# *Girard Translation and Logical Predicates*

Masahito Hasegawa
*Research Institute for Mathematical Sciences, Kyoto University*
*Kyoto 606-8502 Japan*
(*e-mail:* hassei@kurims.kyoto-u.ac.jp)

**Abstract**

We present a short proof of a folklore result: the *Girard translation* from the simply typed lambda calculus to the linear lambda calculus is *fully complete*. The proof makes use of a notion of *logical predicates* for intuitionistic linear logic. While the main result is of independent interest, this paper can be read as a tutorial on this proof technique for reasoning about relations between type theories.

## 1 Introduction

There is a celebrated translation from the propositional intuitionistic logic to the propositional (intuitionistic) linear logic called *Girard translation* (Girard, 1987), which decomposes the intuitionistic implication "$\rightarrow$" in terms of the linear implication "$\multimap$" and the modality "!". A substantial work on this translation has been carried out over decades, both from syntactic (proof-theoretic) and semantic (category-theoretic) sides: here we shall mention only (Danos *et al.*, 1995) and (Benton *et al.*, 1993a) as the relevant sources of these two aspects. The soundness and conservativity of Girard translation, not just at the provability (types) level but also the proofs (terms) level, are widely known. In particular, the soundness has a clean semantic explanation in terms of monoidal comonads (or symmetric monoidal adjunctions), see *ibid.* Many attempts to apply linear logic to functional programming are based on these facts; see for instance (Braüner, 1995; Braüner, 1996) for a study of the translation from PCF to its linear variant.

Here we show that Girard translation satisfies a stronger property called *full completeness*, which states that, in addition to the soundness and conservativity, if a type of linear logic is definable in intuitionistic logic via Girard translation then its proofs are also definable in the intuitionistic logic. More precisely, we show the following result for the Girard translation $(-)^\circ$ from the simply typed lambda calculus to the linear lambda calculus (Theorem 5.6; the detailed explanation of notations will be given in Section 2 and 3):

Let $\Gamma$ be a context and $\sigma$ a type of the simply typed lambda calculus, and suppose that $\Gamma^\circ\ ;\ \emptyset \vdash M : \sigma^\circ$ is derivable in the linear lambda calculus. Then there exists $\Gamma \vdash N : \sigma$ derivable in the simply typed lambda calculus such that $\Gamma^\circ\ ;\ \emptyset \vdash M = N^\circ : \sigma^\circ$ holds.

In other words, full completeness ensures that Girard translation involves no "junk", thus strengthens the claim that Girard translation is the canonical translation from intuitionistic logic to linear logic. This seems to be folklore among specialists, though we are not aware of this result explicitly mentioned in the literature.

Our proof (Section 4 and 5) is short, but not as straightforward as those of soundness and conservativity. It involves a notion of *logical predicates* (unary logical relations) for linear logic, which were originally introduced for category-theoretic models of linear logic (Hasegawa, 1999a). The idea of logical relations is not new at all, which goes back to the Tait-Girard methods (Tait, 1967; Girard, 1972) for showing the strong normalization of typed lambda calculi. The applications to the lambda-definability problems have been studied since Plotkin's work (Plotkin, 1980). The logical predicate used in this paper is parameterized; the significance of such *Kripke logical predicates* (logical relations with varying arity) was demonstrated in (Jung and Tiuryn, 1993) as a complete characterization of lambda-definability, see also (Alimohamed, 1995). Technically, the most crucial point in our proof is that we derive such a complete logical predicate for the simply typed lambda calculus from a logical predicate for the linear lambda calculus, via the Girard translation. We will explain this step in detail.

In this paper, we present the proof in an entirely syntactic manner, primarily to make the presentation short, self-contained and also easier to access for wider audience. It is true that the original insight of this proof came to the author when he was studying the categorical approach to logical predicates mentioned as above. However, while the categorical axiomatics provides a more uniform approach to this kind of problems, it also requires several delicate issues on the category-theoretic models, which are far from trivial. We believe that our "specialized" presentation better shows the essential structure of the proof than an explanation involving the category-theoretic details, even for those familiar with category-theoretic models of linear logic. See Section 6 for further discussions, including a sketch of how the proof can be carried out within the categorical framework.

We hope that this paper serves as an accessible tutorial which demonstrates how the traditional idea of logical relations can be applied for reasoning about linear logic, and more generally about the relations between type theories.

## 2 Preliminaries

In this paper we consider the minimal setting for discussing Girard translation, thus that involving only the intuitionistic implication, linear implication and the modality !. We use the standard simply typed lambda calculus as the source language. The target language is the fragment of intuitionistic linear logic with $\multimap$ and !, formulated as a linear lambda calculus below. Our presentation is based on a dual-context type system for intuitionistic linear logic (called DILL) due to Barber and Plotkin (Barber and Plotkin, 1997), but any equivalent theory should work as well – see *ibid.* for an equivalence result with a single-context system in (Benton *et al.*, 1993b), and also with Benton's LNL logic (Benton, 1995; Benton and Wadler, 96).

A set of *base types* ($b$ ranges over them) is fixed throughout this paper.

## *2.1 The Simply Typed Lambda Calculus*

We employ a fairly standard syntax:

**Types and Terms**

$$\begin{array}{rcl} \sigma & ::= & b \mid \sigma \to \sigma \\ M & ::= & x \mid \lambda x^{\sigma}.M \mid MM \end{array}$$

We may omit the type superscripts of the lambda abstraction for ease of presentation.

