Math. Struct. in Comp. Science (2005), *vol.* 15, *pp.* 323–342. © 2005 Cambridge University Press
doi:10.1017/S0960129504004621 Printed in the United Kingdom

Classical linear logic of implications

MASAHITO HASEGAWA

Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-8502 Japan
and
PRESTO, Japan Science and Technology Agency
Email: hassei@kurims.kyoto-u.ac.jp

Received 16 December 2003; revised 30 June 2004

We give a simple term calculus for the multiplicative exponential fragment of Classical Linear Logic, by extending Barber and Plotkin's dual-context system for the intuitionistic case. The calculus has the non-linear and linear implications as the basic constructs, and this design choice allows a technically manageable axiomatisation without commuting conversions. Despite this simplicity, the calculus is shown to be sound and complete for category-theoretic models given by $*$-autonomous categories with linear exponential comonads.

1. Introduction

We propose a simply typed linear lambda calculus called *Dual Classical Linear Logic* (DCLL) for the multiplicative exponential fragment of Classical Linear Logic (Girard 1987), which is often called MELL in the literature. It may be regarded as an extension of the *Dual Intuitionistic Linear Logic* (DILL) of Barber and Plotkin (Barber 1997; Barber and Plotkin 1997), which is a system for the multiplicative exponential fragment of Intuitionistic Linear Logic (IMELL).

The main feature of DCLL is its *simplicity* and *expressiveness*: just three logical connectives (intuitionistic implication $\rightarrow$, linear implication $\multimap$, and the bottom type $\bot$) and six axioms for the equational theory on terms (proofs), which are just the familiar β and η axioms of the lambda calculus (one of each for $\rightarrow$ and $\multimap$) plus two axioms saying that the type $(\sigma \multimap \bot) \multimap \bot$ is canonically isomorphic to σ. In particular, we can avoid axioms for *commuting conversions* (equalities for identifying terms representing the same proof modulo trivial proof permutations), which have always been troublesome on term calculi for Linear Logic. Other logical connectives and their proof expressions of MELL are easily derived in DCLL; for instance the exponential ! is given by $!\sigma \equiv (\sigma \rightarrow \bot) \multimap \bot$. All the desired equalities between terms, including the commuting conversions, are provable from the simple axioms of DCLL.

Thus DCLL can be used as a compact linear syntax for reasoning about MELL, to complement the drawbacks of conventional proof nets-based presentations, which are often tiresome to formulate and deal with. For instance, it is much easier to describe and analyse the translations between type systems if we use term calculi like DCLL instead of graph-based systems. Also, techniques of logical relations (see, for example, Hasegawa (1999), Streicher (1999) and Hyland and Schalk (2003)) seem to work more smoothly on term-based systems. As future work, we plan to study the compilations of call-by-value programming languages into linearly typed intermediate languages (Berdine *et al.* 2001; Berdine *et al.* 2002; Hasegawa 2002a) using DCLL as a target calculus. In fact, our choice of the logical connectives has been motivated by this research direction – see the discussion in Section 7.

Despite its simplicity, we show that DCLL is sound and complete for categorical models of MELL given by $*$-autonomous categories with symmetric monoidal comonads satisfying some coherence conditions (called linear exponential comonads (Hyland and Schalk 2003)). It turns out that our simple axioms are sufficient to give such a categorical structure for the term model. Although this may not be a great surprise, there do not seem to be many systems for Linear Logic supported by this sort of semantic completeness at the level of proofs, and we think that this completeness result gives a justification for our design of DCLL.

This paper is organised as follows. We introduce the system DCLL in Section 2, with some basic results, which will be used in later sections. Section 3 gives a comparison between DCLL and its precursor DILL. Section 4 then states the completeness result for DCLL with respect to the categorical models for MELL. In Section 5, an extension with additives is discussed. Section 6 is devoted to a variant of DCLL based on the $\lambda\mu$-calculus, called μDCLL. We conclude the paper with some discussions in Section 7. Appendix A gives a summary of DILL, and Appendix B describes an alternative axiomatisation of DCLL with no base type.

This is a revised and expanded version of the work presented at the Computer Science Logic (CSL'02) conference (Hasegawa 2002b).

2. DCLL

2.1. *The system DCLL*

In this paper we employ a 'dual-context'[†] formulation of the linear lambda calculus as developed in Barber and Plotkin (1997) (similar systems are proposed, for example, in Wadler (1993) and Blute *et al.* (1997) – see Barber (1997) for a more comprehensive survey). In this formulation of the linear lambda calculus, a typing judgement takes the form $\Gamma;\ \Delta \vdash M : \tau$, where Γ represents an intuitionistic (or additive) context, while Δ is a linear (multiplicative) context. We assume that all variables in Γ and Δ are distinct. While the variables in Γ can be used in the term M as many times as we like, those in Δ must be used exactly once. A typing judgement $x_1 : \sigma_1, \ldots, x_m : \sigma_m;\ y_1 : \tau_1, \ldots, y_n : \tau_n \vdash M : \sigma$ can be considered as the proof of the sequent $!\sigma_1, \ldots, !\sigma_m, \tau_1, \ldots, \tau_n \vdash \sigma$, or the proposition $!\sigma_1 \otimes \ldots \otimes !\sigma_m \otimes \tau_1 \otimes \ldots \otimes \tau_n \multimap \sigma$.

As we mentioned in the introduction, the system features both intuitionistic (non-linear) arrow type $\rightarrow$ and linear arrow type $\multimap$. We use $\boldsymbol{\lambda} x^\sigma.M$ and $M \mathbin{@} N$ for the non-linear lambda abstraction and application, respectively, and $\lambda x^\sigma.M$ and $M\,N$ for the linear ones.

[†] As noted in Barber and Plotkin (1997), the word 'dual' in DILL (and DCLL) comes from this dual-context typing, and has nothing to do with the duality of Classical Linear Logic.

In order to express the duality of Classical Linear Logic, there is also a special combinator C_σ, which serves as the isomorphism from $(\sigma \multimap \bot) \multimap \bot$ to σ (which, however, can be eliminated when we have no base type – see the discussion at the end of this section). For those familar with the theory of functional programming languages with first-class control primitives, C can be understood as a linear analogue of Felleisen's $\mathscr{C}$-operator (Felleisen *et al.* 1987)[†].

Types and terms

$$\begin{array}{rcl} \sigma & ::= & b \mid \sigma \to \sigma \mid \sigma \multimap \sigma \mid \bot \\ M & ::= & x \mid \boldsymbol{\lambda} x^\sigma.M \mid M \,@\, M \mid \lambda x^\sigma.M \mid M\,M \mid \mathsf{C}_\sigma \end{array}$$

where b ranges over a set of base types. We will sometimes omit the type subscripts for ease of presentation.

Typing

$$\frac{}{\Gamma_1, x:\sigma, \Gamma_2;\ \emptyset \vdash x:\sigma}\ (\text{Int-Ax}) \qquad \frac{}{\Gamma;\ x:\sigma \vdash x:\sigma}\ (\text{Lin-Ax})$$

$$\frac{\Gamma, x:\sigma_1;\ \Delta \vdash M:\sigma_2}{\Gamma;\ \Delta \vdash \boldsymbol{\lambda} x^{\sigma_1}.M : \sigma_1 \to \sigma_2}\ (\to \text{I}) \qquad \frac{\Gamma;\ \Delta \vdash M:\sigma_1 \to \sigma_2 \quad \Gamma;\ \emptyset \vdash N:\sigma_1}{\Gamma;\ \Delta \vdash M \,@\, N:\sigma_2}\ (\to \text{E})$$

$$\frac{\Gamma;\ \Delta, x:\sigma_1 \vdash M:\sigma_2}{\Gamma;\ \Delta \vdash \lambda x^{\sigma_1}.M : \sigma_1 \multimap \sigma_2}\ (\multimap \text{I}) \qquad \frac{\Gamma;\ \Delta_1 \vdash M:\sigma_1 \multimap \sigma_2 \quad \Gamma;\ \Delta_2 \vdash N:\sigma_1}{\Gamma;\ \Delta_1 \sharp \Delta_2 \vdash M\,N:\sigma_2}\ (\multimap \text{E})$$

$$\frac{}{\Gamma;\ \emptyset \vdash \mathsf{C}_\sigma : ((\sigma \multimap \bot) \multimap \bot) \multimap \sigma}\ (\mathsf{C})$$

where $\emptyset$ is the empty context, and $\Delta_1 \sharp \Delta_2$ is a merge of Δ_1 and Δ_2 (Barber 1997; Barber and Plotkin 1997). Thus, $\Delta_1 \sharp \Delta_2$ represents one of the possible merges of Δ_1 and Δ_2 as finite lists. More explicitly, we can define the relation 'Δ is a merge of Δ_1 and Δ_2' inductively as follows (Barber 1997):

— Δ is a merge of $\emptyset$ and Δ
— Δ is a merge of Δ and $\emptyset$
— if Δ is a merge of Δ_1 and Δ_2, then $x:\sigma, \Delta$ is a merge of $x:\sigma, \Delta_1$ and Δ_2
— if Δ is a merge of Δ_1 and Δ_2, then $x:\sigma, \Delta$ is a merge of Δ_1 and $x:\sigma, \Delta_2$.

We assume that, when we introduce $\Delta_1 \sharp \Delta_2$, there is no variable occurring both in Δ_1 and in Δ_2. We note that any typing judgement has a unique derivation (hence a typing judgement can be identified with its derivation).

[†] In fact, in a recent work by Führmann and Thielecke (Führmann and Thielecke 2004), it is observed that Felleisen's $\mathscr{C}$ can also be axiomatised as the canonical isomorphism from the values of type $(\sigma \to 0) \to 0$ to the computations of σ in the typed call-by-value seting.

