1(\vec{c}, \vec{x}) + 4 & \text{if } c' \text{ is not a replicator clause} \\ & \text{and } x' \text{ is unresolved} \end{cases}$$

Define the first complexity polynomial as follows:

Definition 9.3.3 (First Complexity Polynomial)
The first complexity polynomial for a relation D is given:

$$p^{1C}(D)(x) = \sum_{\phi \in \mathsf{TPath}(D)} x^{\mathsf{Com}^1(\phi)}$$

Now the intuition is that all our rewrites will at worst replace one path of a certain complexity with many of lower complexity, but because in the polynomial ordering one x^c of higher coefficient outweighs many $x^{c'}$ of lower coefficient, the complexity polynomial will still strictly decrease. We can now show that the one-step $\Delta - 1, 2, 3, 6$-rewrites and one-step $F - \beta\eta$-rewrites decrease the first complexity polynomial of a relation.

Lemma 9.3.4
If $D \to D'$ by a one-step $\Delta - 1, 2, 3, 6$-rewrite or a one-step $F - \beta\eta$-rewrite, then $p^{1C}(D)(x) > p^{1C}(D')(x)$.

Proof We will refer to the form of the rewrites given on page 217. Consider the rewrites separately:

$\Delta - 1$ In this case, any total path in D' yields a total path in D simply by replacing any occurrence of the new replicator clause added by the rewrite with the replicator clause in D which has as output the required variables. Further, the first complexity of the generated path in D is the same as that of the original path in D', since we have just replaced a replicator clause with another one. We can see easily that any two distinct total paths in D' give rise to distinct total paths in D via this procedure. Moreover, there exists a total path in D which is not a total path in D', namely that which passes through the replicator clause deleted by the rewrite. Call this total path ϕ. Now we know that the first complexity polynomial of D must have coefficients greater than or equal to those of the first complexity polynomial of D'. But the coefficient of $x^{\mathsf{Com}^1(\phi)}$ must be strictly greater in the first complexity polynomial of D, so that this polynomial must be strictly greater than the first complexity polynomial of D'.

Δ_2 In this case we can see that any total path of D' arises from a total path of D either identically, or by replacing the subsequence of the variable sequence yx with y and removing the replicator clause in the clause sequence. This procedure obviously yields total paths of less than or equal first complexity in general, and in the case where the replicator clause is removed yields total paths of strictly lesser first complexity. Note that there exists at least one total path which has strictly lesser first complexity by this argument since at least one total path passes through the replicator clause. This correspondence gives a one-to-one mapping of total paths and hence since it is strictly reducing on the first complexity of at least one total path, the first complexity polynomial of the reduct D' must be strictly less than that of D.

$\Delta - 3$ This argument proceeds similarly to the previous one simply by constructing the obvious correspondence between total paths, which turns out to be strictly reducing of first complexity on total paths passing through the replicator clause which is deleted.

$\Delta - 6$ This is the key case. It is easy to see that given any total path in D which is not a path of D', we can give a total path of D' which has strictly lesser first complexity. Further, although there are many such candidate total paths, these all have strictly lesser first complexity, and the only total paths of D' which are not paths of D are these paths of lesser first complexity. Consider the total path ϕ that we know must exist in D passing through the replicator, and which is therefore not a path of D'. The first complexity polynomial of D has strictly greater $x^{\mathsf{Com}^1(\phi)}$ coefficient than the first complexity polynomial of D', and although by our discussion it may have strictly lesser coefficients of smaller powers of x, this is still enough to make the first complexity polynomial of D strictly greater than that of D'.

$F - \beta$ In this case, there is a straightforward correspondence between total paths of D and D', which strictly reduces the first complexity of any path that passes through either of the clauses which are deleted. Hence by standard reasoning the first complexity polynomial of D is strictly greater than that of D'.

$F - \eta$ This is a more complicated case since the obvious correspondence between paths increases the length of paths. However, it can be shown that because the paths contain one fewer unresolved variable, the first complexity of paths is strictly reduced by going from D to D' and hence the first complexity polynomial of D is strictly greater than that of D'. □

This forms the major part of our proof. We can proceed to consider the only two remaining rewrites, the $\Delta - 4$ and $\Delta - 5$-rewrites. First we note that the first-complexity polynomial is preserved by these one-step rewrites, by virtue of the lemma:

Lemma 9.3.5
If $D \to D'$ by a one-step $\Delta - 4$ or $\Delta - 5$-rewrite, there is a first-complexity-preserving correspondence between the total paths of D and D'.

Proof The first-complexity-preserving correspondence is the identity. □

Let $\mathsf{Op}(D)$ be the set of operator clauses contained in D, and define the nesting level of a clause as follows:

Definition 9.3.6 (Nesting Level)
Given a clause $c \in D$, define the *nesting level* of c in D, written $\mathsf{Nest}(D)(c)$, inductively as follows:

$$\begin{aligned}
&\mathsf{Nest}(D)(c) = 0 \text{ if } c \in D\\
&\mathsf{Nest}(D)(c) = \mathsf{Nest}(D')(c) + 1\\
&\qquad\qquad \text{if } (y = (\dots)O(\dots,, (\vec{x}'_1; \vec{x}'_2)D(x''), \dots; \dots)) \in D\\
&\qquad\qquad \text{and } c \text{ is contained in } D'
\end{aligned}$$

Now we can define the nesting polynomial.

Definition 9.3.7 (Nesting Polynomial)
Define the *nesting polynomial*, of a relation D, written $p^N(D)(x)$, as follows:

$$p^N(D)(x) = \sum_{c \in \mathsf{Op}(D)} x^{-\mathsf{Nest}(D)(c)}$$

Clearly, we can now prove:

Lemma 9.3.8
If $D \to D'$ by a one-step $\Delta - 5$-rewrite, then $p^N(D)(x) > p^N(D')(x)$.

Proof This is quite easy to see since the rewrite reduces by one the coefficient of $x^{-\mathsf{Nest}(D)(c)}$ where c is the intuitionistic operator clause which is moved inside the other operator clause in the rewrite, and no other operator clause of lesser nesting (which means higher power of x in the nesting polynomial) is affected.□

Also, we can quite easily see that since the one-step $\Delta - 4$-rewrite does not affect the nesting level of any operator clause, it must preserve the nesting polynomial. Hence, we have almost completed the details of normalisation. Finally, let $\mathsf{Rep}(D)$ be the set of replicator clauses occurring in the relation D. We have;

Lemma 9.3.9
If $D \to D'$ by a one-step $\Delta - 4$ rewrite, then

$$\sum_{c \in \mathsf{Rep}(D)} \mathsf{Nest}(D)(c) > \sum_{c \in \mathsf{Rep}(D')} \mathsf{Nest}(D')(c)$$

Proof Again, this is easily seen, since the rewrite precisely decreases the nesting level of the replicator clause involved by one, and leaves all other replicators as they are. $\square$

We can now put all these pieces together to prove:

LEMMA 9.3.10 (NORMALISATION)
The transitive closure $\to_{\Delta F}$ of the union of the rewrites $\to_F$ and $\to_\Delta$ over the equality $=_{vm}$ is normalising.

Proof Consider the set of triples $(p(x), q(x), r)$ consisting of two polynomials and an integer, ordered by the lexicographic ordering inherited from the polynomial ordering and the integer ordering. We claim that the measure

$$(p^{1C}(D)(x), p^N(D)(x), \sum_{c \in \mathsf{Rep}(D)} \mathsf{Nest}(D)(c))$$

forces the $\to_{\Delta - F}$-rewrite to terminate.

In order to show this we need to show that the measure is bounded below in the ordering for all relations, and that every one-step rewrite strictly reduces the measure according to the given ordering. We can see firstly that the measure is bounded below by the element $(0, 0, 0)$ of the measure set, where the polynomial 0 is that polynomial having all zero coefficients. This is the case since clearly the two polynomials must have nonnegative coefficients which are just the sizes of sets in each case (the set of total paths of complexity n and the set of operator clauses of nesting level n), and the nesting level of any clause must be nonnegative.

Secondly, to show that the measure is reduced by each one-step rewrite note that the $\Delta - 1, 2, 3$ and $F - \beta\eta$-rewrites strictly reduce the first measure by lemma 9.3.4, that the $\Delta - 5$ one-step rewrite preserves the first measure and reduces the second by lemma 9.3.8, hence reducing the measure in the lexicographic order, and finally that the $\Delta - 4$ one-step rewrite preserves the first-complexity polynomial and the nesting polynomial, and reduces the final component of the triple by lemma 9.3.9. Hence there can only be a finite number of rewrites in any sequence of one-step $\Delta - 1, 2, 3, 4, 5, 6, -F - \beta\eta$-rewrites. $\square$

Hence we have that:

THEOREM 8 (DECIDABILITY)
The equality $=_{vm\Delta F}$ is decidable.

Proof This is easy to see since any rewrite sequence on any two relations will terminate, and by confluence we must have that if the reducts are equal, then they are equal by $=_{vm}$, which is also decidable. $\square$

Relations and Terms

Now we must show that the extended rewrite we have defined over terms is soundly mapped to and from term equality. We start by considering the translation ρ.

First we need a lemma on intuitionistic substitution.

LEMMA 9.3.11 (INTUITIONISTIC SUBSTITUTION)
Given an intuitionistic term $\Gamma; \_ \vdash v\!:\!Q$ of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$, for any term $\Gamma, x\!:\!Q; \Delta \vdash w\!:\!A$, we have

$$\rho_{\Gamma;\Delta}(\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w) = \rho_{\Gamma;\Delta}(w\{v/x\})$$

The proof is by induction on the derivation of the term w. However, we note that if the intuitionistic term w is simply some intuitionistic variable y, then $\rho_{\Gamma;\Delta}(\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ w)$ is

$$\rho_{\vec{z}_1:\vec{R},y:Q,x:Q;\Delta}(w) \cup \{(\{\vec{z}_1, \vec{z}_2\} = \vec{z}), (\{\} =^{y'} \vec{z}_2), (y' = F(y)), (F(x) = y')\}$$

and this rewrites to

$$\rho_{\vec{z}_1:\vec{R},y:Q,x:Q;\Delta}(w)\{\vec{z}, y/\vec{z}_1, x\}$$

as required.

LEMMA 9.3.12 (SOUNDNESS)
If $\Gamma; \Delta \vdash v = w\!:\!A$ in $\mathsf{Lin}(\mathbb{O}, \emptyset)$, then $\rho_{\Gamma;\Delta}(v)$ and $\rho_{\Gamma;\Delta}(w)$ have a common reduct in the transitive closure of $\rightarrow_{\Delta F}$.

This is again easily shown by calculation.

Now considering the translation σ, we have

LEMMA 9.3.13
Given $\Gamma; \Delta$-relations D and D' of type A, if D rewrites to D' by a one-step F- or Δ-rewrite, then

$$\Gamma; \Delta \vdash \sigma_{\Gamma;\Delta}(D) = \sigma_{\Gamma;\Delta}(D')\!:\!A$$

in $\mathsf{Lin}(\mathbb{O}, \emptyset)$.

This is proved by considering suitable rewrite sequences and the term parts of the resulting intermediates.

Given these results, we have the following corollary of theorem 8:

COROLLARY 8.1 (DECIDABILITY)
Given two $\Gamma; \Delta$-terms v and w of type A, we can decide whether or not they are provably equal in the type-theory $\mathsf{Lin}(\mathbb{O}, \emptyset)$.

This follows easily since we can see that the two terms are equal iff their images under ρ are equal, and this latter statement is decidable.

9.4 The Higher-Order Case

We now need to consider decidability in the higher-order case. We will consider the system of relations for the higher-order type-theory constructed in chapter 7. We will then give a confluent and normalising rewrite system for it built on the one we have just presented for a general linear type-theory.