**Typing**

$$\frac{}{\Gamma_1, x : \sigma, \Gamma_2 \vdash x : \sigma} \qquad \frac{\Gamma, x : \sigma_1 \vdash M : \sigma_2}{\Gamma \vdash \lambda x^{\sigma_1}.M : \sigma_1 \to \sigma_2} \qquad \frac{\Gamma \vdash M : \sigma_1 \to \sigma_2 \quad \Gamma \vdash N : \sigma_1}{\Gamma \vdash MN : \sigma_2}$$

where $\Gamma$ is a context, i.e. a finite list of variables annotated with types, in which a variable occurs at most once. We note that any typing judgement has a unique derivation.

**Axioms**

$$\begin{array}{rcl} (\lambda x.M)N & = & M[N/x] \\ \lambda x.Mx & = & M \quad (x \notin FV(M)) \end{array}$$

We assume usual conditions on variables for avoiding undesirable captures. The equality judgement $\Gamma \vdash M = N : \sigma$, where $\Gamma \vdash M : \sigma$ and $\Gamma \vdash N : \sigma$, is defined as the congruence relation on well-typed terms of the same type under the same context, generated from these axioms.

## *2.2 The Linear Lambda Calculus*

In this formulation of the linear lambda calculus, a typing judgement takes the form $\Gamma \ ; \ \Delta \vdash M : \tau$ in which $\Gamma$ represents an intuitionistic (or additive) context whereas $\Delta$ is a linear (multiplicative) context.

**Types and Terms**

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

**Typing**

$$\frac{}{\Gamma \ ; \ x : \tau \vdash x : \tau} \qquad \frac{}{\Gamma_1, x : \tau, \Gamma_2 \ ; \ \emptyset \vdash x : \tau}$$

$$\frac{\Gamma \ ; \ \Delta, x : \tau_1 \vdash M : \tau_2}{\Gamma \ ; \ \Delta \vdash \lambda x^{\tau_1}.M : \tau_1 \multimap \tau_2} \qquad \frac{\Gamma \ ; \ \Delta_1 \vdash M : \tau_1 \multimap \tau_2 \quad \Gamma \ ; \ \Delta_2 \vdash N : \tau_1}{\Gamma \ ; \ \Delta_1 \sharp \Delta_2 \vdash MN : \tau_2}$$

$$\frac{\Gamma \ ; \ \emptyset \vdash M : \tau}{\Gamma \ ; \ \emptyset \vdash !M : !\tau} \qquad \frac{\Gamma \ ; \ \Delta_1 \vdash M : !\tau_1 \quad \Gamma, x : \tau_1 \ ; \ \Delta_2 \vdash N : \tau_2}{\Gamma \ ; \ \Delta_1 \sharp \Delta_2 \vdash \mathsf{let}\ !x^{\tau_1}\ \mathsf{be}\ M\ \mathsf{in}\ N : \tau_2}$$

where $\Delta_1 \sharp \Delta_2$ is a merge of $\Delta_1$ and $\Delta_2$ (this notation is taken from (Barber and Plotkin, 1997)). Thus, $\Delta_1 \sharp \Delta_2$ represents one of possible merges of $\Delta_1$ and $\Delta_2$ as finite lists. We assume that, when we introduce $\Delta_1 \sharp \Delta_2$, there is no variable occurring both in $\Delta_1$ and in $\Delta_2$. We write $\emptyset$ for the empty context. Again we note that any typing judgement has a unique derivation.

**Axioms**

$$\begin{array}{rcl}(\lambda x.M)N & = & M[N/x] \\ \lambda x.Mx & = & M \\ \mathsf{let}\ !x\ \mathsf{be}\ !M\ \mathsf{in}\ N & = & N[M/x] \\ \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ !x & = & M \\ C[\mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ N] & = & \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ C[N]\end{array}$$

where $C[-]$ is a linear context (no ! binds $[-]$); formally it is generated from the following grammar.

$$C \ ::= \ [-] \mid \lambda x.C \mid CM \mid MC \mid \mathsf{let}\ !x\ \mathsf{be}\ C\ \mathsf{in}\ M \mid \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ C$$

The equality judgement $\Gamma\ ;\ \Delta \vdash M = N : \tau$ is defined in the same way as the case of the simply typed lambda calculus.

**Notations:** We will use $\equiv$ for the syntactic identity on terms, and $\equiv_\alpha$ for the $\alpha$-congruence relation.

As discussed in (Barber and Plotkin, 1997), we can regard our linear lambda calculus as a term-assignment system for a natural deduction presentation of (the !, $\multimap$-fragment of) intuitionistic linear logic. It can be easily shown that one can derive a judgement $x_1 : \tau_1, \ldots, x_m : \tau_m\ ;\ y_1 : \tau'_1, \ldots, y_n : \tau'_n \vdash M : \tau$ iff one can prove $!\tau_1, \ldots, !\tau_m, \tau'_1, \ldots, \tau'_n \vdash \tau$ in the original sequent calculus by Girard.

## 3 Girard Translation

The *Girard translation* from the simply typed lambda calculus to the linear lambda calculus is described as follows.

$$\begin{array}{rcl} b^\circ & = & b \\ (\sigma_1 \to \sigma_2)^\circ & = & !\sigma_1^\circ \multimap \sigma_2^\circ \\ \\ x^\circ & \equiv & x \\ (\lambda x^\sigma.M)^\circ & \equiv & \lambda y^{!\sigma^\circ}.\mathsf{let}\ !x^{\sigma^\circ}\ \mathsf{be}\ y\ \mathsf{in}\ M^\circ \\ (MN)^\circ & \equiv & M^\circ(!N^\circ)\end{array}$$

Note that the translation of lambda abstraction involves a new variable $y$, thus depends on a choice of such variables. This means that some results on this translation are stated only up to $\alpha$-congruence.