Axioms

$$\begin{array}{llll}
(\beta_{\to}) & (\lambda x.M) \mathbin{@} N & = M[N/x] & \\
(\eta_{\to}) & \lambda x.M \mathbin{@} x & = M & (x \notin FV(M)) \\
(\beta_{\multimap}) & (\lambda x.M)\, N & = M[N/x] & \\
(\eta_{\multimap}) & \lambda x.M\, x & = M & \\
(\mathsf{C}_1) & L\,(\mathsf{C}_\sigma\, M) & = M\, L & (L : \sigma \multimap \bot) \\
(\mathsf{C}_2) & \mathsf{C}_\sigma\,(\lambda k^{\sigma \multimap \bot}.k\, M) & = M &
\end{array}$$

where $M[N/x]$ denotes the capture-free substitution. Note that there is no side condition $x \notin FV(M)$ for the axiom $\eta_{\multimap}$ (or C_2), as linearity prevents x from occuring in M. The equality judgement $\Gamma;\ \Delta \vdash M = N : \sigma$ for $\Gamma;\ \Delta \vdash M : \sigma$ and $\Gamma;\ \Delta \vdash N : \sigma$ is defined as usual.

We note that the axiom C_1 is equivalent to $\lambda k^{\sigma \multimap \bot}.k\,(\mathsf{C}_\sigma\, M) = M$; thus the last two axioms say that C_σ is the inverse of $\lambda x^\sigma.\lambda k^{\sigma \multimap \bot}.k\, x : \sigma \multimap (\sigma \multimap \bot) \multimap \bot$. As a consequence, we obtain the 'naturality' of C for free, as shown by the following lemma.

Lemma 2.1. The following equation is provable in DCLL:

$$L^{\sigma \multimap \tau}\,(\mathsf{C}_\sigma\, M^{(\sigma \multimap \bot) \multimap \bot}) \;=\; \mathsf{C}_\tau\,(\lambda k^{\tau \multimap \bot}.M\,(\lambda x^\sigma.k\,(L\, x))) \;:\; \tau.$$

$$\begin{tikzcd}
(\sigma \multimap \bot) \multimap \bot \arrow[r, "(L \multimap \bot) \multimap \bot"] \arrow[d, "\mathsf{C}_\sigma"'] & (\tau \multimap \bot) \multimap \bot \arrow[d, "\mathsf{C}_\tau"] \\
\sigma \arrow[r, "L"'] & \tau
\end{tikzcd}$$

Proof.

$$\begin{aligned}
L\,(\mathsf{C}\, M) &\overset{\mathsf{C}_2}{=} \mathsf{C}\,(\lambda k.k\,(L\,(\mathsf{C}\, M))) \\
&\overset{\beta_{\multimap}}{=} \mathsf{C}\,(\lambda k.(\lambda x.k\,(L\, x))\,(\mathsf{C}\, M)) \\
&\overset{\mathsf{C}_1}{=} \mathsf{C}\,(\lambda k.M\,(\lambda x.k\,(L\, x))).
\end{aligned}$$

□

2.2. *Some basic results for DCLL*

In DCLL, the equations in the following lemma are provable.

Lemma 2.2.

1 $\mathsf{C}_\bot = \lambda m^{(\bot \multimap \bot) \multimap \bot}.m\,(\lambda x^\bot.x)$.
2 $\mathsf{C}_{\sigma \to \tau} = \lambda m^{((\sigma \to \tau) \multimap \bot) \multimap \bot}.\lambda x^\sigma.\mathsf{C}_\tau\,(\lambda k^{\tau \multimap \bot}.m\,(\lambda f^{\sigma \to \tau}.k\,(f \mathbin{@} x)))$.
3 $\mathsf{C}_{\sigma \multimap \tau} = \lambda m^{((\sigma \multimap \tau) \multimap \bot) \multimap \bot}.\lambda x^\sigma.\mathsf{C}_\tau\,(\lambda k^{\tau \multimap \bot}.m\,(\lambda f^{\sigma \multimap \tau}.k\,(f\, x)))$.

Proof.

1
$$\begin{aligned}
\mathsf{C}_\bot\, m &= (\lambda x^\bot.x)\,(\mathsf{C}_\bot\, m) \\
&= m\,(\lambda x^\bot.x).
\end{aligned}$$

2
$$\begin{aligned}\mathsf{C}_{\sigma\to\tau}\, m \mathbin{@} x &= \mathsf{C}_\tau\,(\lambda k.k\,(\mathsf{C}_{\sigma\to\tau}\, m \mathbin{@} x))\\ &= \mathsf{C}_\tau\,(\lambda k.(\lambda f.k\,(f \mathbin{@} x))\,(\mathsf{C}_{\sigma\to\tau}\, m))\\ &= \mathsf{C}_\tau\,(\lambda k.m\,(\lambda f.k\,(f \mathbin{@} x))).\end{aligned}$$

3
$$\begin{aligned}\mathsf{C}_{\sigma\multimap\tau}\, m\, x &= \mathsf{C}_\tau\,(\lambda k.k\,(\mathsf{C}_{\sigma\multimap\tau}\, m\, x))\\ &= \mathsf{C}_\tau\,(\lambda k.(\lambda f.k\,(f\, x))\,(\mathsf{C}_{\sigma\multimap\tau}\, m))\\ &= \mathsf{C}_\tau\,(\lambda k.m\,(\lambda f.k\,(f\, x))).\end{aligned}$$ □

By induction we can show the following proposition.

Proposition 2.1. For $\sigma = \sigma_1 \Rightarrow_1 \ldots \sigma_n \Rightarrow_n \bot$ (where $\Rightarrow_i$ is either $\to$ or $\multimap$)

$$\mathsf{C}_\sigma\, M \star_1 N_1 \ldots \star_n N_n = M\,(\lambda f^\sigma.f \star_1 N_1 \ldots \star_n N_n) : \bot$$

is provable in DCLL, where $M : (\sigma \multimap \bot) \multimap \bot$, $N_i : \sigma_i$, and $\star_i$ is a non-linear application if $\Rightarrow_i$ is $\to$, or a linear application if $\Rightarrow_i$ is $\multimap$.

We can now give an interesting implication of these results. If we do not have base types, all DCLL terms can be expressed as just (non-linear and linear) lambda terms, without using the combinator C; we can *define* C's as lambda terms by the equations of Lemma 2.2 or Proposition 2.1. Note that, if we do so, the axiom C_2 follows just from the $\beta\eta$-axioms for $\to$ and $\multimap$. Therefore it is possible to axiomatise DCLL with no base type as a quotient of the $\{\to, \multimap\}$-calculus on the single base type $\bot$ obtained by adding the axiom C_1 for these defined C's. In fact, all of them are derivable from the following single instance and the $\beta\eta$-axioms for $\to$ and $\multimap$:

$$L\,(\lambda x^\sigma.M\,(\lambda f^{\sigma\multimap\bot}.f\,x)) = M\,L$$

where $L : (\sigma \multimap \bot) \multimap \bot$ and $M : ((\sigma \multimap \bot) \multimap \bot) \multimap \bot$. So it suffices to have just the standard $\beta\eta$-axioms and this equation: Appendix B describes the resulting system.

Remark 2.1. The last equation, if one replaces $\bot$ by I, in fact amounts to the infamous (in)equality known as the 'triple unit problem', which asks if two canonical endomorphisms on $((A \multimap I) \multimap I) \multimap I$ are the same in a symmetric monoidal closed category (Murawski and Ong 1999; Kelly and Mac Lane 1971).

3. DILL in DCLL

The primitive constructs of DILL (summarised in Appendix A) can be defined in DCLL as follows:

$$\begin{aligned} I &\equiv \bot \multimap \bot\\ \sigma_1 \otimes \sigma_2 &\equiv (\sigma_1 \multimap \sigma_2 \multimap \bot) \multimap \bot\\ !\,\sigma &\equiv (\sigma \to \bot) \multimap \bot \end{aligned}$$

$$\begin{array}{ll}
* & \equiv \lambda x^{\bot}.x \\
\mathsf{let}\ *\ \mathsf{be}\ M^{I}\ \mathsf{in}\ N^{\tau} & \equiv \mathsf{C}_{\tau}\,(\lambda k^{\tau\multimap\bot}.M\,(k\,N)) \\
M^{\sigma_1}\otimes N^{\sigma_2} & \equiv \lambda k^{\sigma_1\multimap\sigma_2\multimap\bot}.k\,M\,N \\
\mathsf{let}\ x^{\sigma_1}\otimes y^{\sigma_2}\ \mathsf{be}\ M^{\sigma_1\otimes\sigma_2}\ \mathsf{in}\ N^{\tau} & \equiv \mathsf{C}_{\tau}\,(\lambda k^{\tau\multimap\bot}.M\,(\lambda x^{\sigma_1}.\lambda y^{\sigma_2}.k\,N)) \\
!M^{\sigma} & \equiv \lambda h^{\sigma\to\bot}.h\,@\,M \\
\mathsf{let}\ !x^{\sigma}\ \mathsf{be}\ M^{!\sigma}\ \mathsf{in}\ N^{\tau} & \equiv \mathsf{C}_{\tau}\,(\lambda k^{\tau\multimap\bot}.M\,(\lambda x^{\sigma}.k\,N)).
\end{array}$$

We can also introduce connectives ? and $\invamp$ by $?\sigma \equiv (\sigma \multimap \bot) \to \bot$ and $\sigma_1 \invamp \sigma_2 \equiv (\sigma_1 \multimap \bot) \multimap (\sigma_2 \multimap \bot) \multimap \bot$ (or $(\sigma_1 \multimap \bot) \multimap \sigma_2$, if we prefer a less symmetric but shorter encoding). However, giving the term expressions associated to these connectives seems less obvious – there seems to be no agreed syntax for them in the literature.