Intuitively, we will take the expected β-reductions and η-expansions on unresolved variables, where we will extend the definition of unresolved variables appropriately. Then we will be able to extend the definition of complexity of a path, and define a revised version of the complexity polynomial which will still enjoy the same properties.

Firstly, however, we need to consider the system of relations for higher-order type-theories.

Higher-Order Relations

The system of higher-order relations for $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$ is the same as the system of relations for the generalised linear type-theory which defines $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$, with the one exception that we change the syntax for the output-natural sets of operators $\otimes^L_{A,B}$ and I^L.

First consider the system $\mathsf{Rel}(\mathbb{O}^{\mathbb{H}})$. We will use the following abbreviations:

$$\begin{gathered}
\text{write } (x = *) \text{ for } (x = (;)I^R(;)) \\
\text{write } (z = x \otimes y) \text{ for } (x = (;) \otimes^R (;(;)(x)(;)(y))) \\
\text{write } (z = \lambda x.y) \text{ for } (z = (;)\lambda(;(;x)(y))) \\
\text{write } (z = xy) \text{ for } (z = (;xy)\mathsf{ap}(;)) \\
\text{write } (z = !(\vec{x})D(y)) \text{ for } (z = (;)!((\vec{x};)D(y);)) \\
\text{write } (y = \mathsf{i}x) \text{ for } (y = (;x)\mathsf{i}(;))
\end{gathered}$$

Now we need to define the relational system $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$, in which the I^L and $\otimes^L$ sets of operators are captured using different syntax. Essentially, this is given by replacing every occurrence of a tensor elimination operator $(z' = (;x') \otimes^L (;(x_1x_2)(z)))$ with a clause $(x_1 \otimes x_2 =^z x')$ and replacing z' by z in the rest of the relation, and similarly for the I^L operator. The intuition is that the output-naturality of the tensor-elimination and unit-elimination operators corresponds to the fact that the vestigial in the new clauses should be movable.

Definition 9.4.1 (Clauses of $\mathrm{Rel}^{\mathrm{H}}(\mathbb{H})$)
Given a higher-order signature $\mathbb{H} = (\mathsf{M}_I, \mathsf{M}_L, \mathcal{O}_I, \mathcal{O}_L)$, the *clauses* of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ are

precisely those of $\mathsf{Rel}(\mathsf{M}^{\mathsf{H}}_I, \mathsf{M}^{\mathsf{H}}_L, \mathcal{O}_I, \mathcal{O}_L)$ with the addition of:

$$\begin{array}{cccc} (x = *) & (z = x \otimes y) & (z = \lambda x.y) & (z =!(\vec{x})D(y)) \\ (* =^z x) & (x \otimes y =^{z'} z) & (z = xy) & (y = \mathsf{i}x) \end{array}$$

Now,

- In the clause $(x = *)$, x is an output occurrence.
- In the clause $(z = x \otimes y)$, z is an output occurrence and x and y are both input occurrences.
- In the clause $(z = \lambda x.y)$, z is an output occurrence, x is a binding occurrence and y is an input occurrence.
- In the clause $(z =!(\vec{x})D(y))$, z is an output occurrence.
- In the clause $(* =^z x)$, z is a vestigial occurrence and x is an input occurrence.
- In the clause $(x \otimes y =^{z'} z)$, x and y are output occurrences, z' is a vestigial occurrence and z is an input occurrence.
- In the clause $(z = xy)$, z is an output occurrence and x and y are input occurrences.
- In the clause $(y = \mathsf{i}x)$ y is an output occurrence and x is an input occurrence.

Pre-relations of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ are sets of clauses of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$, and we now define both connectedness and paths exactly as for $\mathsf{Rel}(\mathbb{O})$ given the extended definition of input, output, vestigial and binding occurrences.

DEFINITION 9.4.2 (RELATIONS OF $\mathrm{REL}^{\mathrm{H}}(\mathbb{H})$)
A *relation* of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ is a pre-relation of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ which satisfies the familiar conditions of definition 8.2.10 for $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ paths.

We can take pre-relations of $\mathsf{Rel}(\mathbb{O}^{\mathbb{H}})$ to those of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ by mapping each non-higher-order clause to itself, and mapping each higher-order clause to its familiar abbreviation except for the clause $(z' = (;x)I^L(;(;)(z))$, which we map to $(* =^z x)$ whilst substituting z for the input occurrence of z', and the clause $(z' = (;x') \otimes^L (;(;x_1x_2)(z)))$, which we map to $(x_1 \otimes x_2 =^z x')$, again substituting z for the input occurrence of z'.

Conversely, we can take pre-relations of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ to those of $\mathsf{Rel}(\mathbb{O}^{\mathsf{H}})$ by mapping each non-higher-order clause to itself, and mapping each higher-order clause to its familiar abbreviation except for the clause $(* =^z x)$, which we map to $(z' = (;x)I^L(;(;)(z)))$ where z' is fresh, and we substitute it for the input occurrence of z, and the clause $(x_1 \otimes x_2 =^z x')$, which we map to $(z' = (;x') \otimes^L (;(;x_1 x_2)(z)))$ where z' is fresh, and we substitute it for the input occurrence of z. The result we now want is as follows:

Lemma 9.4.3
The relations of $\mathsf{Rel}(\mathbb{O}^{\mathbb{H}})$ are isomorphic to the relations of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$.

This is easily seen by considering the mappings we have already given, and observing that they map paths to paths. Further, we have more importantly that:

Lemma 9.4.4
Under the isomorphism of relations, the vestigial moves and $\to_{F\Delta}$-rewrites of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ are isomorphic to the vestigial moves and $\to_{F\Delta}$ augmented with the equalities in $\rho(\mathsf{ONat}(\otimes^L, \mathbb{O}^n\mathbb{H}))$ and $\rho(\mathsf{ONat}(I^L, \mathbb{O}^{\mathbb{H}}))$, where these equalities correspond to the vestigial moves of the clauses for $\otimes^L$ and I^L.

We assume firstly that the isomorphism of relations preserves rewrites in both directions, and also preserves the vestigial equalities of operators other than the tensor and unit eliminations. This then leaves us with two proof obligations. The first of these is to show that two relations which are equal via the image under ρ of the output-naturality equalities for the tensor and unit-elimination map to relations which are vestigially equal using the vestigial equalities for those connectives. This is most easily done by demonstrating that the image under ρ of the output-naturality equalities can be derived from a larger set of simpler equalities along the lines of those sketched on page 40, which correspond directly to the individual vestigial moves.

The second proof obligation is to show that relations which are vestigially equal using the vestigial equality for the tensor and the unit map to relations which are equal via the image under ρ of the output-naturality equalities. This is done by considering the elementary vestigial moves on which the equality was built. These are all easily shown to correspond to elementary output-naturality equalities.

Rewriting

We will define a rewrite over the relations of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ which extends the rewrite previously given over $\mathsf{Rel}(\mathbb{O})$. Firstly, we need to extend the definition of resolved wires in order to restrict the η-rewrite.

Definition 9.4.5 (Resolved Variables)
Extend the definition of resolved variables in the higher-order case so that, given a relation D:

- a variable of a type $Q \in \mathsf{M}_I$ is resolved if it is intuitionistic and it is not of the form $!A$, or if it is linear and it is the input of a clause $(F(y) = x)$ or the output of a clause $(x = F(y))$,
- a variable of type I is resolved if if it is the input of a $(* =^z x)$ clause or the output of a $(x = *)$ clause,
- a variable of type $A \otimes B$ is resolved if is the input of a clause $(x \otimes y = x)$ or the output of a clause $(x = y \otimes z)$,
- a variable of type $A \multimap B$ is resolved if is the input of a clause $(z = xy)$ or the output of a clause $(x = \lambda y.z)$,
- and finally, an intuitionistic variable of type $!A$ is resolved if it is the input of a clause $(y = F(x))$ where y is the input of a clause $(z = \text{¡}y)$, or if it is the output of a clause $(x =!(\vec{y})D'(z))$.

We now say that the *order* of a type is given by the number of connectives used in its construction, where $I, \otimes, \multimap$ and $!$ are constructors for the purpose of this definition.

We give the rewrite $\rightarrow_{\beta\eta}$ on the relational system $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$.

Definition 9.4.6 (The Rewrite $\rightarrow_{\beta\eta}$)
Define the one-step rewrite $\rightarrow_{\beta\eta}$ to be given by the following cases:

$$
\begin{aligned}
D \cup \{(* =^y x), (x = *)\} &\rightarrow_\beta D \\
D \cup \{(x \otimes y = z), (z = x' \otimes y')\} &\rightarrow_\beta D\{x', y'/x, y\} \\
D \cup \{(z = \lambda x.y), (y' = zx')\} &\rightarrow_\beta D\{x', y/x, y'\} \\
D \cup \{(z =!(\vec{x})D'(y)), (x' = F(z)), (y' = \text{¡}x')\} &\rightarrow_\beta (D \cup D')\{y/y'\}
\end{aligned}
$$

$$
\begin{aligned}
D\{x/x'\} &\rightarrow_\eta D \cup \{(x' = *), (* =^{x'} x)\} \\
D\{x/x'\} &\rightarrow_\eta D \cup \{(x' = y \otimes z), (y \otimes z = x)\} \\
D\{x/x'\} &\rightarrow_\eta D \cup \{(x' = \lambda y.z), (z = xy)\} \\
D\{x/x'\} &\rightarrow_\eta D \cup \{(x' =!(x)\{(z = \text{¡}y), (y = F(x))\}(z))\}
\end{aligned}
$$

Here, η-rewrites are only allowed on unresolved variables, which must also in the case of the $I - \eta$ rewrite be variables of type I, in the case of the $\otimes - \eta$-rewrite must be variables of type $A \otimes B$ for some A and B, in the case of the $\eta - \multimap$-rewrite must be variables of type $A \multimap B$ for some A and B, and finally in the case of the $! - \eta$-rewrite must be intuitionistic variables of type $!A$ for some A.

Confluence

Confluence is shown for this system of rewrites in exactly the way we showed if for the last, which is by considering all possible interacting pairs of local rewrites. We are saved some work in this by noting that the only rewrites which can interact in the $\beta\eta$-rewrite are the β-rewrite and the η-rewrite corresponding to the same type constructor.

LEMMA 9.4.7 (CONFLUENCE OF $\rightarrow_{F-\Delta-\beta\eta}$)
The transitive closure of the one-step rewrite on $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ defined by the various cases of $\rightarrow_F$, $\rightarrow_\Delta$ and $\rightarrow_{\beta\eta}$ is confluent.

Proof In order to do this it suffices to show that any two one-step rewrites of these kinds have reducts which in turn have a common reduct in the transitive closure. We have already seen that this is true for any two one-step rewrites of type $\rightarrow_{\Delta-F}$ in the case of $\mathsf{Rel}(\mathbb{O})$, and the proof in that case goes through identically in this case. Further, as we observed above, the only way any two one-step $\rightarrow_{\beta\eta}$-rewrites might interact is if they were the β and the η-rewrite corresponding to the same constructor. In this case, any interaction is seen to be impossible by inspection. Equally, it is obvious that (excepting the case of the !-constructor) the only interaction between a $\beta\eta$-rewrite and a $F\Delta$-rewrite is in the case where the $\beta\eta$-rewrite in an intuitionistic operator clause and is duplicated or discarded by an instance of $\Delta - 6$. In this case we can see that the confluence holds using a number of the same $\beta\eta$-rewrites. Finally, considering the !-constructor, we must check that the non-standard way in which the $! - \beta\eta$-rewrites include instances of the clause $(x = F(y))$ does not interfere with confluence, but it can be see that in the critical cases, those in which we consider the $F - \beta\eta$-rewrites and the $! - \beta\eta$-rewrites, there is no failure of confluence. Hence we have the result.
□

Normalisation

The intuition behind the proof of normalisation for this case is exactly the same as that in the previous case, except that in order to account for the fact that the η-rewrites are expansionary, we need to weight unresolved variables according to the order of their type. Then the intention is that when an η-rewrite occurs, an unresolved variable of a higher type order is replaced by some possibly unresolved variables *of lower type order* and some clauses in the relation. Hence the overall effect is to reduce the complexity of each path through the η-redex and hence the complexity polynomial.