*Proposition 3.1* (*type soundness*)

If $\Gamma \vdash M : \sigma$ is derivable in the simply typed lambda calculus, then $\Gamma^\circ \;;\; \emptyset \vdash M^\circ : \sigma^\circ$ is derivable in the linear lambda calculus, where $\Gamma^\circ = x_1 : \sigma_1^\circ, \ldots, x_n : \sigma_n^\circ$ for $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$.

*Proof*
See the derivations

$$\frac{}{\Gamma_1^\circ, x : \sigma^\circ, \Gamma_2^\circ \;;\; \emptyset \vdash x : \sigma^\circ}$$

$$\dfrac{\dfrac{\dfrac{}{\Gamma^\circ \;;\; y : !\sigma_1^\circ \vdash y : !\sigma_1^\circ} \quad \Gamma^\circ, x : \sigma_1^\circ \;;\; \emptyset \vdash M^\circ : \sigma_2^\circ}{\Gamma^\circ \;;\; y : !\sigma_1^\circ \vdash \mathsf{let}\ !x\ \mathsf{be}\ y\ \mathsf{in}\ M^\circ : \sigma_2^\circ}}{\Gamma^\circ \;;\; \emptyset \vdash \lambda y.\mathsf{let}\ !x\ \mathsf{be}\ y\ \mathsf{in}\ M^\circ : !\sigma_1^\circ \multimap \sigma_2^\circ}$$

$$\dfrac{\Gamma^\circ \;;\; \emptyset \vdash M^\circ : !\sigma_1^\circ \multimap \sigma_2^\circ \quad \dfrac{\Gamma^\circ \;;\; \emptyset \vdash N^\circ : \sigma_1^\circ}{\Gamma^\circ \;;\; \emptyset \vdash !N^\circ : !\sigma_1^\circ}}{\Gamma^\circ \;;\; \emptyset \vdash M^\circ(!N^\circ) : \sigma_2^\circ} \qquad \Box$$

*Lemma 3.2*
$M^\circ[N^\circ/x] \equiv_\alpha (M[N/x])^\circ$.

*Proposition 3.3* (*soundness*)
If $\Gamma \vdash M = N : \sigma$ then $\Gamma^\circ \;;\; \emptyset \vdash M^\circ = N^\circ : \sigma^\circ$ holds.

*Proof*
It suffices to see that the axioms are soundly interpreted:

$$\begin{array}{rcl}
((\lambda x.M)N)^\circ & \equiv & (\lambda y.\mathsf{let}\ !x\ \mathsf{be}\ y\ \mathsf{in}\ M^\circ)(!N^\circ) \\
& = & \mathsf{let}\ !x\ \mathsf{be}\ !N^\circ\ \mathsf{in}\ M^\circ \\
& = & M^\circ[N^\circ/x] \\
& = & (M[N/x])^\circ \quad \text{(by Lemma 3.2)}
\end{array}$$

$$\begin{array}{rcl}
(\lambda x.Mx)^\circ & \equiv & \lambda y.\mathsf{let}\ !x\ \mathsf{be}\ y\ \mathsf{in}\ M^\circ !x \\
& = & \lambda y.M^\circ(\mathsf{let}\ !x\ \mathsf{be}\ y\ \mathsf{in}\ !x) \\
& = & \lambda y.M^\circ y \\
& = & M^\circ \qquad \Box
\end{array}$$

*Proposition 3.4* (*conservativity*)
If $\Gamma^\circ \;;\; \emptyset \vdash M^\circ = N^\circ : \sigma^\circ$ then $\Gamma \vdash M = N : \sigma$ holds.

*Proof*
This can be shown by defining an "inverse" $(-)^\bullet$ of $(-)^\circ$ which forgets any information on linearity: $b^\bullet = b$, $(\tau_1 \multimap \tau_2)^\bullet = \tau_1^\bullet \to \tau_2^\bullet$, $(!\tau)^\bullet = \tau^\bullet$ and

$$\begin{array}{rcl}
x^\bullet & \equiv & x \\
(\lambda x^\tau.M)^\bullet & \equiv & \lambda x^{\tau^\bullet}.M^\bullet \\
(MN)^\bullet & \equiv & M^\bullet N^\bullet \\
(!M)^\bullet & \equiv & M^\bullet \\
(\mathsf{let}\ !x^\tau\ \mathsf{be}\ M\ \mathsf{in}\ N)^\bullet & \equiv & N^\bullet[M^\bullet/x]
\end{array}$$

With little efforts one can show that $(\sigma^\circ)^\bullet = \sigma$ and $(M^\circ)^\bullet \equiv_\alpha M$, and also $(-)^\bullet$ satisfies the type soundness and soundness. So we can conclude that $\Gamma^\circ \;;\; \emptyset \vdash M^\circ = N^\circ : \sigma^\circ$ implies $\Gamma \vdash M \equiv_\alpha (M^\circ)^\bullet = (N^\circ)^\bullet \equiv_\alpha N : \sigma$. $\Box$

## 4 Logical Predicates

Let us first sketch our plan. We are going to answer a "lambda-definability problem" of the following form: any term of the linear lambda calculus of a lambda-definable type is lambda-definable via Girard translation (up to the provable equality). Logical predicates are suitable for proving such issues. In particular, from (Jung and Tiuryn, 1993; Alimohamed, 1995), we know that there exist "complete" logical predicates which characterize all definable elements in arbitrary models of the simply typed lambda calculus. These logical predicates are parameterized, for dealing with free variables properly.