We shall see below that this encoding is sound for both the typing and equational theory.

3.1. *Type soundness*

Lemma 3.1. The derivation rules of typing judgements in DILL are derivable in DCLL.

Proof. We shall spell out the cases of introduction and elimination rules for !

$$\dfrac{\Gamma;\ \emptyset \vdash M : \sigma}{\Gamma;\ \emptyset \vdash !M : !\sigma}\ (!\,\mathrm{I})$$

$$\dfrac{\Gamma;\ \Delta_1 \vdash M : !\sigma \quad \Gamma, x:\sigma;\ \Delta_2 \vdash N : \tau}{\Gamma;\ \Delta_1\sharp\Delta_2 \vdash \mathsf{let}\ !x^{\sigma}\ \mathsf{be}\ M\ \mathsf{in}\ N : \tau}\ (!\,\mathrm{E}),$$

which are derivable in DCLL as follows.

$$\dfrac{\dfrac{\dfrac{}{\Gamma;\ h:\sigma\to\bot \vdash h:\sigma\to\bot}\ \text{Lin-Ax} \qquad \Gamma;\ \emptyset\vdash M:\sigma}{\Gamma;\ h:\sigma\to\bot \vdash h\,@\,M:\bot}\ {\to}\mathrm{E}}{\Gamma;\ \emptyset\vdash !M \equiv \lambda h^{\sigma\to\bot}.h\,@\,M : (\sigma\to\bot)\multimap\bot \equiv !\sigma}\ {\multimap}\mathrm{I}$$

$$\dfrac{\dfrac{}{\Gamma;\ \emptyset\vdash \mathsf{C}_{\tau} : ((\tau\multimap\bot)\multimap\bot)\multimap\tau}\ \mathsf{C} \qquad \dfrac{\dfrac{\Gamma;\ \Delta_1\vdash M : !\sigma \equiv (\sigma\to\bot)\multimap\bot \qquad \dfrac{\dfrac{\dfrac{}{\Gamma,x:\sigma;\ k:\tau\multimap\bot\vdash k:\tau\multimap\bot}\ \text{Lin-Ax} \qquad \Gamma,x:\sigma;\ \Delta_2\vdash N:\tau}{\Gamma,x:\sigma;\ \Delta_2,k:\tau\multimap\bot\vdash k\,N:\bot}\ {\multimap}\mathrm{E}}{\Gamma;\ \Delta_2,k:\tau\multimap\bot\vdash \lambda x^{\sigma}.k\,N:\sigma\to\bot}\ {\to}\mathrm{I}}{\Gamma;\ \Delta_1\sharp\Delta_2,k:\tau\multimap\bot\vdash M\,(\lambda x^{\sigma}.k\,N):\bot}\ {\multimap}\mathrm{E}}{\Gamma;\ \Delta_1\sharp\Delta_2\vdash \lambda k^{\tau\multimap\bot}.M\,(\lambda x^{\sigma}.k\,N) : (\tau\multimap\bot)\multimap\bot}\ {\multimap}\mathrm{I}}{\Gamma;\ \Delta_1\sharp\Delta_2\vdash \mathsf{let}\ !x^{\sigma}\ \mathsf{be}\ M\ \mathsf{in}\ N \equiv \mathsf{C}_{\tau}\,(\lambda k^{\tau\multimap\bot}.M\,(\lambda x^{\sigma}.k\,N)) : \tau}\ {\multimap}\mathrm{E}$$

The rules for I and $\otimes$ are derived similarly. □

3.2. *A reduced axiomatisation of DILL*

Before showing the equational soundness of the encoding, we shall give an alternative simple axiomatisation of DILL, in which the η-axioms other than $\eta_{\multimap}$ and all commuting conversions are replaced by just three simple equations.

Proposition 3.1. The equational theory of DILL can be axiomatised by the following set of axioms:

$$
\begin{array}{lrcll}
(\beta_I) & \mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ M & = & M & \\
(\beta_\otimes) & \mathsf{let}\ x\otimes y\ \mathsf{be}\ M\otimes N\ \mathsf{in}\ L & = & L[M/x, N/y] & \\
(\beta_\multimap) & (\lambda x.M)\,N & = & M[N/x] & \\
(\beta_!) & \mathsf{let}\ !x\ \mathsf{be}\ !M\ \mathsf{in}\ N & = & N[M/x] & \\
(\eta_\multimap) & \lambda x.M\,x & = & M & \\
(\mathbf{com}_I) & \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ L\,* & = & L\,M & \\
(\mathbf{com}_\otimes) & \mathsf{let}\ x\otimes y\ \mathsf{be}\ M\ \mathsf{in}\ L\,(x\otimes y) & = & L\,M & \\
(\mathbf{com}_!) & \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ L\,(!x) & = & L\,M & (x\notin FV(L)).
\end{array}
$$

Proof. The η-axioms for I, $\otimes$ and ! follow from the (**com**)-axioms and $(\beta_\multimap)$ by just letting L's be the identities. Commuting conversions are derived as follows:

$$
\begin{array}{rcll}
C[\mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ N] & = & (\lambda u^I.C[\mathsf{let}\ *\ \mathsf{be}\ u\ \mathsf{in}\ N])\,M & (\beta_\multimap)\\
& = & \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ (\lambda u^I.C[\mathsf{let}\ *\ \mathsf{be}\ u\ \mathsf{in}\ N])\,* & (\mathbf{com}_I)\\
& = & \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ C[\mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ N] & (\beta_\multimap)\\
& = & \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ C[N] & (\beta_I)
\end{array}
$$

$$
\begin{array}{rcll}
C[\mathsf{let}\ x^{\sigma_1}\otimes y^{\sigma_2}\ \mathsf{be}\ M\ \mathsf{in}\ N] & = & (\lambda w^{\sigma_1\otimes\sigma_2}.C[\mathsf{let}\ x^{\sigma_1}\otimes y^{\sigma_2}\ \mathsf{be}\ w\ \mathsf{in}\ N])\,M & (\beta_\multimap)\\
& = & \mathsf{let}\ x'^{\sigma_1}\otimes y'^{\sigma_2}\ \mathsf{be}\ M\ \mathsf{in} & \\
& & (\lambda w^{\sigma_1\otimes\sigma_2}.C[\mathsf{let}\ x^{\sigma_1}\otimes y^{\sigma_2}\ \mathsf{be}\ w\ \mathsf{in}\ N])\,(x'\otimes y') & (\mathbf{com}_\otimes)\\
& = & \mathsf{let}\ x'^{\sigma_1}\otimes y'^{\sigma_2}\ \mathsf{be}\ M\ \mathsf{in} & \\
& & C[\mathsf{let}\ x^{\sigma_1}\otimes y^{\sigma_2}\ \mathsf{be}\ x'\otimes y'\ \mathsf{in}\ N] & (\beta_\multimap)\\
& = & \mathsf{let}\ x\otimes y\ \mathsf{be}\ M\ \mathsf{in}\ C[N] & (\beta_\otimes)
\end{array}
$$

$$
\begin{array}{rcll}
C[\mathsf{let}\ !x^{\sigma}\ \mathsf{be}\ M\ \mathsf{in}\ N] & = & (\lambda v^{!\sigma}.C[\mathsf{let}\ !x^{\sigma}\ \mathsf{be}\ v\ \mathsf{in}\ N])\,M & (\beta_\multimap)\\
& = & \mathsf{let}\ x'^{\sigma}\ \mathsf{be}\ M\ \mathsf{in}\ (\lambda v^{!\sigma}.C[\mathsf{let}\ !x^{\sigma}\ \mathsf{be}\ v\ \mathsf{in}\ N])\,(!x') & (\mathbf{com}_!)\\
& = & \mathsf{let}\ x'^{\sigma}\ \mathsf{be}\ M\ \mathsf{in}\ C[\mathsf{let}\ !x^{\sigma}\ \mathsf{be}\ !x'\ \mathsf{in}\ N] & (\beta_\multimap)\\
& = & \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ C[N] & (\beta_!)\quad \square
\end{array}
$$

Remark 3.1. The (**com**)-axioms are equations ensuring, respectively, the following canonical isomorphisms:

$$
\begin{array}{rcl}
I\multimap\tau & \simeq & \tau\\
(\sigma_1\otimes\sigma_2)\multimap\tau & \simeq & \sigma_1\multimap\sigma_2\multimap\tau\\
!\sigma\multimap\tau & \simeq & \sigma\to\tau.
\end{array}
$$

3.3. *Equational soundness*

Theorem 3.1. All equations derivable in DILL are derivable in DCLL via the encoding.