There is also a factor which we have not mentioned, which is that the result of a β-rewrite can increase this measure of complexity for many paths since any total path whose first clause has as input the binder of the λ-clause will not be total after the rewrite since the variable is no longer bound. In order to take care of this problem, we need to weight every application clause with the maximal complexity of the highest paths above its argument input.

First we define a measure of the *order* of a type.

DEFINITION 9.4.8 (TYPE ORDER)
Define the *order* of a type A inductively, written $\mathsf{o}(A)$ as follows:

$$\begin{aligned}\mathsf{o}(l) &= 0\\ \mathsf{o}(I) &= 1\\ \mathsf{o}(A\otimes B) = \mathsf{o}(A \multimap B) &= \mathsf{o}(A)+\mathsf{o}(B)\\ \mathsf{o}(!A) &= 1+\mathsf{o}(A)\end{aligned}$$

DEFINITION 9.4.9 (VARIABLE ORDER)
Given a variable x and a relation D in which it occurs, define the *variable order* of x in D, which we will write $\mathsf{o}_D(x)$, as:

$$\mathsf{o}_D(x) = \begin{cases} 2\mathsf{o}(A)+1 & \text{if the variable } x \text{ is linear of type } Q \text{ in } D\\ 2\mathsf{o}(A) & \text{otherwise}\end{cases}$$

We now define the second complexity of a path in a relation, which is analogous to the first complexity except for the incorporation of the order of the types of wires. This amendment means that we need to be a little careful to avoid a circular definition. Call a path in a relation which cannot be extended by prepending a (c, x) pair to it an *initial* path.

DEFINITION 9.4.10 (COMPLEXITY II)
Given a path ϕ in a relation D, define the *second complexity* of it, written $\mathsf{Com}^2(\phi)$ inductively *over the sum of the lengths of the initial paths terminating at the last*

clause of the path ϕ:

$$\mathsf{Com}^2(c,\varepsilon) = 1$$

$$\mathsf{Com}^2(\vec{c}c',\vec{x}x') = \begin{cases} 2\mathsf{Com}^2(\vec{c},\vec{x}) & \text{if } c' \text{ is a replicator clause and } x' \text{ is resolved} \\ 2\mathsf{Com}^2(\vec{c},\vec{x}) + 4\mathsf{o}_D(x') & \text{if } c' \text{ is a replicator clause and } x' \text{ is unresolved} \\ \mathsf{Com}^2(\vec{c},\vec{x}) + M + 1 & \text{if } c' \text{ is an application clause and } x' \text{ is resolved} \\ \mathsf{Com}^2(\vec{c},\vec{x}) + M + 4\mathsf{o}_D(x') & \text{if } c' \text{ is an application clause and } x' \text{ is unresolved} \\ \mathsf{Com}^2(\vec{c},\vec{x}) + 1 & \text{if } c' \text{ is not a replicator or application clause} \\ & \text{and } x' \text{ is resolved} \\ \mathsf{Com}^2(\vec{c},\vec{x}) + 4\mathsf{o}_D(x') & \text{if } c' \text{ is not a replicator or application clause} \\ & \text{and } x' \text{ is unresolved} \end{cases}$$

where M is a number greater than the maximal complexity of an initial path terminating at a clause having immediately above the application clause.

This is a definition because at any point, the sum of lengths of initial paths terminating immediately above the application clause are smaller than the sum of lengths of initial paths terminating at the application clause, and so we can always determine M.

Definition 9.4.11 (Second Complexity Polynomial)
The second complexity polynomial for a relation D is given:

$$p^{2C}(D)(x) = \sum_{\phi \in \mathsf{TPath}(D)} x^{\mathsf{Com}^2(\phi)}$$

We now prove the crucial lemma, which shows that the second complexity polynomial is reduced by the relevant rewrites.

Lemma 9.4.12
If $D \to D'$ by a one-step $\Delta - 1,2,3,6$-rewrite, a one-step $\beta\eta$-rewrite or a one-step $F - \beta\eta$-rewrite then

$$p^{2C}(D)(x) > p^{2C}(D')(x)$$

Proof As before, we prove this by considering all the named reductions in turn.

$\Delta - 1,2,3$ These cases are easily seen to reduce the complexity of at least one path in exactly the same way as in the previous proof.

$\Delta - 6$ Again, this case goes through exactly as previously.

$F - \beta$ As before, this rewrite reduces the complexity of any path containing it.

$F - \eta$ Note that this rewrite can only occur on a variable of intuitionistic type Q. Further, there is at most one unresolved variable in the reduct. Assume that a general path ϕ contains the variable. Its complexity will be a function of the expression $4(o(Q)+1)$, and that of the path with the variable replaced by the reduct will be the same function of the expression $4(o(Q))+2$ or 2 depending on whether the type Q has the form $!A$ or not. In either of these cases the reduct path has strictly smaller complexity than the original.

$I - \eta$**,**$\otimes - \eta$**,** $\multimap - \eta$ **and** $! - \eta$ In each of these cases, any path containing the variable on which the η-rewrite takes place corresponds to several paths in the relation given by the rewrite, each of which has lesser complexity than the original by virtue of the fact that the only unresolved variables which may be present after the rewrite have lower variable order.

$I - \beta$**,** $\otimes - \beta$ **and** $! - \beta$ In each of these rewrites, a path is replaced straightforwardly by another path of lesser complexity.

$\multimap - \beta$ This case is the most subtle, and is the reason for the addition of the constant M to the definition of complexity. There are two possible restrictions of total paths to the redex. Consider the redex

$$\{(z = xy), (x = \lambda y'.z')\}$$

If we abbreviate the first of these clauses as c_1 and the second as c_2, the two restrictions of paths are $(c_2c_1, z'xz)$ and (c_1, yz). Now note that any total path having the second of these restrictions has complexity which is a function of the expression $(M+1)$ where M is a number greater than the maximal complexity of an initial path terminating at a clause immediately above c_1. Hence, when such a path is mapped to a total path in the relation given by the rewrite in which y' and y are identified, the complexity of such a total path is still less than that of the original path, as it is the same function of some number less than M.

Secondly, any total path restricting to $(c_2c_1, z'xz)$ has complexity which is a function of some number less than M. Any total path in the relation given by the rewrite which is generated from the first total path must also be an instance of a total path generated from a total path having the other restriction. Since we know that the complexities of such total paths are strictly less than those of the corresponding total paths of the kind mentioned above, we know that we have removed one path at the cost of adding others of lesser complexity, and so the second complexity polynomial is reduced. $\square$

As before, it is easy to see that both the $\Delta - 4$ and $\Delta - 5$-rewrites preserve the second complexity polynomial. Further, we again have that the nesting polynomial is reduced by the $\Delta - 5$-rewrite, that it is preserved by the $\Delta - 4$-rewrite, and that $\sum_{c \in \mathsf{Rep}(D)} \mathsf{Nest}(D)(c)$ is reduced by the $\Delta - 4$ rewrite.

Hence we again have the required lemma:

LEMMA 9.4.13 (NORMALISATION)
The transitive closure $\to_{\Delta F \beta\eta}$ of the union of the rewrites $\to_{\Delta}$, $\to_{F\beta\eta}$ and $\to_{\beta\eta}$ over the equality $=_{vm}$ is normalising.

Proof As before, consider the set of triples $(p(x), q(x), r)$ consisting of two polynomials and an integer, ordered by the lexicographic ordering inherited from the polynomial ordering and the integer ordering. The argument proceeds exactly as before using the measure

$$(p^{2C}(D)(x), p^{N}(D)(x), \sum_{c \in \mathsf{Rep}(D)} \mathsf{Nest}(D)(c))$$

and the lower bound $(0, 0, 0)$. □

This then gives us the required result.

THEOREM 9 (STRONG NORMALISATION)
If two relations D and E are equal in the transitive reflexive closure of the rewrite relation $\to_{\Delta F \beta\eta}$ and the equality $=_{vm}$, then any maximal sequence of one-step rewrites of either D or E will terminate after a finite number of steps in a common normal form up to $=_{vm}$.

This easily follows from confluence and termination of the rewrite.

Relations and Terms

Having given the rewrite on relations of $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$, we need to establish that it corresponds to the equality of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \emptyset)$, up to vestigial equality.

LEMMA 9.4.14
Given two $\Gamma; \Delta$-terms v and w of type A in $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \emptyset)$, $\Gamma; \Delta \vdash v = w : A$ iff $\rho_{\Gamma;\Delta}(v)$ and $\rho_{\Gamma;\Delta}(w)$ have the same normal forms up to vestigial equality.

We already know by the isomorphism of relational systems that the relations $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$ under the $=_{vm,\Delta,F\beta\eta}$-equality are isomorphic to the typing system $\mathsf{Lin}(\mathbb{H}, \mathsf{Onat}(\mathbb{H}, \otimes^{L}, I^{L}))$. Therefore, we only need to show firstly that the higher-order $\beta\eta$-equalities are soundly mapped to the closure of $\to_{\beta\eta}$ on $\mathsf{Rel}^{\mathsf{H}}(\mathbb{H})$, and that this rewrite is soundly mapped to higher-order $\beta\eta$-equality in the type theory. These proofs follow the standard pattern.

9.5 Corollaries

Firstly we present some general corollaries of our strong normalisation result.

Corollary 9.1 (Decidability)
Given two $\Gamma;\Delta$-terms v and w of type A in the type-theory $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H},\emptyset)$, we can decide whether or not they are provably equal.

Proof As before, simply reduce the relations corresponding to both terms to their normal forms and test whether these are equal by $=_{vm}$ decidably. $\square$

Corollary 9.2 (Conservativity (Syntactic))
For $\Gamma;\Delta$-terms v and w of $\mathsf{Lin}(\mathbb{O},\emptyset)$, if $\Gamma;\Delta \vdash v = w : A$ in $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O},\emptyset)$ then $\Gamma;\Delta \vdash v = w : A$ in $\mathsf{Lin}(\mathbb{O},\emptyset)$.

Proof Just as the terms v and w of $\mathsf{Lin}(\mathbb{O})$ are identically terms of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O})$, the relations $D = \rho_{\Gamma;\Delta}(v)$ and $D' = \rho_{\Gamma;\Delta}(w)$ are identically relations in both the system $\mathsf{Rel}(\mathbb{O})$ and the system $\mathsf{Rel}^{\mathsf{H}}(\mathbb{O})$. Moreover, it is easy to see by considering the form of the rules that if a higher-order relation contains no variables of higher-order types (and hence no higher-order operator clauses) no higher-order rewrite can apply to it. Hence, we have that no higher-order rewrite can be used in rewriting the two relations D and D' to their normal forms, and so the rewrites that are used can also be used on the relations in the system $\mathsf{Rel}(\mathbb{O})$. Hence the normal forms of the relations are the same in both systems, and therefore they are equal up to $=_{vm}$ in the higher-order system iff they are in the general system. This shows that if two relations are equal in the higher-order system they must also be equal in the general system. $\square$

We can now use our result to prove a fundamental result about the equality of $\mathsf{DILL}(\mathbb{C})$. This result was first proved directly using similar methods by the author early in 1996, and subsequently proved independently by Ghani [Gha96] using a term-based method.

Corollary 9.3 (Decidability of DILL($\mathbb{C}$))
$\mathsf{DILL}(\mathbb{C})$ is decidable.