Our situation calls for one more twist. Since we are reasoning about a translation between type theories, we also need to relate a logical predicate on the target theory to a logical predicate on the source theory. Suppose that we have a sound translation $\Phi : T_1 \to T_2$ from a type theory $T_1$ to another $T_2$, defined inductively along the type structure of $T_1$. Moreover suppose that we are able to define a notion of logical predicates on $T_2$ by induction along the type structure of $T_2$. We then should be able to show the "Basic Lemma" for logical predicates, that is, any term of $T_2$ satisfies a logical predicate. On the other hand, a logical predicate on $T_2$ induces a logical predicate on $T_1$ via the translation $\Phi$ – a predicate on a $T_1$-type $\sigma$ is given by the predicate on $\Phi(\sigma)$. If the induced logical predicate is complete, i.e., characterizes all $T_1$-definable elements, then we are done; following the Basic Lemma, we know that any elements of $T_2$ of the $T_1$-definable types satisfy the complete logical predicate, thus are definable in $T_1$ via $\Phi$.

The critical step in this proof is to find the logical predicate on $T_2$ so that it induces a complete logical predicate on $T_1$. Fortunately, in the case of Girard translation, this is achieved quite smoothly, as will be shown in the next section.

In the rest of this section, we introduce a notion of parameterized logical predicates and show the Basic Lemma.

**Notations:** We write $\Lambda$ and $\Lambda^\ell$ for the sets of terms of the simply typed lambda calculus and the linear lambda calculus respectively, and use the following notations.

$$\begin{array}{rcl}\Lambda_\sigma(\Gamma) & = & \{M \in \Lambda \mid \Gamma \vdash M : \sigma\} \\ \Lambda^\ell_\tau(\Gamma;\Delta) & = & \{M \in \Lambda^\ell \mid \Gamma \ ; \ \Delta \vdash M : \tau\}\end{array}$$

Also we introduce a partial order $\leq$ on the contexts of the simply typed lambda calculus by $\Gamma_1 \leq \Gamma_2$ iff $\Gamma_2 = \Gamma, \Gamma_1$ for some $\Gamma$.

*Definition 4.1*
Let $\tau$ be a type of the linear lambda calculus. A family $P$ of sets indexed by the contexts of the simply typed lambda calculus is called a $\Lambda$-*predicate on* $\tau$ when $P(\Gamma) \subseteq \Lambda^\ell_\tau(\Gamma^\circ \ ; \ \emptyset)$ and

- **(renaming)** for $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$ and $\Gamma' = y_1 : \sigma_1, \ldots, y_n : \sigma_n$, $M \in P(\Gamma)$ implies $M[y_1/x_1, \ldots, y_n/x_n] \in P(\Gamma')$;
- **(weakening)** $P(\Gamma) \subseteq P(\Gamma')$ if $\Gamma \leq \Gamma'$;
- **(equality)** if $M \in P(\Gamma)$ and $\Gamma^\circ \ ; \ \emptyset \vdash M = M' : \tau$ is provable in the linear lambda calculus, then $M' \in P(\Gamma)$.

The following lemmas determine how to interpret the connectives $\multimap$ and ! on $\Lambda$-predicates.

*Lemma 4.2*
Let $P_1$, $P_2$ be $\Lambda$-predicates on $\tau_1$ and $\tau_2$ respectively. Then there is a $\Lambda$-predicate $P_1 \multimap P_2$ on $\tau_1 \multimap \tau_2$ defined by

$$(P_1 \multimap P_2)(\Gamma) = \\ \{M \in \Lambda^{\ell}_{\tau_1 \multimap \tau_2}(\Gamma^{\circ} \ ; \ \emptyset) \mid N \in P_1(\Gamma') \text{ and } \Gamma \leq \Gamma' \text{ imply } MN \in P_2(\Gamma')\}.$$

*Lemma 4.3*
Let $P$ be a $\Lambda$-predicate on $\tau$. Then we have a $\Lambda$-predicate $!P$ on $!\tau$ given by

$$(!P)(\Gamma) = \{M \in \Lambda^{\ell}_{!\tau}(\Gamma^{\circ} \ ; \ \emptyset) \mid M = !N \text{ for } \exists N \in P(\Gamma)\}.$$

*Definition 4.4*
A family $\{P_\tau\}$ of $\Lambda$-predicates is called a *logical* $\Lambda$*-predicate* when

- $P_\tau$ is a $\Lambda$-predicate on $\tau$; and
- $P_{\tau_1 \multimap \tau_2} = P_{\tau_1} \multimap P_{\tau_2}$ and $P_{!\tau} = !P_\tau$ hold.