Proof. We shall check the each axiom of the reduced axiomatisation given above. The β-axioms are easy:

$$\begin{aligned}
\mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ N &\equiv \mathsf{C}\,(\lambda k.(\lambda x.x)\,(k\,N)) \\
&= \mathsf{C}\,(\lambda k.k\,N) \\
&= N.
\end{aligned}$$

$$\begin{aligned}
\mathsf{let}\ x \otimes y\ \mathsf{be}\ M_1 \otimes M_2\ \mathsf{in}\ N &\equiv \mathsf{C}\,(\lambda k.(\lambda h.h\,M_1\,M_2)\,(\lambda x.\lambda y.k\,N)) \\
&= \mathsf{C}\,(\lambda k.(\lambda x.\lambda y.k\,N)\,M_1\,M_2) \\
&= \mathsf{C}\,(\lambda k.k\,N[M_1/x, M_2/y]) \\
&= N[M_1/x, M_2/y].
\end{aligned}$$

$$\begin{aligned}
\mathsf{let}\ !x\ \mathsf{be}\ !M\ \mathsf{in}\ N &\equiv \mathsf{C}\,(\lambda k.(\lambda h.h \mathbin{@} M)\,(\boldsymbol{\lambda} x.k\,N)) \\
&= \mathsf{C}\,(\lambda k.(\boldsymbol{\lambda} x.k\,N) \mathbin{@} M) \\
&= \mathsf{C}\,(\lambda k.k\,N[M/x]) \\
&= N[M/x].
\end{aligned}$$

The $\eta_{\multimap}$ axiom is included in the axioms of DCLL. There remain three **com**-axioms:

$$\begin{aligned}
\mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ L^{I \multimap \tau}\,* &\equiv \mathsf{C}_\tau\,(\lambda k.M\,(k\,(L\,(\lambda x.x)))) \\
&= \mathsf{C}_\tau\,(\lambda k.(\lambda h.M\,(h\,(\lambda x.x)))\,(\lambda u.k\,(L\,u))) \\
&= L\,(\mathsf{C}_I\,(\lambda h.M\,(h\,(\lambda x.x)))) && \text{(Lem.2.1)} \\
&= L\,(\lambda y.(\lambda h.M\,(h\,(\lambda x.x)))\,(\lambda f.f\,y)) && \text{(Prop.2.1)} \\
&= L\,(\lambda y.M\,((\lambda f.f\,y)\,(\lambda x.x))) \\
&= L\,(\lambda y.M\,((\lambda x.x)\,y)) \\
&= L\,(\lambda y.M\,y) \\
&= L\,M.
\end{aligned}$$

$$\begin{aligned}
&\mathsf{let}\ x^{\sigma_1} \otimes y^{\sigma_2}\ \mathsf{be}\ M\ \mathsf{in} \\
&L^{\sigma_1 \otimes \sigma_2 \multimap \tau}\,(x \otimes y) \equiv \mathsf{C}_\tau\,(\lambda k.M\,(\lambda xy.k\,(L\,(\lambda n.n\,x\,y)))) \\
&\qquad = \mathsf{C}_\tau\,(\lambda k.(\lambda h.M\,(\lambda xy.h\,(\lambda n.n\,x\,y)))\,(\lambda u.k\,(L\,u))) \\
&\qquad = L\,(\mathsf{C}_{\sigma_1 \otimes \sigma_2}\,(\lambda h.M\,(\lambda xy.h\,(\lambda n.n\,x\,y)))) && \text{(Lem.2.1)} \\
&\qquad = L\,(\lambda z.(\lambda h.M\,(\lambda xy.h\,(\lambda n.n\,x\,y)))\,(\lambda f.f\,z)) && \text{(Prop.2.1)} \\
&\qquad = L\,(\lambda z.M\,(\lambda xy.(\lambda f.f\,z)\,(\lambda n.n\,x\,y))) \\
&\qquad = L\,(\lambda z.M\,(\lambda xy.(\lambda n.n\,x\,y)\,z)) \\
&\qquad = L\,(\lambda z.M\,(\lambda xy.z\,x\,y)) \\
&\qquad = L\,(\lambda z.M\,z) \\
&\qquad = L\,M.
\end{aligned}$$

$$\begin{aligned}
\mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ L^{!\sigma \multimap \tau}\,(!x) &\equiv \mathsf{C}_\tau\,(\lambda k.M\,(\boldsymbol{\lambda} x.k\,(L\,(\lambda h.h \mathbin{@} x)))) \\
&= \mathsf{C}_\tau\,(\lambda k.(\lambda m.M\,(\boldsymbol{\lambda} x.m\,(\lambda h.h \mathbin{@} x)))\,(\lambda u.k\,(L\,u))) \\
&= L\,(\mathsf{C}_{!\sigma}\,(\lambda m.M\,(\boldsymbol{\lambda} x.m\,(\lambda h.h \mathbin{@} x)))) && \text{(Lem.2.1)} \\
&= L\,(\lambda y.(\lambda m.M\,(\boldsymbol{\lambda} x.m\,(\lambda h.h \mathbin{@} x)))\,(\lambda f.f\,y)) && \text{(Prop.2.1)} \\
&= L\,(\lambda y.M\,(\boldsymbol{\lambda} x.(\lambda f.f\,y)\,(\lambda h.h \mathbin{@} x))) \\
&= L\,(\lambda y.M\,(\boldsymbol{\lambda} x.(\lambda h.h \mathbin{@} x)\,y)) \\
&= L\,(\lambda y.M\,(\boldsymbol{\lambda} x.y\,x)) \\
&= L\,(\lambda y.M\,y) \\
&= L\,M.
\end{aligned}$$

□

4. Completeness for categorical models

An important implication of Theorem 3.1, when taken together with the result in Barber and Plotkin (1997) (completeness via the term model construction), is that the term model of DCLL forms a model of DILL, that is, a symmetric monoidal closed category equipped with a symmetric monoidal comonad satisfying certain coherence conditions (see, for example, Seely (1989) and Bierman (1995)), which we shall call a 'linear exponential comonad', following Hyland and Schalk (2003).

Definition 4.1 (Linear exponential comonad). A symmetric monoidal comonad $! = (!, \varepsilon, \delta, m_{A,B}, m_I)$ on a symmetric monoidal category $\mathscr{C}$ is called a *linear exponential comonad* when the category of its coalgebras is a category of commutative comonoids – that is:

- there are specified monoidal natural transformations $e_A : !A \to I$ and $d_A : !A \to !A \otimes !A$ that form a commutative comonoid $(!A, e_A, d_A)$ in $\mathscr{C}$ and are also coalgebra morphisms from $(!A, \delta_A)$ to (I, m_I) and $(!A \otimes !A, m_{!A,!A} \circ (\delta_A \otimes \delta_A))$, respectively; and
- any coalgebra morphism from $(!A, \delta_A)$ to $(!B, \delta_B)$ is also a comonoid morphism from $(!A, e_A, d_A)$ to $(!B, e_B, d_B)$.

Remark 4.1. In Barber and Plotkin (1997), a model of DILL is described as a symmetric monoidal adjunction between a cartesian closed category and a symmetric monoidal closed category (Benton's LNL model (Benton 1995)). It is known that such an 'adjunction model' gives rise to a linear exponential comonad on the symmetric monoidal closed category part. Conversely, a symmetric monoidal closed category with a linear exponential comonad has at least one symmetric monoidal adjunction from a cartesian closed category so that it induces the linear exponential comonad (however, such an adjunction is not, in general, unique). Therefore, for our purposes (the completeness result as stated here), it does not matter which class of structures we choose as models. (However, we must be careful when we talk about the morphisms between models, for example, to use the term model of DILL (or DCLL) as a classifying category of such structures. In particular, although we have the completeness result below, the term model of DCLL is *not* isomorphic to the free $*$-autonomous category with a linear exponential comonad – it is only equivalent to such a free structure via a suitable structure-preserving equivalence.)

Moreover, the symmetric monoidal closed category given by the term model of DCLL is a *$*$-autonomous category* (Barr 1979; Barr 1991) if we take $\bot$ as the dualising object. Recall that a $*$-autonomous category can be characterised as a symmetric monoidal closed category with an object $\bot$ such that the canonical morphism from σ to $(\sigma \multimap \bot) \multimap \bot$ is an isomorphism – in the term model of DCLL, the inverse is given by the combinator C_σ.

On the other hand, all the axioms of DCLL are sound with respect to interpretations in such categorical models, where a typing judgement

$$x_1 : \sigma_1, \ldots, x_m : \sigma_m;\ y_1 : \tau_1, \ldots, y_n : \tau_n \vdash M : \sigma$$

is inductively interpreted as a morphism $[\![x_1 : \sigma_1, \ldots; y_1 : \tau_1, \ldots \vdash M : \sigma]\!]$ from $![\![\sigma_1]\!] \otimes \ldots \otimes ![\![\sigma_m]\!] \otimes [\![\tau_1]\!] \otimes \ldots \otimes [\![\tau_n]\!]$ to $[\![\sigma]\!]$ in the $*$-autonomous category with the linear exponential comonad !. Thus we have the following theorem.

Theorem 4.1 (Categorical completeness). The equational theory of DCLL is sound and complete for categorical models given by $*$-autonomous categories with linear exponential comonads: $\Gamma;\ \Delta \vdash M = N : \sigma$ is provable if and only if $[\![\Gamma;\ \Delta \vdash M : \sigma]\!] = [\![\Gamma;\ \Delta \vdash N : \sigma]\!]$ holds for every such model.

5. Additives

It is fairly routine to enrich DCLL with additives. We add the cartesian product $\&$ and its unit $\top$, and terms

$$\frac{}{\Gamma;\ \Delta \vdash \langle\,\rangle : \top}\ (\top\,\mathrm{I}) \qquad \frac{\Gamma;\ \Delta \vdash M : \sigma \quad \Gamma;\ \Delta \vdash N : \tau}{\Gamma;\ \Delta \vdash \langle M, N\rangle : \sigma \,\&\, \tau}\ (\&\,\mathrm{I})$$

$$\frac{\Gamma;\ \Delta \vdash M : \sigma \,\&\, \tau}{\Gamma;\ \Delta \vdash \mathsf{fst}_{\sigma,\tau}\, M : \sigma}\ (\&\,\mathrm{E}_L) \qquad \frac{\Gamma;\ \Delta \vdash M : \sigma \,\&\, \tau}{\Gamma;\ \Delta \vdash \mathsf{snd}_{\sigma,\tau}\, M : \tau}\ (\&\,\mathrm{E}_R)$$

together with the standard axioms

$$\begin{array}{lll} M & = \langle\,\rangle & (M : \top) \\ \mathsf{fst}\,\langle M, N\rangle & = M & \\ \mathsf{snd}\,\langle M, N\rangle & = N & \\ \langle \mathsf{fst}\, M, \mathsf{snd}\, M\rangle & = M. & \end{array}$$

Again, we do not need any additional axiom for commuting conversions. Furthermore, it is possible to eliminate the C combinators for additives, as we can prove (using Lemma 2.1 for the latter case).