This follows since $\mathsf{DILL}(\mathbb{C})$ is isomorphic by lemma 7.6.9 to the higher-order type-theory $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}^{\mathbb{C}},\emptyset)$ which is decidable by corollary 9.1.

Corollary 9.4 (Decidability of ILL($\mathbb{C}$))
$\mathsf{ILL}(\mathbb{C})$ is decidable.

This follows from the previous result by theorem 1.

In order to give our next corollary, we need a lemma.

LEMMA 9.5.1
There is a conservative map of higher-order type-theories:

$$\mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}}) \to \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \emptyset)$$

such that two terms in the source not involving the non-strict tensor are equal iff their images are equal in the target.

Proof We prove this by giving translations in both directions:

$$\begin{aligned}\iota_6 :& \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}}) \to \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \emptyset)\\ \iota_7 :& \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \emptyset) \to \mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}})\end{aligned}$$

such that $\Gamma; \Delta \vdash (\iota_7(\iota_6(v)) = v : A$ is a provable equality judgement for any $\Gamma; \Delta$-term v of type A in $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}})$ which does not involve the non-strict tensor.

We now define the maps. Firstly, ι_6 is the identity except on the tensor constructs, where it maps the strict tensor types and terms to the corresponding left-bracketed forms of the non-strict tensor.

Conversely, ι_7 is the identity except on the (non-strict) tensor constructs, where it maps these onto the strict tensor constructs.

It is easily seen that both these maps are sound, and that $\iota_6 \circ \iota_7$ is the identity up to provable equality on terms not containing non-strict tensors of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}})$. The result now follows since firstly if two such terms are equal then their images under ι_6 are equal by soundness, and if their images under ι_6 are equal, it follows that their images under $\iota_6 \circ \iota_7$ are equal, from which we can deduce that they are equal. $\square$

We can now prove that the equational theories of all the action calculi we have discussed are decidable. Similar results were previously obtained for the cases $\mathsf{AC}(\mathbb{K})$ and $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$ by Milner [Mil93b], using a normal form syntax.

COROLLARY 9.5 (DECIDABILITY OF AC($\mathbb{K}$))
The theory of $\mathsf{AC}(\mathbb{K})$ is decidable, and so are those of $\mathsf{AC}_{\boxtimes}(\mathbb{K})$, $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$ and $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$.

First note that we have conservative maps from $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$, $\mathsf{Lin}^{\mathsf{A}^{\boxtimes}}(\mathbb{K})$, $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$ and $\mathsf{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K})$ to $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}})$, so that in order to decide an equality of any one of these type theories it suffices to decide the image of the equality in $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \mathbb{A}^{\mathsf{A}})$, which can be done. But we can see by the form of the map ι_5

that no term in the image of it involves the non-strict tensors. Hence any two such terms are equal by lemma 9.5 iff their images are equal in $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O}^{\mathsf{A}}, \emptyset)$, but this is decidable by corollary 9.1. Hence the result follows easily by the isomorphisms between the action calculi $\mathsf{AC}(\mathbb{K})$, $\mathsf{AC}_{\boxtimes}(\mathbb{K})$, $\mathsf{AC}_{\Rightarrow}(\mathbb{K})$ and $\mathsf{AC}_{\boxtimes,\multimap}(\mathbb{K})$ and their respective type-theories.

It is interesting to note that it is straightforward to give relations for $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$, and to prove that they have normal forms under rewriting, up to vestigial equality. These are then normal forms for the action calculus by virtue of lemma 6.3.8.

However, we also have directly normal forms for the intuitionistic action calculus, the molecular forms, due to Milner [Mil93b]. Hence it must be the case that these two normal forms are ismorphic. We give a brief intuitive outline of the translation from relations to molecular forms.

Relations map to molecular forms in which each operator clause goes to a molecule, and the input variables of each clause become the binding inputs of the molecule, and the outputs become the binding outputs of the molecule. Clauses other than the operator clauses, which is to say tensor clauses, copy and discard clauses and F-clauses are mapped to substitutions, as in the intuitionistic action calculus every arity is intuitionistic prime.

In the reverse direction each molecular form is mapped to a set by taking each molecule to an operator clause in the set, with inputs and outputs connected to F-clauses and tensor clauses as appropriate in order to give the correct arity.

Chapter 10

Conclusions

We now conclude by summarising our results, discussing the implications of the work presented in this thesis, and then considering directions for further work, in progress and longer-term. We argue that our generalised linear type-theory is a useful general tool for the study of systems of typed computation. We showed that it is expressive enough to account for a wide range of examples, and we proved a substantial number of general results, applying to each instance of the type-theory.

10.1 Results

Although we started this thesis by introducing DILL as an alternative formulation of linear logic, in retrospect the main thread of the thesis is more clearly seen based on the generalised linear type theory we introduced, and its semantics and properties. In this view, the starting point is the following triangle:

$$\begin{tikzcd}
& \text{Logic } \mathsf{Lin}(\mathbb{O}) \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{Type-Theory } \mathsf{Lin}(\mathbb{O}, \mathbb{A}) \arrow[rr, leftrightarrow] & & \mathsf{Lin}(\mathbb{O}, \mathbb{A})\text{-Models}
\end{tikzcd}$$

In chapters 4 and 5 we gave the vertices of this diagram, and made precise their connections. Further, we showed in chapters 8 and 9 that the equality of the type-theory $\mathsf{Lin}(\mathbb{O}, \emptyset)$ is decidable. This equality is analogous to the equivalence of proofs naturally generated by some simple permutations, and perhaps more significantly, it is analogous to the equality of arrows in the initial $\mathsf{Lin}(\mathbb{O})$-interpretation. The coherence problem for $\mathsf{Lin}(\mathbb{O})$-interpretations is the problem of determining which diagrams of arrows built up from those present in every $\mathsf{Lin}(\mathbb{O})$-interpretation commute in all $\mathsf{Lin}(\mathbb{O})$-interpretations. We solved this problem using our term $\mathsf{Lin}(\mathbb{O})$-interpretation by showing that any diagram of such arrows commutes in every $\mathsf{Lin}(\mathbb{O})$-interpretation precisely when it commutes in the term $\mathsf{Lin}(\mathbb{O})$-interpretation, which by definition is when the appropriate term equality is provable. Now this is decidable, and so we have an effective procedure for determining when a diagram of arrows commutes in all $\mathsf{Lin}(\mathbb{O})$-interpretations.

We also considered an important sub-case of the triangle stated above in chapter 7, as follows:

$$\begin{tikzcd}
 & \text{Logic } \mathsf{Lin}^{\mathsf{H}}(\mathbb{H}) \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{Type-Theory } \mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A}) \arrow[rr, leftrightarrow] & & \mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A})\text{-models}
\end{tikzcd}$$

This is the higher-order subcase, in which the logic is equipped with an exponential and an arrow type, the type-theory incorporates a linear λ-calculus and the models also have exponential and closed structure. There is then a very natural embedding of any linear type-theory into a higher-order linear type-theory built over it, and we showed using a semantic construction that this is a conservative extension.

We then extended the techniques used to decide the equality of $\mathsf{Lin}(\mathbb{O}, \emptyset)$ in chapter 9 to decide the equality of $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \emptyset)$. This corresponds to deciding the coherence problem for the higher-order $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H})$-interpretations, which is significant, since for example it includes coherence for models of multiplicative exponential linear logic. This same extension also enables us to show syntactically that $\mathsf{Lin}^{\mathsf{H}}(\mathbb{O}, \emptyset)$ conservatively extends $\mathsf{Lin}(\mathbb{O}, \emptyset)$.

In addition to presenting this well-developed theory of linear operators in the case of the logic, the type theory and the models, we considered some examples. Firstly, we showed that DILL, its type-theory and models arise as instances of higher-order generalised linear logics, type-theories and models. We then have as an easy corollary of our decidability results that the equality of $\mathsf{DILL}(\mathbb{C})$ is decidable.

Secondly, and more surprisingly, we showed that Milner's action calculi and their models correspond to a particular range of instances of the generalised linear type-theories and models, as do their higher-order extensions. Again, our decidability results enabled us to show that all of these action calculi are decidable, and further we showed that the embedding of an action calculus into its higher-order extension is conservative, via either our semantic or our syntactic proofs.

10.2 Extending $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$

Given the results we already proved in this thesis, we go on to consider some desirable extensions to the framework, with their advantages and disadvantages. The first two of these consist of extensions to the syntax and semantics of the framework, to incorporate important computational ideas which we have so far overlooked. The last three follow the program of chapter 7, where we pick out a particular set of types, operators and equalities as a canonical expression of a certain computational concept. In chapter 7, the computational concept is higher-order behaviour, and in this section we will consider choice, recursion and the negation of linear logic.

Rewriting

In contrast to our general linear type theories, Milner's action calculi [Mil96] are enriched with a notion of *dynamics*, which consists of a rewrite over the equality satisfying certain simple conditions. In the representation of process calculi, the dynamics is crucial, since as the name suggests, it captures the operational behaviour of the process calculi, whereas the equality represents a structural congruence on processes. It therefore seems an obvious idea to add a *rewriting judgement* to our general theory, possibly of the form $\Gamma; \Delta \vdash v \to w : A$. Such a rewrite would be closed under the equality, and under certain contexts, although in action calculi dynamics are not closed under controls. It might further be subject to some conditions, for example perhaps sufficient to prove the property:

$$\frac{\Gamma; \Delta \vdash v \to u : A}{\Gamma; \Delta \vdash v = u : A}$$

for values v. Another possibility would be to add a labelled rewrite, although we shall not consider this further here.

We might then follow the example of Power [Pow96] and consider the semantics of such a rewrite judgement, modelling it by an enrichment of the SMC part of the semantics with a preorder. This preorder should then be subject to some conditions analogous to those on the rewrite, for example possibly that morphisms in the image of the functor F are maximal in the preorder.

Such an extension opens many possible lines of enquiry. Obviously results and techniques of general term rewriting theory become immediately relevant, and may well be adaptable to this framework, and similarly it makes possible the study of process calculi dynamics, (even, perhaps, including bisimulation), in a type-theoretic setting. Another interesting possibility, which we will briefly sketch, is to relate the equality and the rewrite and study the interplay between them. Given an extension of our type-theory with a rewrite judgement, firstly define a morphism of type-theories in the obvious way based on maps on types and operators, preserving the rewrite and the equality. Then say that one instance of our type-theory *implements* another if $\Gamma; \Delta \vdash v = v' : A$ in the first implies that there exists u in the second such that $\Gamma; \Delta \vdash v \to u : A$ and $\Gamma; \Delta \vdash v' \to u : A$ in the second, and there is a morphism of type-theories from the second to the first.

Of course, this definition is most useful when there is a sub-rewrite of the rewrite in the first type-theory which has the Church-Rosser property. For example, an instance of our type theory with the types and terms of $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$, but having the β and η-rewrites rather than equalities would be an implementation of the version presented with equalities. One might then be able to develop this idea to relate operational behaviours and equality relations in the same framework.

Structure on Types

We might also extend our framework by adding more structure on types. Whereas we have a complicated structure allowing us general operators on terms, the only structure present in the type set P_L is the distinction between members of P_I and non-members of P_I. However, in many cases there is obvious structure implicit in the choice of type set, for example in the case of higher-order instances of our theory. The type set in this case is freely closed under the binary operations $\multimap$, $\otimes$, the unary operator $!$, and the nullary operation I. Hence, we might specify that the type set be constructed over primitive types using *type operators*, which would then have arities of the general form

$$\left(\prod_{i=1\ldots r} \mathsf{P}_I\right) \times \left(\prod_{j=1\ldots s} \mathsf{P}_L\right) \to \mathsf{P}_L$$

This should be read as saying that the type operator would take r intuitionistic arguments and s linear ones, and return an arity. For example, the obvious type operator $\otimes$ would then have an arity $\mathsf{P}_L \times \mathsf{P}_L \to \mathsf{P}_L$. *Intuitionistic type operators* would then be defined similarly returning arities in P_I.