*Proposition 4.5* (*Basic Lemma*)
Let $\{P_\tau\}$ be a logical $\Lambda$-predicate. For $M_i \in P_{!\tau_i}(\Gamma)$ $(1 \leq i \leq m)$, $M'_j \in P_{\tau'_j}(\Gamma)$ $(1 \leq j \leq n)$ and $x_1 : \tau_1, \ldots, x_m : \tau_m \ ; \ y_1 : \tau'_1, \ldots, y_n : \tau'_n \vdash N : \tau$, it follows that

$$\mathsf{let}\ !x_1\ \mathsf{be}\ M_1\ \mathsf{in}\ \ldots \mathsf{let}\ !x_m\ \mathsf{be}\ M_m\ \mathsf{in}\ N[M'_1/y_1, \ldots, M'_n/y_n] \in P_\tau(\Gamma).$$

*Proof*
Induction on the derivation of terms (typing judgements). Below we write $\Sigma$ for $x_1 : \tau_1, \ldots, x_m : \tau_m$ and $\Delta$ for $y_1 : \tau'_1, \ldots, y_n : \tau'_n$. Since $P_{!\tau_i} = !P_{\tau_i}$, the statement above is equivalent to the following: for $M_i \in P_{\tau_i}(\Gamma)$ $(1 \leq i \leq m)$, $M'_j \in P_{\tau'_j}(\Gamma)$ $(1 \leq j \leq n)$ and $\Sigma \ ; \ \Delta \vdash N : \tau$, it follows that

$$N[M_1/x_1, \ldots, M_m/x_m, M'_1/y_1, \ldots, M'_n/y_n] \in P_\tau(\Gamma).$$

So we often consider this statement instead of the one above.

1. (The cases of variables) The statement obviously holds.
2. (Introduction of $\multimap$) Consider $\Sigma \ ; \ \Delta \vdash \lambda z.N : \tau''_1 \multimap \tau''_2$. Suppose $M_i \in P_{\tau_i}(\Gamma)$ $(1 \leq i \leq m)$ and $M'_j \in P_{\tau'_j}(\Gamma)$ $(1 \leq j \leq n)$. Then $\lambda z.N[M_i/x_i, M'_j/y_j] \in P_{\tau_1 \multimap \tau''_2}(\Gamma)$ because, for any $L \in P_{\tau''_1}(\Gamma')$ $(\Gamma \leq \Gamma')$,
$$(\lambda z.N[M_i/x_i, M'_j/y_j])L = N[M_i/x_i, M'_j/y_j, L/z] \in P_{\tau''_2}(\Gamma')$$
by the induction hypothesis on $N$.
3. (Elimination of $\multimap$) Consider $\Sigma \ ; \ \Delta \vdash N_1 N_2 : \tau''_2$ derived from $\Sigma \ ; \ \Delta_1 \vdash N_1 : \tau''_1 \multimap \tau''_2$ and $\Sigma \ ; \ \Delta_2 \vdash N_2 : \tau''_1$ (thus $\Delta = \Delta_1 \sharp \Delta_2$). Suppose that $M_i \in P_{\tau_i}(\Gamma)$ $(1 \leq i \leq m)$ and $M'_j \in P_{\tau'_j}(\Gamma)$ $(1 \leq j \leq n)$. Then $(N_1 N_2)[M_i/x_i, M'_j/y_j] \in P_{\tau''_2}(\Gamma)$ because
$$(N_1 N_2)[M_i/x_i, M'_j/y_j] \equiv (N_1[M_i/x_i, M'_j/y_j])(N_2[M_i/x_i, M'_j/y_j])$$
while $N_1[M_i/x_i, M'_j/y_j] \in P_{\tau''_1 \multimap \tau''_2}(\Gamma)$ and $N_2[M_i/x_i, M'_j/y_j] \in P_{\tau''_1}(\Gamma)$ by induction hypotheses.

4. (Introduction of !) Consider $\Sigma$ ; $\emptyset \vdash !N :!\tau$. Suppose that $M_i \in P_{\tau_i}(\Gamma)$ $(1 \leq i \leq m)$. Then $(!N)[M_i/x_i] \equiv !(N[M_i/x_i]) \in P_{!\tau}$ as $N[M_i/x_i] \in P_\tau$ by induction hypothesis.
5. (Elimination of !) Consider $\Sigma$ ; $\Delta \vdash \mathsf{let}\ !z\ \mathsf{be}\ N_1\ \mathsf{in}\ N_2 : \tau_2''$ derived from $\Sigma$ ; $\Delta_1 \vdash N_1 :!\tau_1''$ and $\Sigma, z : \tau_1''$ ; $\Delta_2 \vdash N_2 : \tau_2''$ (thus $\Delta = \Delta_1 \sharp \Delta_2$). Suppose that $M_i \in P_{\tau_i}(\Gamma)$ $(1 \leq i \leq m)$ and $M_j' \in P_{\tau_j'}(\Gamma)$ $(1 \leq j \leq n)$. Then

$$\begin{aligned}&(\mathsf{let}\ !z\ \mathsf{be}\ N_1\ \mathsf{in}\ N_2)[M_i/x_i, M_j'/y_j] \equiv \\ &\quad \mathsf{let}\ !z\ \mathsf{be}\ N_1[M_i/x_i, M_j'/y_j]\ \mathsf{in}\ N_2[M_i/x_i, M_j'/y_j] \in P_{\tau_2''}(\Gamma)\end{aligned}$$

by the induction hypothesis on $N_2$, where $N_1[M_i/x_i, M_j'/y_j] \in P_{!\tau_1''}(\Gamma)$ because of the induction hypothesis on $N_1$. $\square$

Note that for closed terms Basic Lemma takes a simpler from:

*Corollary 4.6* (*Basic Lemma for closed terms*)
Let $\{P_\tau\}$ be a logical $\Lambda$-predicate. Then, for any $\emptyset$ ; $\emptyset \vdash M : \tau$, it follows that $M \in P_\tau(\emptyset)$.