Lemma 5.1.

1 $\mathsf{C}_\top = \lambda m^{(\top \multimap \bot) \multimap \bot}.\langle\,\rangle$.

2 $\mathsf{C}_{\sigma \,\&\, \tau} = \lambda m^{((\sigma \,\&\, \tau) \multimap \bot) \multimap \bot}.$
$$\langle \mathsf{C}_\sigma\,(\lambda k^{\sigma \multimap \bot}.m\,(\lambda z^{\sigma \,\&\, \tau}.k\,(\mathsf{fst}_{\sigma,\tau}\, z))), \mathsf{C}_\tau\,(\lambda h^{\tau \multimap \bot}.m\,(\lambda z^{\sigma \,\&\, \tau}.h\,(\mathsf{snd}_{\sigma,\tau}\, z)))\rangle.$$

As a consequence, if we do not have base types, it is possible to axiomatise DCLL with additives as a quotient of a typed lambda calculus (with $\rightarrow$, $\multimap$, $\top$, $\&$) on a single base type $\bot$, in the same way as described at the end of Section 2.

The coproduct $\oplus$ and its unit 0 are given by $\sigma_1 \oplus \sigma_2 \equiv ((\sigma_1 \multimap \bot) \,\&\, (\sigma_2 \multimap \bot)) \multimap \bot$ and $0 \equiv \top \multimap \bot$, as usual. The associated term constructs are

$$\frac{\Gamma;\ \Delta \vdash M : 0}{\Gamma;\ \Delta \vdash \mathsf{abort}_\sigma\, M \equiv \mathsf{C}_\sigma\,(\lambda k^{\sigma \multimap \bot}.M\,\langle\,\rangle) : \sigma}\ (0\,\mathrm{E})$$

$$\frac{\Gamma;\ \Delta \vdash M : \sigma}{\Gamma;\ \Delta \vdash \mathsf{inl}_{\sigma,\tau}\, M \equiv \lambda k^{(\sigma \multimap \bot) \,\&\, (\tau \multimap \bot)}.\mathsf{fst}_{\sigma \multimap \bot, \tau \multimap \bot}\, k\, M : \sigma \oplus \tau}\ (\oplus\,\mathrm{I}_L)$$

$$\frac{\Gamma;\ \Delta \vdash N : \tau}{\Gamma;\ \Delta \vdash \mathsf{inr}_{\sigma,\tau}\, N \equiv \lambda k^{(\sigma \multimap \bot) \,\&\, (\tau \multimap \bot)}.\mathsf{snd}_{\sigma \multimap \bot, \tau \multimap \bot}\, k\, N : \sigma \oplus \tau}\ (\oplus\,\mathrm{I}_R)$$

$$\frac{\Gamma;\ \Delta_1 \vdash L : \sigma \oplus \tau \quad \Gamma;\ \Delta_2, x : \sigma \vdash M : \theta \quad \Gamma;\ \Delta_2, y : \tau \vdash N : \theta}{\begin{array}{r}\Gamma;\ \Delta_1 \sharp \Delta_2 \vdash \mathsf{case}\ L\ \mathsf{of}\ \mathsf{inl}\, x^\sigma \mapsto M \parallel \mathsf{inr}\, y^\tau \mapsto N \equiv \\ \mathsf{C}_\theta\,(\lambda k^{\theta \multimap \bot}.L\,\langle \lambda x^\sigma.k\, M, \lambda y^\tau.k\, N\rangle) : \theta\end{array}}\ (\oplus\,\mathrm{E}).$$

These satisfy the standard axioms for coproducts as well as commuting conversion axioms.

A category-theoretic model of DCLL extended with additives can be given as a ∗-autonomous category with a linear exponential comonad and finite products. The soundness and completeness results in the last section can be easily extended to this setting.

6. Formulation based on the $\lambda\mu$-calculus

Instead of the combinator C for the double-negation elimination, we could use the syntax of the $\lambda\mu$-calculus (Parigot 1992) for expressing the duality, as done in Koh and Ong (1999) for the multiplicative fragment (MLL). Below, we present such a system, μDCLL, which is routinely seen to be equivalent to DCLL. While the $\lambda\mu$-calculus style formulation requires us to introduce yet another typing context, a potential benefit of the $\lambda\mu$-calculus approach is that it may give a confluent and normalising reduction system (up to a certain equivalence class of terms, as in Koh and Ong (1999)); it also allows a natural treatment of the connective ⅋ (by introducing the binary μ-bindings). Bierman (1999) also has relevant results.

6.1. *The system μDCLL*

Types and Terms

$$\begin{array}{rcl} \sigma & ::= & b \mid \sigma \to \sigma \mid \sigma \multimap \sigma \mid \bot \\ M & ::= & x \mid \boldsymbol{\lambda} x^\sigma.M \mid M \mathbin{@} M \mid \lambda x^\sigma.M \mid M\,M \mid [\alpha]M \mid \mu\alpha^\sigma.M. \end{array}$$

Typing

$$\frac{}{\Gamma_1, x:\sigma, \Gamma_2;\ \emptyset \vdash x:\sigma \mid \Sigma}\ (\text{Int-Ax}) \qquad \frac{}{\Gamma;\ x:\sigma \vdash x:\sigma \mid \emptyset}\ (\text{Lin-Ax})$$

$$\frac{\Gamma, x:\sigma_1;\ \Delta \vdash M:\sigma_2 \mid \Sigma}{\Gamma;\ \Delta \vdash \boldsymbol{\lambda} x^{\sigma_1}.M:\sigma_1 \to \sigma_2 \mid \Sigma}\ (\to\text{I}) \qquad \frac{\Gamma;\ \Delta \vdash M:\sigma_1 \to \sigma_2 \mid \Sigma \qquad \Gamma;\ \emptyset \vdash N:\sigma_1 \mid \emptyset}{\Gamma;\ \Delta \vdash M \mathbin{@} N:\sigma_2 \mid \Sigma}\ (\to\text{E})$$

$$\frac{\Gamma;\ \Delta, x:\sigma_1 \vdash M:\sigma_2 \mid \Sigma}{\Gamma;\ \Delta \vdash \lambda x^{\sigma_1}.M:\sigma_1 \multimap \sigma_2 \mid \Sigma}\ (\multimap\text{I}) \qquad \frac{\Gamma;\ \Delta_1 \vdash M:\sigma_1 \multimap \sigma_2 \mid \Sigma_1 \qquad \Gamma;\ \Delta_2 \vdash N:\sigma_1 \mid \Sigma_2}{\Gamma;\ \Delta_1 \sharp \Delta_2 \vdash MN:\sigma_2 \mid \Sigma_1 \sharp \Sigma_2}\ (\multimap\text{E})$$

$$\frac{\Gamma;\ \Delta \vdash M:\sigma \mid \Sigma}{\Gamma;\ \Delta \vdash [\alpha]M:\bot \mid \{\alpha:\sigma\}\sharp\Sigma}\ (\bot\text{I}) \qquad \frac{\Gamma;\ \Delta \vdash M:\bot \mid \alpha:\sigma, \Sigma}{\Gamma;\ \Delta \vdash \mu\alpha^\sigma.M:\sigma \mid \Sigma}\ (\bot\text{E}).$$

Axioms

$$\begin{array}{lll} (\boldsymbol{\lambda} x.M) \mathbin{@} N = M[N/x] & & \\ \boldsymbol{\lambda} x.M \mathbin{@} x \quad = M & & (x \notin FV(M)) \\ (\lambda x.M)\,N \quad = M[N/x] & & \\ \lambda x.M\,x \quad\;\; = M & & \\ L\,(\mu\alpha^\sigma.M) = M\left[{}^{L(-)}/_{[\alpha](-)}\right] & & (L:\sigma \multimap \bot) \\ \mu\alpha.[\alpha]M \quad\; = M, & & \end{array}$$

where $M[{}^{L(-)}/_{[\alpha](-)}]$ is obtained by replacing the (unique) subterm of the form $[\alpha]N$ by $L\,N$ in the capture-free way.

Lemma 6.1. The following equations are provable in μDCLL:

— $L\,(\mu\alpha^{\sigma}.M) = \mu\beta^{\tau}.M[{}^{[\beta]L(-)}/_{[\alpha](-)}]$ where $L : \sigma \multimap \tau$
— $[\alpha'](\mu\alpha^{\sigma}.M) = M[\alpha'/\alpha]$
— $\mu\alpha^{\perp}.M = M[{}^{(-)}/_{[\alpha](-)}]$
— $\mu\gamma^{\sigma\to\tau}.M = \boldsymbol{\lambda} x^{\sigma}.\mu\beta^{\tau}.M[{}^{[\beta](-)\,@\,x}/_{[\gamma](-)}]$
— $\mu\gamma^{\sigma\multimap\tau}.M = \lambda x^{\sigma}.\mu\beta^{\tau}.M[{}^{[\beta](-)x}/_{[\gamma](-)}]$.