Given this structure on types, we could refine our specification of operators. For example, the tensor introduction operator, which for every arities A and B has an instance with the arity

$$\frac{;\quad ()A \quad ()B}{()A \otimes B}$$

would be specified as a single operator of arity

$$\frac{;\quad ()_{-} \quad ()_{=}}{()_{-}\otimes_{=}}$$

This would be interpreted as meaning that for every two arities A and B, there would be an instance taking pairs of terms of type A and terms of type B, and returning a term of type the tensor type operator applied to A and B.

This would also have a significant effect on the semantics. Clearly, type operators would be interpreted as a first step as functions on the objects of the category in question. It would then be natural to insist that in the case of the tensor, rather than having an appropriate natural transformation for each operator instance over arities A and B, we should have one for each two objects of the category. This change would substantially simplify the semantics of our examples, for example by forcing the interpretation of the tensor operator to be a tensor functor in all models of the appropriate equational theory.

Choice Constructs

Choice constructs are crucial to programming languages. However, although it is possible to represent a coproduct construction in our general type-theory using families of operators:

$$\frac{;}{(A)B \oplus A}\ \mathsf{inr}_{A,B} \qquad \frac{;}{(A)A \oplus B}\ \mathsf{inl}_{A,B} \qquad \frac{(CA)C' \quad (CB)C';}{(CA \oplus B)C'}\ \mathsf{cases}_{A,B,C,C'}$$

with appropriate equalities, this does not capture certain naturality properties of the cases operator. Consider the bound assumptions C in the above instance of the operator. The operator is natural in the assumptions C in the sense that a cut into one of these assumptions should be proof-equivalent to two cuts, one into each premiss of the operator instance.

Since the naturality we have incorporated into our type-theory is inherently a multiplicative naturality, based on the implicit tensor of the context on the left, it is insufficient to represent this additive naturality. We could amend this by adding a class of *additive* operators, where we rename the existing operators *multiplicative* operators. These additive operators would have the defining property that an operator of arity for example:

$$\frac{;(A)A' \quad (B)B'}{(C)C'}$$

would have the logical rule

$$\frac{\Gamma;\Delta, A \vdash A' \quad \Gamma;\Delta, B \vdash B'}{\Gamma;\Delta, C \vdash C'}$$

in which the contexts are combined additively rather than multiplicatively. Naturality would then be defined with respect to this additive context discipline, and the cases operator with its naturality would be satisfactorily captured.

However, it is not clear how this development might be extended to the semantics. It seems likely that the cases operator given above would be soundly modelled by a coproduct in our models, given suitable distributivity conditions, exactly as the additive connective $+$ in intuitionistic linear logic is modelled by a coproduct [BBdPH93a]. However, it is not clear what would constitute a sound model of a type-theory with arbitrary additive operators. Further, we can already see that equipping such a type-theory with normal forms based on proof-nets is a difficult problem, as the proof-net technology handling the additives [Gir96] is substantially more complicated than that for the multiplicative-exponential fragment, and in general, the proper treatment of the additives is a key point in linear logic [Gir94]. Although there are presentations of proof-nets for linear logic including the additives [Gir96], they all increase the complexity of the proof-net representation substantially, and any analogous extension of relations would cause the same complication.

In view of this state of affairs, finding the best way of adding choice constructs to our framework seems to be one of the most outstanding existing problems.

Recursion

Another essential component of serious programming languages is some form of iterative or recursive construct. We may define in our framework the obvious recursion operators with arity

$$\frac{;(A)A}{()A}$$

for each arity A, with the expected equalities. We would then expect this to correspond to adding a trace operator on the symmetric monoidal part of the models, in view of Hasegawa's work [Has97]. Further, also following work of Hasegawa, such a recursion operator might well be modelled primitively in the proof-nets simply by allowing nets to be cyclic, with certain extra primitive equivalences. It is not then clear how the extended proof-net framework might be equipped with a confluent rewrite, and this problem is a substantial one.

We should also note that there are other approaches to adding recursion to our models, notably by asking that the cartesian category have a fixpoint operator, as in work of Fiore and Plotkin [Plo93b, FP94]. Since the models of our type-theories interpret operators as natural transformations on the SM part of the model, this semantic notion is not captured precisely by any instance of our general type-theory. It remains ongoing and interesting work to relate these two possible types of models for recursion.

Negation

Given that the linear type-theory corresponding to the multiplicative fragment of linear logic has been shown to have a direct relation to the higher-order action calculus, it is worth considering other connectives of linear logic, and their incorporation in general linear type-theories. We have already mentioned the choice constructions associated with the coproduct of linear logic, and clearly this is essential to practical programming languages, parallel or not. The one remaining connective is the classical negation. For as long as linear logic has existed, there have been connections postulated between it and concurrency, for example [Gir87, Abr93] and many others.

Further, there has been much work investigating the computational significance of the boundary between classical logic and intuitionistic logic in both conventional and linear situations [Par92, Bie96b, Bie96a, Ong96].

Using our framework, we can incorporate some classical behaviour, motivated by the semantics of classical linear logic. Just as intuitionistic multiplicative linear logic (without the exponentials) has as categorical models symmetric monoidal closed categories, *-autonomous categories, which are symmetric monoidal closed categories with a dualising object (intuitively modelling the nullary $\perp$) are thought to be models of classical linear logic. Moreover, it has been shown in [CS97] that *-autonomous categories are equivalent to symmetric monoidal closed categories having a unary negation satisfying certain simple equalities.

Hence, we might add a unary classical negation operator (written $(\_)^{\perp}$) to any general linear type theory by first closing the arities under the unary operator $(\_)^{\perp}$ and secondly by adding the two constants having arity:

$$(A^{\perp\perp})A \qquad (A)A^{\perp\perp}$$

and suitable equalities, based on the categorical semantics. A first question is what relationship the instance of our general type-theory corresponding to $\mathsf{DILL}(\mathbb{C})$, augmented with this negation, bears to classical linear logic. Secondly, we can now consider the extension of the system corresponding to higher-order action calculi with this negation, and in particular its relationship to type-theories for process algebra.

Investigating these two questions, we will obtain a better picture of the relationship between classical linear logic and concurrency not only on the syntactic level but also via the inherited semantics of our type-theories.

10.3 Aims and Objectives

Finally, we consider general questions and directions for study which we feel are raised by the work in this thesis.

Frameworks

Some might say (and indeed have!) that there is a needless proliferation of syntaxes in general, and of those which are variants of linear λ-calculus in particular. Moreover, there are many possible syntaxes for the same underlying semantic situation.

We hope that our general linear type-theory may provide a means of correlating these syntaxes and giving them a uniform presentation. By providing a tight link between a range of interesting variants of the linear semantics and a range of our type-theories, we have made it easier to reconstruct syntaxes for particular semantic situations of this kind, and reason about their properties. In addition to this uniformity of syntax and semantics, a general framework allows us to consider a wider picture of syntaxes and translations between them, in an organised setting.

Turning to the corresponding semantics, this organised setting makes it natural to study the translations between syntaxes as morphisms between their categories of models, and characterise interpretations as initiality morphisms, amongst other things. This perspective leads to interesting insights, including the Yoneda argument of chapter 7.

On the other hand, we can see in this thesis that there is a place for diversity of syntax; certainly the action calculus and our linear type-theory for it, whilst having essentially the same semantics, each have salient features which justify their use for particular applications. It is interesting to note that although historically their connection was discovered via the syntax, the semantics provided by far the clearest indication of the connection once it was clearly established in both cases.

We feel that the way forward given these points must be towards general frameworks for syntax, with solid semantic foundations. Such constructions will limit the proliferation of syntax, if adopted, and important variant syntaxes can be presented and given semantics via translation to the appropriate instance of the framework. In order to assist in this, we aim to extend our given framework with canonical choice and recursion operators, and to consider other common computational connectives to see if they have a canonical presentation in this setting.

Proof Nets and Action Graphs

A slightly novel part of our presentation is the work on proof-nets, used to investigate the properties of various equalities. Although this work is not familiar in the context of type-theories, it is very significant that Milner [Mil96] developed a theory of *action graphs*. Furthermore, Milner's molecular forms [Mil93b] are related to our relational normal forms. We know from Milner's work [Mil93b] that two actions are equal in the theory of action calculi if and only if their molecular forms are equal. We also know that two terms of the corresponding instance of our type-theory are equal if and only if their relational normal forms are equal under the vestigial equality. Hence, it must be the case that molecular forms and relational normal forms are in one-to-one correspondence up to the relevant equalities.

This result shows that there exists a close correspondence between these two systems, but provides very little information about the intuition behind the correspondence. We aim to investigate this correspondence, and the relative advantages of the two syntaxes. More generally, we need to investigate the advantages of graphical formulations over more traditional approaches for investigating decidability.

Process Algebras and Type-Theories

On a more detailed level, the connection between action calculus and our general linear type-theory raises many questions. Particularly interesting are those which are created by the presentation of a familiar syntax in an unfamiliar setting. Consider for example the π-calculus as an action calculus, and therefore as a type-theory (given the rewriting extension proposed earlier in this chapter). This is a very unfamiliar presentation of a process algebra, which immediately highlights certain points. One such is that the π-calculus in its original form has very little typing structure compared to its term structure, in contrast with the normal pattern in type-theories. It might be more natural to also consider typed variants such as that proposed for example in [PRT93], and their relationship to the original. It might also be worth studying the form of the input and output controls in the type-theory, and comparing them with more familiar logical and type-theoretic rules. These and other considerations would have an immediate impact on process algebra theory.

In the reverse direction, we have already seen that the first-class rewriting of process algebras can profitably be incorporated into our type-theory. Such rewrites are often very different from those normally encountered in type-theory, as they are rarely confluent or normalising. However, their behaviour is the key to the theory of process algebras.

In the same way, we might expect other cross-fertilisation between these two areas due to their common semantic foundation.

Linear Logic and Process Algebras

As has been previously mentioned in this thesis, broad connections have been suggested between linear logic and process algebras. More particularly, process calculi directly built on the proof structure of classical linear logic have been given [Abr91, Abr94].

We aim to investigate this further, firstly by giving a set of instances of our framework incorporating the classical negation operator of linear logic, and then by considering the relationship between these and instances corresponding to process calculi. There are obvious disparities between the proofs of linear logic and conventional process calculi, notably the typing of linear logic contrasted with the less detailed typing of process calculi, and the dynamics of process calculi contrasted with the essentially static proof equivalence of linear logic.

However, we aim to construct a range of intermediate instances between classical linear logic and various process algebras, and thereby discover whether classical linear logic underlies them in the sense of being soundly embeddable into them.

If this is concluded positively, then it may become possible to consider a canonical foundation for process algebras and use it to classify some of them.

10.4 Final Remarks

In this thesis, we have focussed on linearity. The wide range of applications of our general framework, together with the examples in other areas of computer science, provide evidence that computation is inherently linear with an underlying intuitionistic calculus of values. Although the theory of intuitionistic logics, type-theories and their semantics is very well developed in comparison to that of our framework and similar constructions, it seems obvious from this work that once the theory of linear-non-linear situations is equally developed, it will provide a clearer foundation for studying a wide range of computational situations, from resource-sensitive though imperative to concurrent computation.

Appendix A

Basic Definitions

A.1 Syntactic Preliminaries

We need to present certain definitions which will be used throughout the thesis. For the purposes of this appendix, we let Ob be a set of objects of some sort, ranged over by $o \ldots$.