## 5 Proof of Fullness

*Definition 5.1*
For a type $\sigma$ and a context $\Gamma$ of the simply typed lambda calculus, define:

$$\mathbb{P}_\sigma(\Gamma) = \{N \in \Lambda^\ell_{\sigma^\circ}(\Gamma^\circ\ ;\ \emptyset) \mid N = M^\circ \text{ for } \exists M \in \Lambda_\sigma(\Gamma)\}.$$

Thus $\mathbb{P}_\sigma(\Gamma)$ is the set of the terms of linear lambda calculus with type $\sigma^\circ$ and context $\Gamma^\circ$ ; $\emptyset$ which are definable in the simply typed lambda calculus via Girard translation. It is immediate to see that

*Lemma 5.2*
$\mathbb{P}_\sigma$ is a $\Lambda$-predicate on $\sigma^\circ$.

Notice that the family $\{\mathbb{P}_\sigma\}$ is *not* a logical $\Lambda$-predicate, as they are defined only on the types of the form $\sigma^\circ$. However, the following lemma shows that $\{\mathbb{P}_\sigma\}$ can be seen a logical predicate for the simply typed lambda calculus, if we define the arrow-type construction on $\Lambda$-predicates by $P \rightarrow P' = !P \multimap P'$.

*Lemma 5.3*
$\mathbb{P}_{\sigma_1 \rightarrow \sigma_2} = !\mathbb{P}_{\sigma_1} \multimap \mathbb{P}_{\sigma_2}$.

*Proof*
Suppose that $M \in \mathbb{P}_{\sigma_1 \rightarrow \sigma_2}(\Gamma)$, thus $M = M'^\circ$ for some $M' \in \Lambda_{\sigma_1 \rightarrow \sigma_2}(\Gamma)$. Let $N \in !\mathbb{P}_{\sigma_1}(\Gamma')$ for some $\Gamma' \geq \Gamma$; this means that $N = !N'$ for some $N' \in \mathbb{P}_{\sigma_1}(\Gamma')$, therefore $N = !N' = !(N''^\circ)$ for some $N'' \in \Lambda_{\sigma_1}(\Gamma')$. Then $MN = M'^\circ(!(N''^\circ)) \equiv (M'N'')^\circ$, hence $MN \in \mathbb{P}_{\sigma_2}(\Gamma')$. Therefore $M \in (!\mathbb{P}_{\sigma_1} \multimap \mathbb{P}_{\sigma_2})(\Gamma)$.

Conversely, suppose that $M \in (!\mathbb{P}_{\sigma_1} \multimap \mathbb{P}_{\sigma_2})(\Gamma)$. Since $!z \in !\mathbb{P}_{\sigma_1}(z : \sigma_1, \Gamma)$, we have $M(!z) \in \mathbb{P}_{\sigma_2}(z : \sigma_1, \Gamma)$, hence there exists $N \in \Lambda_{\sigma_2}(z : \sigma_1, \Gamma)$ such that

$M(!z) = N^\circ$. Now observe that

$$\begin{array}{lcl} M & = & \lambda y^{!\sigma_1^\circ}.My \\ & = & \lambda y^{!\sigma_1^\circ}.M(\mathsf{let}\ !z^{\sigma_1^\circ}\ \mathsf{be}\ y\ \mathsf{in}\ !z) \\ & = & \lambda y^{!\sigma_1^\circ}.\mathsf{let}\ !z^{\sigma_1^\circ}\ \mathsf{be}\ y\ \mathsf{in}\ M(!z) \\ & = & \lambda y^{!\sigma_1^\circ}.\mathsf{let}\ !z^{\sigma_1^\circ}\ \mathsf{be}\ y\ \mathsf{in}\ N^\circ \\ & \equiv_\alpha & (\lambda z^{\sigma_1}.N)^\circ. \end{array}$$

So we conclude that $M \in \mathbb{P}_{\sigma_1 \to \sigma_2}(\Gamma)$. □

It is instructive to observe that how parameters are used to provide free variables in this proof; if we begin with a notion of logical predicates with no parameters (which completely makes sense), we have only $\mathbb{P}_{\sigma_1 \to \sigma_2} \subseteq !\mathbb{P}_{\sigma_1} \multimap \mathbb{P}_{\sigma_2}$ in general.

As explained at the beginning of the previous section, our goal is to find a logical $\Lambda$-predicate $\{\mathbb{P}^*_\tau\}$ which *induces* $\{\mathbb{P}_\sigma\}$, i.e., $\mathbb{P}^*_{\sigma^\circ}$ agrees with $\mathbb{P}_\sigma$. This is achieved by the following observation, which may be seen the Girard translation on the logical predicates.

*Proposition 5.4*
Consider a family $\{P_\sigma\}$ indexed by the types of the simply typed lambda calculus, such that $P_\sigma$ is a $\Lambda$-predicate on $\sigma^\circ$ and satisfies $P_{\sigma_1 \to \sigma_2} = !P_{\sigma_1} \multimap P_{\sigma_2}$. Then there is a logical $\Lambda$-predicate $\{P^*_\tau\}$ such that $P^*_{\sigma^\circ} = P_\sigma$ holds for any $\sigma$.