6.2. *DCLL vs. μDCLL*

We first note that the combinator C_{σ} is easily represented in μDCLL by

$$\mathsf{C}_{\sigma} \;=\; \lambda m^{(\sigma\multimap\perp)\multimap\perp}.\mu\alpha^{\sigma}.m\,(\lambda x^{\sigma}.[\alpha]x) \;:\; ((\sigma \multimap \perp) \multimap \perp) \multimap \sigma.$$

Let us write M° for the induced translation of a DCLL-term M in μDCLL by this encoding.

Lemma 6.2. If $\Gamma;\ \Delta \vdash M : \sigma$ is derivable in DCLL, then $\Gamma;\ \Delta \vdash M^{\circ} : \sigma \mid \emptyset$ is derivable in μDCLL.

Proposition 6.1. If $\Gamma;\ \Delta \vdash M = N : \sigma$ is provable in DCLL, then $\Gamma;\ \Delta \vdash M^{\circ} = N^{\circ} : \sigma \mid \emptyset$ is provable in μDCLL.

Conversely, there is a translation $(-)^{\bullet}$ from μDCLL to DCLL given by

$$\begin{aligned} ([\alpha]M)^{\bullet} &= [\alpha]M^{\bullet} \\ (\mu\alpha^{\sigma}.M)^{\bullet} &= \mathsf{C}_{\sigma}\,(\lambda k.M^{\bullet}\left[{}^{k(-)}/_{[\alpha](-)}\right]), \end{aligned}$$

and so on; for this $(-)^{\bullet}$ we have the following lemma and proposition.

Lemma 6.3. If $\Gamma;\ \Delta \vdash M : \sigma \mid \alpha_1 : \sigma_1, \ldots, \alpha_n : \sigma_n$ is derivable in μDCLL, then $\Gamma;\ \Delta, k_n : \sigma_n \multimap \perp, \ldots, k_1 : \sigma_1 \multimap \perp \vdash M^{\bullet}[{}^{k_1(-)}/_{[\alpha_1](-)}, \ldots, {}^{k_n(-)}/_{[\alpha_n](-)}] : \sigma$ is derivable in DCLL. In particular, if $\Gamma;\ \Delta \vdash M : \sigma \mid \emptyset$ is derivable in μDCLL, then $\Gamma;\ \Delta \vdash M^{\bullet} : \sigma$ is derivable in DCLL.

Proposition 6.2. If $\Gamma;\ \Delta \vdash M = N : \sigma \mid \emptyset$ is provable in μDCLL, then $\Gamma;\ \Delta \vdash M^{\bullet} = N^{\bullet} : \sigma$ is provable in DCLL.

Proposition 6.3. For $\Gamma;\ \Delta \vdash M : \sigma$, we have $\Gamma;\ \Delta \vdash M = M^{\circ\bullet} : \sigma$ in DCLL. For $\Gamma;\ \Delta \vdash M : \sigma \mid \emptyset$, we have $\Gamma;\ \Delta \vdash M = M^{\bullet\circ} : \sigma \mid \emptyset$ in μDCLL.

Thus we conclude that DCLL is identical to the single conclusion-fragment of μDCLL as a typed equational theory.

6.3. *Categorical semantics*

The interpretation of a typing judgement of the form

$$x_1 : \sigma_1, \ldots, x_m : \sigma_m;\ y_1 : \tau_1, \ldots, y_n : \tau_n \vdash M : \sigma \mid \alpha_1 : \theta_1, \ldots, \alpha_k : \theta_k$$

is given as an arrow from $![\![\sigma_1]\!] \otimes \ldots \otimes ![\![\sigma_m]\!] \otimes [\![\tau_1]\!] \otimes \ldots \otimes [\![\tau_n]\!]$ to $[\![\sigma]\!] \parr [\![\theta_1]\!] \parr \ldots \parr [\![\theta_k]\!]$ by routinely extending and modifying the case of DCLL. The soundness and completeness of μDCLL with respect to the same class of categorical models immediately follow.

7. Discussion

7.1. *DCLL as a typed intermediate language*

The design of DCLL is heavily inspired from our experience (and still on-going project) on the study of compiling (mostly call-by-value typed) programming languages into linearly typed (idealised) intermediate languages (Hasegawa 2002a), which was mentioned briefly in the introduction.

In Berdine *et al.* (2001) and Berdine *et al.* (2002) the $\{\rightarrow, \multimap\}$-fragment of DILL (with recursive types) is used as the target language of call-by-value CPS transformations. In Hasegawa (2002a) we extend the idea of *ibid.* to general monadic transformations into a fragment of DILL. The essential idea of this work is that, in programming practice, certain computational effects like continuations are often used *linearly*, and such good (or stylish) usage of computational effects should be explicitly captured by certain linear typing discipline on the compiled codes. In these studies the 'linearly-used continuation monad' $((-) \rightarrow \theta) \multimap \theta$ plays the key role[†]: $\rightarrow$ for continuations and $\multimap$ for the linearity of their passing. Dually, the construction $((-) \multimap \theta) \rightarrow \theta$ plays a similar role for the call-by-name CPS transformation (Hasegawa 2004). The choice of connectives of DCLL then comes to us naturally: $\rightarrow$ and $\multimap$ come first, and we regard the exponential ! as the special case of the linearly-used continuation monad by letting θ be $\bot$: $!\sigma \simeq (!\sigma \multimap \bot) \multimap \bot \simeq (\sigma \rightarrow \bot) \multimap \bot$.

It is also interesting to re-examine the previous work on applying Classical Linear Logic to programming languages with control features (Filinski 1992; Nishizaki 1993) using DCLL: in particular, Filinski's work seems to share several ideas with the design of DCLL – the use of a control operator for expressing the duality is explicitly found in his work.

[†] This is *not* a monad on the term model of DILL; it is a monad on a suitable subcategory of the category of !-coalgebras.

7.2. *Is '!' better than '→'?*

A possible criticism of DCLL is its indirect treatment of the exponentials, which have been regarded as the central feature of Linear Logic by many people (though there are some exceptions, for example, Wadler (1990), Plotkin (1993), Hodas and Miller (1994) and Maietti *et al.* (2000)). We used to consider ! as a primitive and → as a derived connective via Girard's formula $\sigma \to \tau \equiv\, !\sigma \multimap \tau$, but not the converse, that is, $!\sigma \equiv (\sigma \to \bot) \multimap \bot$, as we do in DCLL.

However, even in Intuitionistic Linear Logic, we have the full completeness of the $\{\to, \multimap\}$-fragment in the $\{!, \multimap\}$-fragment, in the following sense. Let $(-)^\circ$ be the embedding from the former into the latter via Girard's translation:

$$\begin{aligned} b^\circ &\equiv b \\ (\sigma_1 \multimap \sigma_2)^\circ &\equiv \sigma_1^\circ \multimap \sigma_2^\circ \\ (\sigma_1 \to \sigma_2)^\circ &\equiv\, !\sigma_1^\circ \multimap \sigma_2^\circ \end{aligned}$$

$$\begin{aligned} x^\circ &\equiv x \\ (\lambda x^\sigma.M)^\circ &\equiv \lambda x^{\sigma^\circ}.M^\circ \\ (M^{\sigma_1 \multimap \sigma_2}\, N^{\sigma_1})^\circ &\equiv M^\circ\, N^\circ \\ (\boldsymbol{\lambda} x^\sigma.M)^\circ &\equiv \lambda y^{!\sigma^\circ}.\mathsf{let}\ !x^{\sigma^\circ}\ \mathsf{be}\ y\ \mathsf{in}\ M^\circ \\ (M^{\sigma_1 \to \sigma_2} \,@\, N^{\sigma_1})^\circ &\equiv M^\circ\,(!N^\circ). \end{aligned}$$

It is not hard to see that $(-)^\circ$ is type-sound (preserves typing), and, also, equationally sound and complete (two terms in the source calculus are equal if and only if their translations are equal in the target). But we can say more (Hasegawa 2002a), as in the following theorem.

Theorem 7.1. Suppose that $\Gamma^\circ; \Delta^\circ \vdash N : \sigma^\circ$ is derivable in the $\{!, \multimap\}$-fragment. Then there exists $\Gamma; \Delta \vdash M : \sigma$ derivable in the $\{\to, \multimap\}$-fragment such that $\Gamma^\circ; \Delta^\circ \vdash M^\circ = N : \sigma^\circ$ holds.

This can be shown by mildly extending the proof of full completeness of Girard's translation from the simply typed lambda calculus into the $\{!, \multimap\}$-fragment of DILL (Hasegawa 2000). This observation tells us that → is no less delicate than ! at the level of proofs (terms), while $\{\to, \multimap\}$ enjoys much simpler term structures and nice properties like confluence and strong normalisation. And, in Classical Linear Logic, as we have demonstrated in this paper, $\{\to, \multimap, \bot\}$ is literally isomorphic to $\{!, \multimap, \bot\}$, which means it is not unnatural to use the technically simpler presentation.

Moreover, as mentioned above, DCLL does have natural advantages in programming language theory. From such an application-oriented view, we think that the simplicity of DCLL is undeniably attractive. See also Maietti *et al.* (2000) for relevant discussions on the $\{\to, \multimap, \otimes, I, \&, \top\}$-fragment and its fibration-based models (which can be adopted for DCLL without problem).