Sequences

We will often need to discuss sequences of objects of various types, and will need to use functions on such sequences. Consequently, we give some notation:

Notation A.1.1 (Sequences)
Given objects $o \ldots$ of some kind, we will write $\vec{o}$ to denote an arbitrary sequence of such objects. Where it is necessary to make explicit the elements of sequence, we will write it as $o_1 \ldots o_r$. We will generally write concatenation as juxtaposition, and ϵ for the empty sequence. Further, we will write the common operation of 'cons-ing' an element to the head of the sequence using a comma, so that for example the expression $o', \vec{O}$ represents the sequence having as first element o' and as its $i+1$th element the ith element of $\vec{O}$, for $i = 1 \ldots r$ where r is the length of the sequence $\vec{O}$.

However, we will need to make an exception to this syntax in order to accommodate common practice in logics and type-theories; when we give sequences of formulae of a logic or a type of a type theory, $A_1 \ldots A_r$, we will write comma for concatenation, and $\_$ for the empty sequence. We will also use this convention for typings $x{:}A$. In this framework we will elide the distinction between a one-element sequence and the single element it contains, so that $A, \vec{A}$ can be read either as the 'cons' of A onto $\vec{A}$ or the concatenation of the singleton sequence A and the sequence $\vec{A}$.

Next we give some notation for applying functions to the elements of sequences. Given a function $f : Ob \to Ob'$, we will write $f\vec{o}$ for the function which takes sequences of objects in Ob to sequences of objects in Ob' simply by using the function f on each element of the sequence.

In the presentation of typing rules that we give, we need to say when one sequence is a merge of two others:

Definition A.1.2 (Merge Relation)
For sequences of objects $\vec{o}_1$ and $\vec{o}_2$, define the *merge relation* $\vec{o} = \vec{o}_1 \# \vec{o}_2$ inductively as follows:

- $\vec{o} = \_\#\vec{o}$ holds, as does $\vec{o} = \vec{o}\#\_$.
- $o', \vec{o} = o', \vec{o}_1 \# \vec{o}_2$ holds if $\vec{o} = \vec{o}_1 \# \vec{o}_2$ does.
- $o', \vec{o} = \vec{o}_1 \# o', \vec{o}_2$ holds if $\vec{o} = \vec{o}_1 \# \vec{o}_2$ does.

Typings and Typing Contexts

We define several typing systems in this thesis, and a common component of each of them is a notion of typing and typing context. Assume that we have a set of variables X ranged over by $x, y, z \ldots$ and a set of types Ty ranged over for the purposes of this section by $T \ldots$.

Now, a *typing* is a pair of a variable and a type, which we write $x{:}T$. We will let sequences of typings be ranged over by $\Gamma, \Delta \ldots$. Where convenient, we will write the sequence of typings $x_1{:}T_1 \ldots x_r{:}T_r$ as $\vec{x}{:}\vec{T}$, where $\vec{x}$ and $\vec{T}$ are the obvious sequences of variables and types respectively. For a sequence of typings $\Gamma = \vec{x}{:}\vec{T}$, we will then say that $|\Gamma|$ is the set of elements in the sequence $\vec{T}$ and further that $\mathsf{dom}(\Gamma)$ is the set of elements in the sequence $\vec{x}$.

Now, a *dual typing context*, which we will often refer to just as a typing context since most of our typing contexts are dual, is a pair of sequences of typings, written $\Gamma; \Delta$. The *intuitionistic* part of the context is Γ and the *linear* part is Δ.

Conversely, a *single typing context* is just a sequence of typings.

A.2 Categorical Definitions

In general, we will refer to categories using the symbols $\mathcal{C}$ and $\mathcal{S}$, and will use $\mathsf{obj}(\mathcal{C})$ to refer to the objects of a category $\mathcal{C}$, ranged over by $X, Y \ldots$. Also, we will use $\mathcal{C}(X, Y)$ to refer to the set of arrows between X and Y in the category $\mathcal{C}$ We will let such arrows be ranged over by $f, g \ldots$, and write $f : X \to Y$ to indicate that f is in $\mathcal{C}(X,Y)$ where the category is clear from the context. We will write the identity arrow at object X as id_X and write the composition of f and g as $f;g$. We will let $\mathsf{v},\mathsf{w}\ldots$ range over natural transformations, and write $\mathsf{v}: F \to G$ to indicate that v is a natural transformation from F to G.

The following definitions can be found in [Mac71].

Definition A.2.1 (Symmetric Monoidal Category)
A Symmetric Monoidal Category (SMC) is a category $\mathcal{C}$ with a bifunctor $\otimes : \mathcal{C}\times\mathcal{C}\to\mathcal{C}$ and natural isomorphisms

$$\begin{array}{rccc}
\mathsf{a}_{X_1,X_2,X_3}: & (X_1\otimes X_2)\otimes X_3 & \to & X_1\otimes(X_2\otimes X_3)\\
\mathsf{ri}_X: & I\otimes X & \to & X\\
\mathsf{li}_X: & X\otimes I & \to & X\\
\mathsf{s}_{X,Y}: & X\otimes Y & \to & Y\otimes X
\end{array}$$

s.t.

$$(\mathsf{a}_{X,Y,X'}\otimes \mathsf{id}_{Y'});\mathsf{a}_{X,(Y\otimes X'),Y'};(\mathsf{id}_X\otimes \mathsf{a}_{Y,X',Y'}) = \mathsf{a}_{(X\otimes Y),X',Y'};\mathsf{a}_{X,Y,(X'\otimes Y')} \tag{A.1}$$

$$\mathsf{a}_{X,I,Y};(\mathsf{id}_X\otimes \mathsf{li}_Y) = (\mathsf{ri}_X\otimes \mathsf{id}_Y) \tag{A.2}$$

$$\mathsf{li}_I = \mathsf{ri}_I \tag{A.3}$$

$$(\mathsf{s}_{X,Y}\otimes \mathsf{id}_{X'});\mathsf{a}_{Y,X,X'};(\mathsf{id}_Y\otimes \mathsf{s}_{X,X'}) = \mathsf{a}_{X,Y,X'};\mathsf{s}_{X,(Y\otimes X')};\mathsf{a}_{Y,X',X} \tag{A.4}$$

$$\mathsf{s}^{-1}_{X,Y} = \mathsf{s}_{Y,X} \tag{A.5}$$

$$\mathsf{s}_{I,X};\mathsf{li}_X = \mathsf{ri}_X \tag{A.6}$$

A symmetric monoidal category is *strict* if the natural transformations li,ri and a are all the respective identities. This implies that the tensor and unit are strictly associative. Note that the definition does not insist that the symmetry be strict.

Definition A.2.2 (Symmetric Monoidal Closed Category)
A (strict) symmetric monoidal closed category (SMCC) is a (strict) SMC s.t. the bifunctor $\otimes$ has a right adjoint $\multimap$, or

$$\mathcal{C}(X\otimes Y, X') \simeq \mathcal{C}(X, Y\multimap X')$$

where $\simeq$ denotes an isomorphism natural in X, Y and X'. We write the closed structure as:

$$\begin{aligned}
\mathsf{ap}_{X,Y} &: (X\multimap Y)\otimes X \to Y\\
\lambda_Y(f) &: X \to (Y\multimap X') \text{ where } f: X\otimes Y \multimap X'
\end{aligned}$$

For convenience, we will abbreviate the unit of the adjunction:

$$\lambda_Y(\mathsf{id}_{X\otimes Y}) = \mathsf{pa}_{X,Y} : X \to (Y\multimap(X\otimes Y))$$

Definition A.2.3 (Cartesian Closed Category)
A *cartesian (closed) category* is a symmetric monoidal (closed) category in which the tensor product is cartesian. We shall typically use the symbol $\times$ to represent the product of a cartesian category, and use CC (CCC) to denote a cartesian (closed) category. Further, we will write the arrows giving the cartesian structure as follows:

$$\begin{aligned} \mathsf{d}_X :& X \to 1 \\ \mathsf{p}_i :& X_1 \times X_2 \to X_i \\ \mathsf{c}_X :& X \to X \times X \\ \langle f, g \rangle :& X \to Y_1 \times Y_2 \text{ where } f : X \to Y_1 \text{ and } g : X \to Y_2 \end{aligned}$$

If the cartesian (closed) category is strict, we can use the generic projections $\mathsf{p}_{i,r} : X_1 \times \ldots \times X_r \to X_i$.

We now need to define the concept of functor between two SMCCs.

Definition A.2.4 (Symmetric Monoidal Functor)
A symmetric monoidal functor, abbreviated to SM functor,

$$(F, \mathsf{m}_{X,Y}, \mathsf{mi}) : (\mathcal{C}, \otimes, I, \mathsf{a}, \mathsf{li}, \mathsf{ri}, \mathsf{s}) \to (\mathcal{C}', \otimes', I', \mathsf{a}', \mathsf{li}', \mathsf{ri}', \mathsf{s}')$$

is a functor $F : \mathcal{C} \to \mathcal{C}'$ with a map $\mathsf{mi} : I' \to F(I)$ and a natural transformation $m_{X,Y} : F(X) \otimes' F(Y) \to F(X \otimes Y)$ s.t.

$$\mathsf{a}'_{FX,FY,FX'}; (\mathsf{id}_{FA} \otimes' \mathsf{m}_{Y,X'}); \mathsf{m}_{X,Y\otimes X'} = (\mathsf{m}_{X,Y} \otimes' \mathsf{id}_{FX'}); \mathsf{m}_{X\otimes Y,X'}; F(\mathsf{a}_{X,Y,X'}) \tag{A.7}$$

$$\mathsf{ri}'_{FX} = (\mathsf{mi} \otimes \mathsf{id}_{FX}); \mathsf{m}_{I,X}; F(\mathsf{ri}_X) \tag{A.8}$$

$$\mathsf{s}'_{FX,FY}; \mathsf{m}_{Y,X} = \mathsf{m}_{X,Y}; F(\mathsf{s}_{X,Y}) \tag{A.9}$$

A SM functor is *strong* if mi is an isomorphism and $\mathsf{m}_{X,Y}$ is a natural isomorphism. It is *strict* when mi is the identity and $\mathsf{m}_{X,Y}$ is the identity transformation.

Under this definition it is easy to check that given two (strict, strong) SM functors $(F, \mathsf{m}_{X,Y}, \mathsf{mi})$ and $(G, \mathsf{m}'_{X,Y}, \mathsf{mi}')$ their compose is (strict, strong) symmetric monoidal when equipped with maps:

$$(GF, (\mathsf{m}'_{FX,FY}; G(\mathsf{m}_{X,Y})), (\mathsf{mi}'; G(\mathsf{mi})))$$

Remark A.2.5
Given a SM functor $F : \mathcal{C} \to \mathcal{C}'$, notice that there is an induced natural transformation

$$\mathsf{k}_{X,Y} = (FX \multimap (F\mathsf{ap}_{X,Y}; \mathsf{m}_{X\multimap Y,X})) \circ \mathsf{pa}'_{FX,F(X\multimap Y)} : F(X \multimap Y) \to (FX \multimap FY)$$

Definition A.2.6 (Symmetric Monoidal Closed Functor)
A symmetric monoidal closed functor, abbreviated SMC functor, is a SM functor. A strong SMC functor is a strong SM functor for which $\mathsf{k}_{X,Y}$ is a natural isomorphism, and a strict SMC functor is a strict SM functor for which $\mathsf{k}_{X,Y}$ is the identity transformation.

Definition A.2.7 (Cartesian (Closed) Functor)
A *cartesian (closed) functor* between two cartesian (closed) categories is a SM (SMC) functor which preserves the cartesian structure up to isomorphism. It is *strict* when it preserves the cartesian structure up to equality.