*Proof*
Define the logical $\Lambda$-predicate $\{P^*_\tau\}$ by $P^*_b = P_b$, $P^*_{\tau_1 \multimap \tau_2} = P^*_{\tau_1} \multimap P^*_{\tau_2}$ and $P^*_{!\tau} = !P^*_\tau$. We proceed by induction on types. The case of base types trivially holds. For arrow types $\sigma_1 \to \sigma_2$, we have

$$\begin{array}{lcll} P^*_{(\sigma_1 \to \sigma_2)^\circ} & = & P^*_{!\sigma_1^\circ \multimap \sigma_2^\circ} & \\ & = & !P^*_{\sigma_1^\circ} \multimap P^*_{\sigma_2^\circ} & \\ & = & !P_{\sigma_1} \multimap P_{\sigma_2} & \text{(by induction hypothesis)} \\ & = & P_{\sigma_1 \to \sigma_2} & \text{(by assumption on } \{P_\sigma\}) \quad \square \end{array}$$

Putting these observations together, we have

*Corollary 5.5*
Define a logical $\Lambda$-predicate $\{\mathbb{P}^*_\tau\}$ by $\mathbb{P}^*_b = \mathbb{P}_b$, $\mathbb{P}^*_{\tau_1 \multimap \tau_2} = \mathbb{P}^*_{\tau_1} \multimap \mathbb{P}^*_{\tau_2}$ and $\mathbb{P}^*_{!\tau} = !\mathbb{P}^*_\tau$. Then $\mathbb{P}^*_{\sigma^\circ} = \mathbb{P}_\sigma$ holds for any type $\sigma$ of the simply typed lambda calculus.

*Theorem 5.6* (*fullness*)
Let $\Gamma$ be a context and $\sigma$ a type of the simply typed lambda calculus, and suppose that $\Gamma^\circ \,;\, \emptyset \vdash M : \sigma^\circ$ is derivable in the linear lambda calculus. Then there exists $\Gamma \vdash N : \sigma$ derivable in the simply typed lambda calculus such that $\Gamma^\circ \,;\, \emptyset \vdash M = N^\circ : \sigma^\circ$ holds.

*Proof*
We apply the Basic Lemma to the logical $\Lambda$-predicate $\{\mathbb{P}^*_\tau\}$. Suppose that $\Gamma^\circ \,;\, \emptyset \vdash M : \sigma^\circ$ and $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$. Since $x_i \in \mathbb{P}_{\sigma_i}(\Gamma) = \mathbb{P}^*_{\sigma_i^\circ}(\Gamma)$, it follows that $M \equiv M[x_1/x_1, \ldots, x_n/x_n] \in \mathbb{P}^*_{\sigma^\circ}(\Gamma) = \mathbb{P}_\sigma(\Gamma)$. Hence there exists $N \in \Lambda_\sigma(\Gamma)$ such that $M = N^\circ$ is provable in the linear lambda calculus. □

## 6 Discussions

### *6.1 Adding Constants, Axioms, and Other Connectives*

Our proof works well under the presence of constants, where we assume that the linear lambda calculus has, for each constant $c$ of type $\sigma$ of the simply typed lambda calculus, a constant (say $\overline{c}$) of type $\sigma^\circ$. Then Girard translation is extended on constants by $c^\circ \equiv \overline{c}$, and all the results are shown in the same way as done above.

It is also natural to ask if we can have additional axioms (equations) without destroying fullness of Girard translation. As easily seen, this is the case, provided the translation is sound for the additional axioms, i.e. respects the induced congruence relations. On the other hand, its conservativity becomes nontrivial, especially when we have axioms on the terms of types of the linear lambda calculus which are not definable in the simply typed lambda calculus.

One may wish to deal with other connectives, like cartesian products and the singleton type (for both the source and target calculi), and also the tensor products and the unit type (for the linear lambda calculus). We can apply our proof technique for such extensions, without essential change. Alternatively, it is possible to use a more semantics-oriented notion of logical predicates, provided the notion of "models" of the system with such connectives is well-established – see below.

### *6.2 Categorical Logical Predicates*

As noted in the introduction, our logical predicates are a syntactic variant of those introduced in (Hasegawa, 1999a) for category-theoretic models of intuitionistic linear type theories. Since the semantic models of such linear type theories are reasonably well-understood in terms of *symmetric monoidal categories* and related structures, we can obtain a logical predicate using model-construction techniques for such structures.

In particular, it is known that a term model of intuitionistic linear logic (with $I$, $\otimes$, $\multimap$ and !) may be given as a symmetric monoidal adjunction $\mathbf{C} \rightleftarrows \mathbf{D}$ between a cartesian closed category (ccc) $\mathbf{C}$ and a symmetric monoidal closed category $\mathbf{D}$ (Bierman, 1995; Benton, 1995; Benton and Wadler, 96; Barber and Plotkin, 1997), and Girard translation amounts to the ccc-functor from a free ccc (the term model of the simply typed lambda calculus) to $\mathbf{C}$. Then full completeness of Girard translation is no other than the full faithfulness of this functor, which can be shown using the technique of *categorical glueing* (Mitchell and Scedrov, 1992). Using the categorical axiomatics, we can formulate (and generalize) the notion of logical predicates for arbitrary categorical models, and carry out the proof. We can show that any logical predicate on $\mathbf{C}$ can be derived from a logical predicate on $\mathbf{D}$ (though it seems that this step does not follow directly from the axiomatics alone), which subsumes the proof in the last section.

The benefit of this semantics-oriented view is not only the conceptual simplicity (for those familiar with the correspondence between syntax and semantics) but also the elegant proofs which avoid handling (often too complicated) syntax directly. Also, for some connectives (including products) the proofs work smoothly. We think that sums do cause nontrivial difficulties, firstly because the lambda-definability problem with sums itself is a substantial question (see (Fiore and Simpson, 1999) for relevant results), and also because the relation of sums in **C** and in **D** seems less clear.

Let us emphasize that these category-theoretic techniques are available only after establishing the tight correspondence between the syntax and the categorical models. In particular, we have to describe the intended structure explicitly, because the proof makes essential use of the notion of structure-preserving functors. While the categorical axiomatics provides a clean and general understanding of the subject, it requires far more technical delicacy than the type-theoretic language does.