7.3. *Coherence of the double negation*

Another possible source of criticism of DCLL would be the way we deal with the duality, which again is the essential feature of Classical Linear Logic. Many systems for Classical Linear Logic, especially those of proof nets, identify the type $\sigma^{\perp\perp}$ $(= (\sigma \multimap \perp) \multimap \perp)$ with σ by definition. On the other hand, in DCLL (and some other term-based systems like Bierman (1999) and a net-based one (Blute *et al.* 1996)) they are just isomorphic, and we explicitly have terms for the isomorphisms. The essential reason of this non-identification in DCLL is that we intend it to have $*$-autonomous categories with linear exponential comonads as models, rather than those with *strict* involution (that is, $(-)^{\perp\perp}$ is the identity functor and the canonical isomorphism $\sigma \xrightarrow{\simeq} \sigma^{\perp\perp}$ is an identity arrow), as we think that having a strict involution is not a natural assumption on semantic models.

Fortunately, it was shown recently (Cockett *et al.* 2003) that any $*$-autonomous category is equivalent to a $*$-autonomous category with strict involution, and that any free $*$-autonomous category is strictly equivalent to a free $*$-autonomous category with strict involution; and the results remain true under the presence of linear exponential comonads and finite products too. These coherence results indicate that from a technical viewpoint it does not matter whether the double negation is made strict or non-strict: it is safe to transfer the results on up-to-isomorphism systems to up-to-equality systems, and *vice versa*.

Thus, this criticism of DCLL is, at least technically, not very essential: the choice of making the double negation strict is just a matter of convenience and taste.

7.4. *Faithful categorical models*

In this paper we have demonstrated that DCLL is sound and complete with respect to the standard categorical models of Linear Logic ($*$-autonomous categories with additional structure). However, it is via the encoding of constructs like tensor products, which are not included in DCLL as primitive constructs. It is an interesting task to identify the categorical structure that is more 'faithful' to DCLL, that is, that can accommodate the interpretation of linear and non-linear implications without requiring a monoidal structure and a linear exponential comonad. A most promising direction would be the one based on multicategories (Lambek 1989), and perhaps polycategories (Szabo 1975) for μDCLL. The story looks fairly clean as long as we work on the multiplicative fragment (see Hyland's analysis of $*$-autonomous categories and $*$-polycategories (Hyland 2002)), but explaining the dual-context feature seems to call for some subtle technical developments.

7.5. *Second-order linear logic of implications*

We conclude this paper by observing an attractive relationship between DCLL and a second-order linear lambda calculus: they are strikingly similar (at least syntactically), but also show some interesting differences.

In Plotkin (1993), Plotkin introduced the *second-order* $\{\to, \multimap\}$-calculus (enriched with fixed-point operators) in which other connectives of DILL, including !, are definable in a similar way to our approach in DCLL, for example $!\sigma$ as $\forall X.(\sigma \to X) \multimap X$. In fact, it suffices to have the axiom (in addition to the standard $\beta\eta$-axioms)

$$L^{\sigma \multimap \tau}\,(M^{\forall X.(\sigma \multimap X) \multimap X}\,\sigma\,(\lambda x^\sigma.x)) = M\,\tau\,L$$

(which just says σ is canonically isomorphic to $\forall X.(\sigma \multimap X) \multimap X$) to give the structure of models of DILL to the term model of this calculus – the story is completely analogous to the case of DCLL, with the encoding of types and terms given as follows:

$$\begin{array}{ll}
I & = \forall X.X \multimap X \\
\sigma_1 \otimes \sigma_2 & = \forall X.(\sigma_1 \multimap \sigma_2 \multimap X) \multimap X \\
!\sigma & = \forall X.(\sigma \to X) \multimap X \\
 & \\
* & = \Lambda X.\lambda x^X.x \\
\mathsf{let}\ *\ \mathsf{be}\ M^I\ \mathsf{in}\ N^\tau & = M\,\tau\,N \\
M^{\sigma_1} \otimes N^{\sigma_2} & = \Lambda X.\lambda k^{\sigma_1 \multimap \sigma_2 \multimap X}.k\,M\,N \\
\mathsf{let}\ x^{\sigma_1} \otimes y^{\sigma_2}\ \mathsf{be}\ M^{\sigma_1 \otimes \sigma_2}\ \mathsf{in}\ N^\tau & = M\,\tau\,(\lambda x^{\sigma_1}.\lambda y^{\sigma_2}.N) \\
!M^\sigma & = \Lambda X.\lambda h^{\sigma \to X}.h \mathbin{@} M \\
\mathsf{let}\ !x^\sigma\ \mathsf{be}\ M^{!\sigma}\ \mathsf{in}\ N^\tau & = M\,\tau\,(\boldsymbol{\lambda} x^\sigma.N)
\end{array}$$

By a very similar argument to the one given in Section 3 (though the proof is longer), we have the following theorem.

Theorem 7.2. Any equation derivable in DILL is derivable in the second-order $\{\to, \multimap\}$-calculus (with the axiom described above) via this encoding.

However, note that we cannot have the connectives $\bot$, ⅋ and ?, since the presence of any of them would enable us to interpret Classical Linear Logic, while there are models of this calculus that are not a model of Classical Linear Logic (for example, domain theoretic models (Plotkin 1993) and the model based on an operational semantics by Bierman, Pitts and Russo (Bierman *et al.* 2000)). In particular, we do not have $\forall X.(\sigma \multimap X) \to X \simeq ?\sigma$ (in contrast to $(\sigma \multimap \bot) \to \bot \simeq ?\sigma$ in DCLL). In fact, under a suitable parametricity assumption (Plotkin 1993; Bierman *et al.* 2000), we have $\forall X.(\sigma \multimap X) \to X \simeq \sigma$.

Despite the syntactic similarity of the encodings of DILL, we think that these observations suggest that the relationship between the semantic structure of Classical Linear Logic and that of Second-Order Intuitionistic Linear Logic is far from obvious; the full story seems yet to be developed.

Appendix A. Dual intuitionistic linear logic

Types and Terms

$$\begin{array}{rcl}
\sigma & ::= & b \mid I \mid \sigma \otimes \sigma \mid \sigma \multimap \sigma \mid !\sigma \\
M & ::= & x \mid * \mid \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ M \mid M \otimes M \mid \mathsf{let}\ x^\sigma \otimes x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ M \mid \\
 & & \lambda x^\sigma.M \mid M M \mid !M \mid \mathsf{let}\ !x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ M.
\end{array}$$

Typing

$$\frac{}{\Gamma_1, x:\sigma, \Gamma_2;\ \emptyset \vdash x:\sigma}(\text{Int-Ax}) \qquad \frac{}{\Gamma;\ x:\sigma \vdash x:\sigma}(\text{Lin-Ax})$$

$$\frac{}{\Gamma;\emptyset \vdash *:I}(I\,\text{I}) \qquad \frac{\Gamma;\Delta_1 \vdash M:I \quad \Gamma;\Delta_2 \vdash N:\sigma}{\Gamma;\Delta_1\sharp\Delta_2 \vdash \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ N:\sigma}(I\,\text{E})$$

$$\frac{\Gamma;\Delta_1 \vdash M:\sigma_1 \quad \Gamma;\Delta_2 \vdash N:\sigma_2}{\Gamma;\Delta_1\sharp\Delta_2 \vdash M\otimes N:\sigma_1\otimes\sigma_2}(\otimes\,\text{I}) \qquad \frac{\Gamma;\Delta_1 \vdash M:\sigma_1\otimes\sigma_2 \quad \Gamma;\Delta_2, x:\sigma_1, y:\sigma_2 \vdash N:\tau}{\Gamma;\Delta_1\sharp\Delta_2 \vdash \mathsf{let}\ x^{\sigma_1}\otimes y^{\sigma_2}\ \mathsf{be}\ M\ \mathsf{in}\ N:\tau}(\otimes\,\text{E})$$

$$\frac{\Gamma;\Delta, x:\sigma_1 \vdash M:\sigma_2}{\Gamma;\Delta \vdash \lambda x^{\sigma_1}.M:\sigma_1\multimap\sigma_2}(\multimap\text{I}) \qquad \frac{\Gamma;\Delta_1 \vdash M:\sigma_1\multimap\sigma_2 \quad \Gamma;\Delta_2 \vdash N:\sigma_1}{\Gamma;\Delta_1\sharp\Delta_2 \vdash M\,N:\sigma_2}(\multimap\text{E})$$

$$\frac{\Gamma;\emptyset \vdash M:\sigma}{\Gamma;\emptyset \vdash !M:!\sigma}(!\,\text{I}) \qquad \frac{\Gamma;\Delta_1 \vdash M:!\sigma \quad \Gamma, x:\sigma;\Delta_2 \vdash N:\tau}{\Gamma;\Delta_1\sharp\Delta_2 \vdash \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ N:\tau}(!\,\text{E}).$$

Axioms

$$\begin{aligned}
\mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ M &= M & \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ * &= M\\
\mathsf{let}\ x\otimes y\ \mathsf{be}\ M\otimes N\ \mathsf{in}\ L &= L[M/x, N/y] & \mathsf{let}\ x\otimes y\ \mathsf{be}\ M\ \mathsf{in}\ x\otimes y &= M\\
(\lambda x.M)\,N &= M[N/x] & \lambda x.M\,x &= M\\
\mathsf{let}\ !x\ \mathsf{be}\ !M\ \mathsf{in}\ N &= N[M/x] & \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ !x &= M
\end{aligned}$$

$$\begin{aligned}
C[\mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ N] &= \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ C[N]\\
C[\mathsf{let}\ x\otimes y\ \mathsf{be}\ M\ \mathsf{in}\ N] &= \mathsf{let}\ x\otimes y\ \mathsf{be}\ M\ \mathsf{in}\ C[N]\\
C[\mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ N] &= \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ C[N],
\end{aligned}$$

where $C[-]$ is a linear context (no ! binds $[-]$).

Appendix B. Formulation without C

As noted in Section 2, we can formalise DCLL using just lambda terms and five axioms, if there is no base type.