Definition A.2.8 (Monoidal Natural Transformation)
A monoidal natural transformation from one symmetric monoidal functor $(F, \mathsf{m}_{X,Y}, \mathsf{mi}) : \mathcal{C} \to \mathcal{C}'$ to another $(G, \mathsf{m}'_{X,Y}, \mathsf{mi}') : \mathcal{C} \to \mathcal{C}'$ is a natural transformation $\mathsf{v} : F \to G$ s.t.

$$\boxed{\begin{aligned} \mathsf{m}_{X,Y}; \mathsf{v}_{X\otimes Y} &= (\mathsf{v}_X \otimes' \mathsf{v}_Y); \mathsf{m}'_{X,Y} \qquad \text{(A.10)} \\ \mathsf{mi}; \mathsf{v}_I &= \mathsf{mi}' \qquad \text{(A.11)} \end{aligned}}$$

Bibliography

[Abr91] Samson Abramsky. Proofs as processes. Unpublished Lecture Notes, 1991.

[Abr93] Samson Abramsky. Computational interpretations of linear logic. *Journal of Theoretical Computer Science*, vol. 111, pp. 3–57, 1993. Earlier version appeared as Imperial College Technical Report DOC 90/20, October 1990.

[Abr94] Samson Abramsky. Proofs as processes. *Theoretical Computer Science*, vol. 135, pp. 5–9, 1994.

[Acz78] Peter Aczel. A general church-rosser theorem. Unpublished manuscript, 1978.

[Acz80] Peter Aczel. Frege structures and the notions of proposition, truth and set. In J. Barwise, H. J. Keisler, and K. Kunen, editors, *The Kleene Symposium*, pp. 31–59. North-Holland Publishing Company, 1980.

[Avr94] A. Avron. What is a logical system? In D. Gabbay, editor, *What is a logical system?*, no. 4 in Studies in Logic and Computation, chapter 8, pp. 217–238. Oxford Science Publications, 1994.

[Bar] A. Barber. Proof nets for dual intuitionistic linear logic. Unpublished note.

[Bar79] Michael Barr. *$*$-Autonomous Categories*, vol. 752 of *Lecture Notes in Mathematics*. Springer-Verlag, 1979.

[Bar91] Michael Barr. $\star$-autonomous categories and linear logic. *Mathematical Structures in Computer Science*, vol. 1 no. 2, pp. 159–178, July 1991.

[Bar96] Andrew Barber. DILL- Dual Intuitionistic Linear Logic. Technical Report LFCS-96-347, LFCS, Edinburgh University, 1996.

[BBdPH93a] N. Benton, G. Bierman, V. de Paiva, and J. M. E. Hyland. Linear lambda-calculus and categorical models revisited. In *Computer Science Logic '92, Selected Papers*, vol. 702 of *Lecture Notes in Computer Science*, pp. 61–84. Springer, 1993.

[BBdPH93b] N. Benton, G. Bierman, V. de Paiva, and J. M. E. Hyland. A term calculus for intuitionistic linear logic. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, vol. 664 of *Lecture Notes in Computer Science*, pp. 75–90. Springer-Verlang, 1993.

[BCS95] R. F. Blute, J. R. B. Cockett, and R. A. G. Seely. Categories for computation in context and unified logic: I. electronically available, 1995.

[BCST96] Richard Blute, J. R. B. Cockett, R. A. G. Seely, and T. H. Trimble. Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra*, vol. 13 no. 3, pp. 229–296, 1996.

[Ben94] N. Benton. A mixed linear and non-linear logic: Proofs, terms and models. Technical report, Computing Laboratory, University of Cambridge, November 1994.

[Ben95a] N. Benton. A mixed linear and non-linear logic; proofs, terms and models. In *Proceedings of Computer Science Logic '94*, vol. 933 of *Lecture Notes in Computer Science*. Verlag, June 1995. 15 page version.

[Ben95b] N. Benton. Strong normalization for the linear term calculus. *Journal of Functional Programming*, vol. 5 no. 1, pp. 65–80, 1995.

[BGHP97] Andrew Barber, Philippa Gardner, Masahito Hasegawa, and Gordon Plotkin. From action calculi to linear logic. To Appear in CSL '97, 1997.

[Bie95] G. Bierman. *On Intuitionistic Linear Logic*. PhD thesis, Computing Laboratory, University of Cambridge, 1995.

[Bie96a] G.M. Bierman. A classical linear λ-calculus. Technical Report 401, University of Cambridge Computer Laboratory, July 1996.

[Bie96b] G.M. Bierman. Towards a classical linear λ-calculus. In *Proceedings of Tokyo Conference on Linear Logic*, vol. 3 of *Electronic Notes in Computer Science*. Elsevier, 1996.

[BKP89] R. Blackwell, G. M. Kelly, and A. J. Power. Two-dimensional monad theory. *The Journal of Pure and Applied Algebra*, vol. 59, pp. 1–41, 1989.

[BW96] N. Benton and P. Wadler. Linear logic, monads and the lambda calculus. In *Proceedings of the 11th symposium on Logic in Computer Science '96*, 1996.

[CP96] Iliano Cervesato and Frank Pfenning. A linear logical framework. In E. Clarke, editor, *Proceedings of the 11th symposium on Logic in Computer Science*, pp. 264–275, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.

[CS97] J. R. B. Cockett and R. A. G. Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, vol. 14 no. 2, pp. 133–173, 1997.

[Cur85] P.-L. Curien. Categorical combinators. *Information and Control*, vol. 69, pp. 188–254, 1985.

[Cur86] P.-L. Curien. *Categorical combinators, sequential algorithms and functional programming*. Research notes in theoretical computer science. Pitman, London, 1986.

[Day70a] B. J. Day. *Construction of Biclosed Categories*. PhD thesis, University of New South Wales, Australia, 1970.

[Day70b] B. J. Day. On closed categories of functors. In *Midwest Category Seminar Reports IV*, no. 137 in Lecture Notes in Mathematics, pp. 1–38. Springer-Verlag, 1970.

[Day73] B. J. Day. An embedding theorem for closed categories. In *Category Seminar Sydney 1972-73*, no. 420 in Lecture Notes in Mathematics, pp. 55–64. Springer-Verlag, 1973.

[DR89] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, vol. 28 no. 3, pp. 181–203, August 1989.

[FP94] M. P. Fiore and G. D. Plotkin. An axiomatisation of computationally adequate models of PCF. In *Proceedings of the 9th symposium on Logic in Computer Science*, pp. 92–102. IEEE, Computer Society Press, 1994.

[GAL92a] Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In *Proceedings of ACM Symposium Principles of Programming Languages*, pp. 15–26, January 1992.

[GAL92b] Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. Linear logic without boxes. In *Proceedings of the 7th symposium on Logic in Computer Science*, pp. 223–234. IEEE Computer Society Press, 1992.

[Gar95] P. Gardner. A name-free account of action calculi. In S. Brookes, M. Main, A. Melton, and M. Mislove, editors, *Mathematical Foundations of Programming Semantics, 1995*, vol. 1 of *Electronic Notes in Theoretical Computer Science*, Tulane Univ, New Orleans, 1995. Elsevier.

[GH90] Juan C. Guzmán and Paul Hudak. Single-threaded polymorphic lambda calculus. In *Proceedings of the 5th symposium on Logic in Computer Science*, pp. 333–343, Philadelphia, Pennsylvania, June 1990. IEEE Computer Society Press.

[Gha95] Neil Ghani. *Adjoint Rewriting*. PhD thesis, LFCS, University of Edinburgh, Nov. 1995.

[Gha96] Neil Ghani. Adjoint rewriting and the !-type constructor. draft, June 1996.

[Gir87] Jean-Yves Girard. Linear Logic. *Theoretical Computer Science*, vol. 50 no. 1, pp. 1–102, 1987.

[Gir93] Jean-Yves Girard. On the unity of logic. *Annals of Pure and Applied Logic*, vol. 59, pp. 201–217, 1993.

[Gir94] Jean-Yves Girard. Geometry of interaction III : accommodating the additives, 1994. Laboratoirs de Mathématics Discrètes, University of Marseille.

[Gir96] Jean-Yves Girard. Proof-nets; the parallel syntax for proof-theory. In P. Agliano and A. Ursini, editors, *Logic and Algebra*. Marcel Dekker, New York, 1996.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, vol. 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[Has97] Masahito Hasegawa. Recursion from cyclic sharing: traced monoidal categories and models of cyclic λ-calculi. In *Proceedings of TCLA '97, Nancy, France*, LNCS. Springer, April 1997. To appear.

[HdP93] J. M. E. Hyland and Valeria C. V. de Paiva. Full intuitionistic linear logic (extended abstract). *Annals of Pure and Applied Logic*, vol. 64 no. 3, pp. 273–291, 1993.

[HG] M. Hasegawa and P. Gardner. On higher order action calculi and notions of computation. To Appear, TACS '97.

[HHP93] Robert W. Harper, Furio Honsell, and Gordon D. Plotkin. A framework for defining logics. *Journal of the ACM*, vol. 40 no. 1, pp. 143–184, January 1993.

[HM94] Joshua Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, vol. 110 no. 2, pp. 327–365, 1994. A preliminary version appeared in the Proceedings of the 6th symposium on Logic in Computer Science, pages 32–42, Amsterdam, The Netherlands, July 1991.

[HP95] C. Hermida and J. Power. Fibrational control structures. In *Proceedings of CONCUR '95*, vol. 962 of *Lecture Notes in Computer Science*, pp. 117–129. Springer, 1995.

[HPW96] James Harland, David Pym, and Michael Winikoff. Programming in Lygon: An overview. In M. Wirsing and M. Nivat, editors, *Algebraic Methodology and Software Technology*, pp. 391–405, Munich, Germany, July 1996. Springer-Verlag LNCS 1101.

[IP] S. Ishtiaq and D. Pym. A relevant analysis of natural deduction. Submitted.

[KL80] G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, vol. 19, pp. 193–213, 1980.

[KM72] G. M. Kelly and S. MacLane. Coherence for closed categories. *Journal of Pure and Applied Algebra*, vol. 1, pp. 97–140, 1972.

[Laf90] Yves Lafont. Interaction nets. In *Proceedings of the Seventeenth ACM Symposium on Principles of Programming Languages*, pp. 95–108. ACM Press, January 1990.

[Laf95] Yves Lafont. From proof-nets to interaction nets. In *Advances in Linear Logic*, no. 222 in London Math. Soc. Lecture Notes, pp. 225–247. Cambridge University Press, 1995.

[Lam89] Joachim Lambek. Multicategories revisited. In J. W. Gray and A. Scedrov, editors, *Categories in Computer Science and Logic*, vol. 92 of *Contemporary Mathematics*, pp. 217–240, 1989.

[Lam90a] Joachim Lambek. Logic without structural rules. Kleene Festschrift, 1990.

[Lam90b] J. Lamping. An algorithm for optimal lambda calculus reduction. In *Proc. 17th Annual ACM Symposium on Principles of Programming Languages, San Francisco*, pp. 16–30. ACM Press, New York, NY, January 1990.

[Lév80] Jean-Jaques Lévy. Optimal reductions in the lambda-calculus. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pp. 159–191. Academic Press, 1980.

[LM92] P.D. Lincoln and J.C Mitchell. Operational aspects of linear lambda calculus. In *Proceedings of the 7th symposium on Logic in Computer Science*, pp. 235–247, 1992.

[LS86] Joachim Lambek and Philip J. Scott. *Introduction to Higher Order Categorical Logic*, vol. 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1986.

[Mac63] S. MacLane. Natural associativity and commutativity. *Rice University Studies*, vol. 49, pp. 28–46, 1963.

[Mac71] S. MacLane. *Categories for the Working Mathematician*. Springer-Verlag, Berlin, 1971.

[Mac94] Ian Mackie. *The Geometry of Implementation*. PhD thesis, Department of Computing, Imperial College of Science Technology and Medicine, 1994.

[Mil93a] Robin Milner. Action calculi III: Higher-order calculi. Electronically available as ac3.ps at ftp://ftp.cl.cam.ac.uk/users/rm135, July 1993.