## *6.3 Further Applications, Related Work*

In this work we used parameterized logical predicates for showing Lemma 5.3 (the crucial lemma saying $\mathbb{P}_{\sigma_1 \to \sigma_2} = !\mathbb{P}_{\sigma_1} \multimap \mathbb{P}_{\sigma_2}$) which fails to hold if we use non-parameterized predicates. The role of parameters is the same as that of the parameters of the complete logical predicates of (Alimohamed, 1995; Jung and Tiuryn, 1993). On the other hand, in (Hasegawa, 1999a; Hasegawa, 1999b) we have introduced parameterized logical predicates for overcoming the difficulties arising from the linearity. The key trick there is that the parameters are asked to form not just a poset (as in this paper) but a category with structure (typically a symmetric monoidal category), which provides a rich structure on the category of predicates.

Since the source language of Girard translation is not a linear type theory, we did not need the full expressive power of parameterized logical predicates in *ibid.* which, for example, can be used for showing the full completeness of translations from MILL (multiplicative intuitionistic linear logic) to its extension with ! called DILL (Barber and Plotkin, 1997), from the type theory for action calculi (Barber *et al.*, 1998) to DILL (or Benton's LNL logic (Benton, 1995)), and also from MILL to MLL (multiplicative linear logic). For the last case we use the technique of *double glueing* (Tan, 1997), which generalizes Loader's linear logical predicates (Loader, 1994). In the recent work by Streicher (Streicher, 1999), an application of logical predicates for a definability problem on full classical propositional linear logic is developed.

## Acknowledgements

I thank the anonymous reviewers and Philip Wadler for their helpful suggestions and criticisms.

## References

Alimohamed, M. (1995) A characterization of lambda definability in categorical models of implicit polymorphism. *Theoret. Comp. Sci.* **146**, 5–23.

Barber, A., Gardner, P., Hasegawa, M. and Plotkin, G. (1998) From action calculi to linear logic. In *Computer Science Logic (CSL'97), Lecture Notes in Computer Science* **1414**, 78–97, Springer-Verlag.

Barber, A. and Plotkin, G. (1997) Dual intuitionistic linear logic. Submitted for publication. Available from `http://www.dcs.ed.ac.uk/home/agb/research.html`.

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

Benton, N., Bierman, G.M., de Paiva, V. and Hyland, J.M.E. (1993a) Linear lambda-calculus and categorical models revisited. In *Computer Science Logic (CSL'92), Lecture Notes in Computer Science* **702**, 61–84. Springer-Verlag.

Benton, N., Bierman, G.M., de Paiva, V. and Hyland, J.M.E. (1993b) A term calculus for intuitionistic linear logic. In *Typed Lambda Calculi and Applications (TLCA'93), Lecture Notes in Computer Science* **664**, 75–90. Springer-Verlag.

Benton, N. and Wadler, P. (1996) Linear logic, monads, and the lambda calculus. In *Logic in Computer Science (LICS'96)*, 420–431. IEEE Computer Society Press.

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

Braüner, T. (1995) The Girard translation extended with recursion, In *Computer Science Logic (CSL'94), Lecture Notes in Computer Science* **933**, 31–45. Springer-Verlag.

Braüner, T. (1996) *An Axiomatic Approach to Adequacy.* Ph.D. thesis, University of Aarhus; Technical report BRICS-DS-96-4, BRICS.

Danos, V., Joinet, J.-B. and Schellinx, H. (1995) On the linear decoration of intuitionistic derivations. *Arch. Math. Logic* **33**, 387–412.

Fiore, M. and Simpson, A. (1999) Lambda definability with sums via Grothendieck logical relations. In *Typed Lambda Calculi and Applications (TLCA'99), Lecture Notes in Computer Science* **1581**, 147–161. Springer-Verlag.

Girard, J.-Y. (1972) *Interprétation Fonctionelle et Elimination des Coupures dans L'Alithmétique D'Ordre Supérieur.* Thèse D'Etat, Université Paris VII.

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

Hasegawa, M. (1999a) Logical predicates for intuitionistic linear type theories. In *Typed Lambda Calculi and Applications (TLCA'99), Lecture Notes in Computer Science* **1581**, 198–212. Springer-Verlag.

Hasegawa, M. (1999b) Categorical glueing and logical predicates for models of linear logic. Preprint RIMS-1223, Kyoto University.

Jung, A. and Tiuryn, J. (1993) A new characterisation of lambda definability. In *Typed Lambda Calculi and Applications (TLCA'93), Lecture Notes in Computer Science* **664**, 230–244. Springer-Verlag.

Loader, R. (1994) *Models of Lambda Calculi and Linear Logic: Structural, Equational and Proof-Theoretic Characterisations.* Ph.D. thesis, St. Hugh's College, Oxford.

Mitchell, J.C. and Scedrov, A. (1992) Notes on sconing and relators. In *Computer Science Logic (CSL'92), Lecture Notes in Computer Science* **702**, 352–378. Springer-Verlag.

Plotkin, G.D. (1980) Lambda-definability in the full type hierarchy. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 363–373. Academic Press.

Streicher, T. (1999) Denotational completeness revisited. In *Category Theory and Computer Science (CTCS'99), Electron. Notes Theor. Comput. Sci.*

Tait, W.W. (1967) Intensional interpretation of functionals of finite type I. *J. Symbolic Logic* **32**, 198–212.

Tan, A.M. (1997) *Full Completeness for Models of Linear Logic.* Ph.D. thesis, University of Cambridge.