Types and Terms

$$\sigma ::= \sigma \to \sigma \mid \sigma \multimap \sigma \mid \bot \qquad M ::= x \mid \boldsymbol{\lambda} x^{\sigma}.M \mid M \,@\, M \mid \lambda x^{\sigma}.M \mid M\,M.$$

Typing

$$\frac{}{\Gamma_1, x:\sigma, \Gamma_2;\ \emptyset \vdash x:\sigma}\ (\text{Int-Ax}) \qquad \frac{}{\Gamma;\ x:\sigma \vdash x:\sigma}\ (\text{Lin-Ax})$$

$$\frac{\Gamma, x:\sigma_1;\ \Delta \vdash M:\sigma_2}{\Gamma;\ \Delta \vdash \boldsymbol{\lambda} x^{\sigma_1}.M:\sigma_1\to\sigma_2}\ (\to\text{I}) \qquad \frac{\Gamma;\ \Delta \vdash M:\sigma_1\to\sigma_2 \quad \Gamma;\ \emptyset \vdash N:\sigma_1}{\Gamma;\ \Delta \vdash M\,@\,N:\sigma_2}\ (\to\text{E})$$

$$\frac{\Gamma;\ \Delta, x:\sigma_1 \vdash M:\sigma_2}{\Gamma;\ \Delta \vdash \lambda x^{\sigma_1}.M:\sigma_1\multimap\sigma_2}\ (\multimap\text{I}) \qquad \frac{\Gamma;\ \Delta_1 \vdash M:\sigma_1\multimap\sigma_2 \quad \Gamma;\ \Delta_2 \vdash N:\sigma_1}{\Gamma;\ \Delta_1\sharp\Delta_2 \vdash M\,N:\sigma_2}\ (\multimap\text{E}).$$

Axioms

$$\begin{array}{lll}
(\boldsymbol{\lambda} x.M) \mathbin{@} N & = M[N/x] & \\
\boldsymbol{\lambda} x.M \mathbin{@} x & = M & (x \notin FV(M)) \\
(\lambda x.M)\, N & = M[N/x] & \\
\lambda x.M\, x & = M & \\
L\, (\lambda x^{\sigma}.M\, (\lambda f^{\sigma \multimap \bot}.f\, x)) & = M\, L & \left(\begin{array}{l} L : (\sigma \multimap \bot) \multimap \bot \\ M : ((\sigma \multimap \bot) \multimap \bot) \multimap \bot \end{array} \right).
\end{array}$$

Acknowledgements

I am grateful to Hayo Thielecke for drawing my attention to the implicational fragments of Linear Logic. I thank Ryu Hasegawa, Martin Hofmann, Yoshihiko Kakutani, Valeria de Paiva and Alex Simpson for discussions and comments related to this work, and to the anonymous reviewers for helpful comments.

References

Barber, A. (1997) *Linear Type Theories, Semantics and Action Calculi*, Ph.D. Thesis, ECS-LFCS-97-371, University of Edinburgh.

Barber, A. and Plotkin, G. (1997) Dual intuitionistic linear logic. Manuscript. (An earlier version is available as Technical Report ECS-LFCS-96-347, Laboratory for Foundations of Computer Science, University of Edinburgh.)

Barr, M. (1979) ∗-Autonomous Categories. *Springer-Verlag Lecture Notes in Mathematics* **752**.

Barr, M. (1991) ∗-autonomous categories and linear logic. *Mathematical Structures in Computer Science* **1** 159–178.

Benton, P. N. (1995) A mixed linear and non-linear logic: proofs, terms and models (extended abstract). In: Computer Science Logic (CSL'94). *Springer-Verlag Lecture Notes in Computer Science* **933** 121–135.

Berdine, J., O'Hearn, P. W., Reddy, U. S. and Thielecke, H. (2001) Linearly used continuations. In: Proc. ACM SIGPLAN Workshop on Continuations (CW'01). Technical Report No. 545, Computer Science Department, Indiana University, 47–54.

Berdine, J., O'Hearn, P. W., Reddy, U. S. and Thielecke, H. (2002) Linear continuation-passing. *Higher-Order and Symbolic Computation* **15** (2/3) 181–203.

Bierman, G. M. (1995) What is a categorical model of intuitionistic linear logic? In: Proc. Typed Lambda Calculi and Applications (TLCA'95). *Springer-Verlag Lecture Notes in Computer Science* **902** 78–93.

Bierman, G. M. (1999) A classical linear lambda-calculus. *Theoret. Comp. Sci.* **227** (1–2) 43–78.

Bierman, G. M., Pitts, A. M. and Russo, C. V. (2000) Operational properties of Lily, a polymorphic linear lambda calculus with recursion. In: Proc. Higher Order Operational Techniques in Semantics (HOOTS 2000). *Electronic Notes in Theoretical Computer Science* **41**.

Blute, R. F., Cockett, J. R. B., Seely, R. A. G. and Trimble, T. H. (1996) Natural deduction and coherence for weakly distributive categories. *J. Pure Appl. Algebra* **113** (3) 229–296.

Blute, R. F., Cockett, J. R. B. and Seely, R. A. G. (1997) Categories for computation in context and unified logic. *J. Pure Appl. Algebra* **116** 49–98.

Cockett, J. R. B., Hasegawa, M. and Seely. R. A. G. (2003) Coherence of the double involution on ∗-autonomous categories. (Submitted for publication.)

Felleisen, M., Friedman, D. P., Kohlbecker, E. E. and Duba, B. F. (1987) A syntactic theory of sequential control. *Theor. Comput. Sci.* **52** 205–237.

Filinski, A. (1992) Linear continuations. In: *Proc. Principles of Programming Languages (POPL'92)* 27–38.

Führmann, C. and Thielecke, H. (2004) On the call-by-value CPS transform and its semantics. *Inform. and Compt.* **188** 241–283.

Girard, J.-Y. (1987) Linear logic. *Theoret. Comp. Sci.* **50** 1–102.

Hasegawa, M. (1999) Logical predicates for intuitionistic linear type theories. In: Proc. Typed Lambda Calculi and Applications (TLCA'99). *Springer-Verlag Lecture Notes in Computer Science* **1581** 198–213.

Hasegawa, M. (2000) Girard translation and logical predicates. *J. Funct. Programming* **10** (1) 77–89.

Hasegawa, M. (2002) Linearly used effects: monadic and CPS transformations into the linear lambda calculus. In: Proc. Functional and Logic Programming (FLOPS2002). *Springer-Verlag Lecture Notes in Computer Science* **2441** 67–182.

Hasegawa, M. (2002) Classical linear logic of implications. In: Proc. Computer Science Logic (CSL'02). *Springer-Verlag Lecture Notes in Computer Science* **2471** 458–472.

Hasegawa, M. (2004) Semantics of linear continuation-passing in call-by-name. In: Proc. Functional and Logic Programming (FLOPS2004). *Springer-Verlag Lecture Notes in Computer Science* **2998** 229–243.

Hodas, J. S. and Miller, D. (1994) Logic programming in a fragment of intuitionistic linear logic. *Inform. and Comput.* **110** (2) 327–365.

Hyland, M. (2002) Proof theory in the abstract. *Ann. Pure Appl. Logic* **114** 43–78.

Hyland, M. and Schalk, A. (2003) Glueing and orthogonality for models of linear logic. *Theoret. Comp. Sci.* **294** (1/2) 183–231.

Kelly, G. M. and Mac Lane, S. (1971) Coherence in closed categories. *J. Pure Appl. Algebra* **1** (1) 97–140.

Koh, T. W. and Ong, C.-H. L. (1999) Explicit substitution internal languages for autonomous and ∗-autonomous categories. In: Proc. Category Theory and Computer Science (CTCS'99). *Electronic Notes in Theoretical Computer Science* **29**.

Lambek, J. (1989) Multicategories revisited. In: Categories in Computer Science. *AMS Contemporary Mathematics* **92** 217–239.

Maietti, M. E., de Paiva, V. and Ritter, E. (2000) Categorical models for intuitionistic and linear type theory. In: Foundations of Software Science and Computation Structure (FoSSaCS 2000). *Springer-Verlag Lecture Notes in Computer Science* **1784** 223–237.

Murawski, A. S. and Ong, C.-H. L. (1999) Exhausting strategies, Joker games and IMLL with units. In: Proc. Category Theory and Computer Science (CTCS'99). *Electronic Notes in Theoretical Computer Science* **29**.

Nishizaki, S. (1993) Programs with continuations and linear logic. *Science of Computer Programming* **21** (2) 165–190.

Parigot, M. (1992) $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In: Proc. Logic Programming and Automated Reasoning. *Springer-Verlag Lecture Notes in Computer Science* **624** 190–201.

Plotkin, G. (1993) Type theory and recursion (extended abstract). In: *Proc. Logic in Computer Science (LICS'93)* 374.

Seely, R. A. G. (1989) Linear logic, ∗-autonomous categories and cofree coalgebras. In: Categories in Computer Science. *AMS Contemporary Mathematics* **92** 371–389.

Streicher, T. (1999) Denotational completeness revisited. In: Proc. Category Theory and Computer Science (CTCS'99). *Electronic Notes in Theoretical Computer Science* **29**.

Szabo, M. E. (1975) Polycategories. *Comm. Algebra* **3** 663–689.

Wadler, P. (1990) Linear types can change the world! In: *Proc. Programming Concepts and Methods*, North-Holland 561–581.

Wadler, P. (1993) A syntax for linear logic. In: Proc. Mathematical Foundations of Programming Semantics (MFPS'93). *Springer-Verlag Lecture Notes in Computer Science* **802** 513–529.