[Mil93b] Robin Milner. Action calculi IV: Molecular forms. Electronically available as ac4.ps at ftp://ftp.cl.cam.ac.uk/users/rm135, November 1993.

[Mil93c] Robin Milner. Action calculi, or syntactic action structures. In Andrzej M. Borzyszkowski and Stefan Sokolowski, editors, *Mathematical Foundations of Computer Science 1993, 18th International Symposium*, vol. 711 of *Lecture Notes in Computer Science*, pp. 105–121, Gdansk, Poland, 3 September–30 August 1993. Springer.

[Mil94a] Dale Miller. A multiple-conclusion meta-logic. In S. Abramsky, editor, *Proceedings of the 9th symposium on Logic in Computer Science*, pp. 272–281, Paris, France, July 1994.

[Mil94b] R. Milner. Higher order action calculi. In *Computer Science Logic '93, Selected Papers*, vol. 832 of *Lecture Notes in Computer Science*, pp. 238–260. Springer, 1994.

[Mil94c] Robin Milner. Action calculi V: Reflexive Molecular forms. Electronically available as ac5.ps at ftp://ftp.cl.cam.ac.uk/users/rm135, June 1994.

[Mil96] Robin Milner. Calculi for interaction. *Acta Informatica*, vol. 33 no. 8, pp. 707–737, 1996.

[MMP95] Alex Mifsud, Robin Milner, and John Power. Control structures. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pp. 188–198, San Diego, California, 26–29 June 1995. IEEE Computer Society Press.

[Mog89] Eugenio Moggi. Computational lambda calculus and monads. In *Proceedings of the 4th symposium on Logic in Computer Science*, pp. 14–23. IEEE Computer Society Press, 1989.

[Mog91] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, vol. 93 no. 1, pp. 55–92, July 1991.

[MPP92] Dale Miller, Gordon Plotkin, and David Pym. A relevant analysis of natural deduction. Talk given at the Workshop on Logical Frameworks, Bastad, Sweden, May 1992.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Part I + II. *Information and Computation*, vol. 100 no. 1, pp. 1–77, 1992.

[MRA93] Ian Mackie, Leopoldo Román, and Samson Abramsky. An internal language for autonomous categories. *Journal of Applied Categorical Structures*, vol. 1 no. 3, pp. 311–343, 1993.

[Ong96] C.-H. L. Ong. A semantic view of classical proofs. In *Proceedings of the 11th symposium on Logic in Computer Science*. IEEE, Computer Society Press, 1996.

[OTPT95] P. W. O'Hearn, M. Takayama, A. J. Power, and R. D. Tennent. Syntactic control of interference revisited. In *MFPS XI, conference on Mathematical Foundations of Program Semantics*, vol. 1 of *Electronic Notes in Theoretical Computer Science*. Elsevier, March 1995.

[Par92] Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proceedings of the International conference on Logic Programming and Automated Deduction*, vol. 624 of *Lecture Notes in Computer Science*, pp. 190–201, 1992.

[Par93] Michel Parigot. Strong normalization for second order classical natural deduction. In *Proceedings of the 8th symposium on Logic in Computer Science*, pp. 39–46. IEEE Computer Society Press, 1993.

[Pfe94] Frank Pfenning. Structural cut-elimination in linear logic. Technical Report CMU-CS-94-222, Dept. of Computer Science, Carnegie Mellon Univ., Dec. 1994.

[PH94] David J. Pym and James A. Harland. The uniform proof-theoretic foundation of linear logic programming. *Journal of Logic and Computation*, vol. 4 no. 2, pp. 175–207, April 1994.

[Plo93a] G. Plotkin. Second-order propositional intuitionistic linear logic. Unpublished Notes, 1993.

[Plo93b] G. Plotkin. Type theory and recursion. In *Proceedings of the 8th symposium on Logic in Computer Science, Montreal*, p. 374. IEEE Computer Society Press, 1993. (Abstract).

[Pow96] A. J. Power. Elementary control structures. In *Proceedings of CONCUR '96*, vol. 1119 of *Lecture Notes in Computer Science*, pp. 115–130. Springer, 1996.

[PR94] J. Power and E. P. Robinson. Premonoidal categories and notions of computation. to appear in MSCS, 1994.

[PRT93] Benjamin C. Pierce, Didier Rémy, and David N. Turner. A typed higher-order programming language based on the pi-calculus. In *Workshop on Type Theory and its Application to Computer Systems, Kyoto University*, jul 1993.

[Sch94] H. Schellinx. *The Noble Art of Linear Decorating.* PhD thesis, Institute of Logic, Language and Computation, University of Amsterdam, 1994. ILLC-Dissertation Series, 1994-1.

[See89] Robert A. G. Seely. Linear logic, $*$-autonomous categories and cofree coalgebras. In John W. Gray and Andre Scedrov, editors, *Categories in Computer Science and Logic*, vol. 92 of *Contemporary Mathematics*, pp. 371–382, Providence, Rhode Island, 1989. American Mathematical Society.

[Sol90] S. V. Soloviev. On the conditions of full coherence in closed categories. *Journal of Pure and Applied Algebra*, vol. 69, pp. 301–329, 1990.

[Sza75] F. Szabo. Polycategories. *Communications in Algebra*, vol. 3, pp. 663–689, 1975.

[Wad71] C. P. Wadsworth. *Semantics and Pragmatics of the lambda calculus.* PhD thesis, Oxford, 1971.

[Wad90] Philip Wadler. Linear types can change the world! In M. Broy and C. Jones, editors, *Programming Concepts and Methods*, Sea of Galilee, Israel, April 1990. IFIP TC2 Working Conference on Programming Concepts and Methods, North-Holland.

[Wad92] Philip Wadler. There's no substitute for linear logic. In *Eighth International Conference on the Mathematical Foundations of Programming Semantics*, Oxford, April 1992.

[Wad93] Philip Wadler. A syntax for linear logic. In *Ninth International Conference on the Mathematical Foundations of Programming Semantics*, vol. 802 of *LNCS*, pp. 513–529. Springer Verlag, April 1993.

Index to Notation

a, b . . . . actions, or terms of the action calculus
c . . . . constants in the type-theory $\mathsf{DILL}(\mathbb{C})$
d, e . . . . clauses of relations and pre-relations
f, g, h . . . . arrows in categories
i, j . . . . indices
l, k . . . . linear primes in the action calculus
m, n . . . . arities in the action calculus
o . . . . generalised objects, only in appendix A
p, q . . . . intuitionistic primes in the action calculus
r, s . . . . integers for index bounds
t, u . . . . terms of $\mathsf{DILL}(\mathbb{C})$
v, w . . . . terms of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$
x, y, z . . . . variables from the set X

A, B, C . . . . formulae and types of $\mathsf{DILL}(\mathbb{C})$ and linear types of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$
D, E . . . . relations and pre-relations
F, G . . . . functors between categories
M, N . . . . terms of $\mathsf{ILL}(\mathbb{C})$
O . . . . operators of a signature $\mathbb{O}$
Q, R . . . . intuitionistic types of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$
X . . . . set of variables
X, Y . . . . objects of categories

$\mathbb{A}$ . . . . axiom sets of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$
$\mathcal{A}$ . . . . action models
C . . . . set of assumptions of DILL
$\mathcal{C}$ . . . . constant set from DILL-signatures
$\mathbb{C}$ . . . . DILL-signature
$\mathcal{G}$ . . . . models of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$
$\mathbb{H}$ . . . . higher-order signatures
$\mathcal{H}$ . . . . higher-order models
K . . . . controls of the action calculus
$\mathbb{K}$ . . . . action calculi signatures
$\mathcal{K}$ . . . . operator sets from action calculi signatures
$\mathcal{L}$ . . . . models of $\mathsf{DILL}(\mathbb{C})$

M_L . . . linear type sets from generalised signatures

M_I . . . intuitionistic type sets from generalised signatures

$\mathcal{O}_L$ . . . linear operators of generalised signatures

$\mathcal{O}_I$ . . . intuitionistic operators of generalised signatures

O . . . output-parameterised sets of operators of $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$

$\mathbb{O}$ . . . generalised signatures

P_L . . . linear prime sets from action calculi signatures, and primitive type set from DILL-signatures

P_I . . . intuitionistic prime sets from action calculi signatures

Index to Definitions

Signatures

action calculus ($\mathbb{K}$) ... 121
action calculus, intuitionistic ($\mathbb{K}$) ... 121
dual intuitionistic linear logic ($\mathbb{C}$) ... 35
generalised ($\mathbb{O}$) ... 85
higher-order ($\mathbb{H}$) ... 148

Type Theories

for action calculi ($\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$) ... 124
for action calculi, higher-order ($\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$) ... 139
for action calculi, linear higher-order ($\mathsf{Lin}^{\mathsf{A}^{\boxtimes,\multimap}}(\mathbb{K})$) ... 140
for action calculi, with code ($\mathsf{Lin}^{\mathsf{A}^{\boxtimes}}(\mathbb{K})$) ... 140
of dual intuitionistic linear logic ($\mathsf{DILL}(\mathbb{C})$) ... 35
generalised linear ($\mathsf{Lin}(\mathbb{O}, \mathbb{A})$) ... 87
generalised linear, higher-order ($\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A})$) ... 150
generalised linear, higher-order constant ($\mathsf{Lin}^{\mathsf{HC}}(\mathbb{H}, \mathbb{A})$) ... 152
generalised linear, higher-order output natural ($\mathsf{Lin}^{\mathsf{HC}}_{\mathsf{O}}(\mathbb{H})$) ... 168
of intuitionistic linear logic ($\mathsf{ILL}(\mathbb{C})$) ... 42
for linear logic, generalised linear ($\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$) ... 170

Models

action model ($\mathcal{A}$) ... 143
for dual intuitionistic type-theory ($\mathcal{L}$) ... 61
for general linear type-theory ($\mathcal{G}$) ... 95
for general linear type-theory, output-natural ($\mathcal{G}$) ... 114
for higher-order generalised linear type-theory ($\mathcal{H}$) ... 154

Categories of Models

of small $\mathsf{Lin}^{\mathsf{A}}(\mathbb{K})$-models ($\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{A}}}(\mathbb{K})$) ... 145
of small $\mathsf{Lin}^{\mathsf{A}^{\Rightarrow}}(\mathbb{K})$-models ($\mathsf{Cat}^{\Rightarrow}_{\mathsf{Lin}^{\mathsf{A}}}(\mathbb{K})$) ... 147
of small $\mathsf{Lin}^{\mathsf{D}}(\mathbb{C})$-models ($\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{D}}}(\mathbb{C})$) ... 177
of small $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-models ($\mathsf{Cat}_{\mathsf{Lin}}(\mathbb{O}, \mathbb{A})$) ... 110
of small $\mathsf{Lin}^{\mathsf{HC}}(\mathbb{O}, \mathbb{A})$-models ($\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{HC}}}(\mathbb{O}, \mathbb{A})$) ... 158
of small $\mathsf{Lin}^{\mathsf{H}}(\mathbb{H}, \mathbb{A})$-models ($\mathsf{Cat}_{\mathsf{Lin}^{\mathsf{H}}}(\mathbb{H}, \mathbb{A})$) ... 158
of small action models ($\mathsf{Cat}_{\mathsf{AC}}(\mathbb{K})$) ... 145
of small dill models ($\mathsf{Cat}_{\mathsf{DILL}}(\mathbb{C})$) ... 176
of small higher-order action models ($\mathsf{Cat}_{\mathsf{AC}^{\Rightarrow}}(\mathbb{K})$) ... 147

of small output-natural $\mathsf{Lin}(\mathbb{O}, \mathbb{A})$-models $(\mathsf{Cat}^{\mathsf{O}}_{\mathsf{Lin}}(\mathbb{O}, \mathbb{A}))$ 117