A Modified GoI Interpretation for a Linear Functional Programming Language and its Adequacy

Naohiko Hoshino

Research Institute for Mathematical Science, Kyoto university
naophiko@kurims.kyoto-u.ac.jp

Abstract. Geometry of Interaction (GoI) introduced by Girard provides a semantics for linear logic and its cut elimination. Several extensions of GoI to programming languages have been proposed, but it is not discussed to what extent they capture behaviour of programs as far as the author knows. In this paper, we study GoI interpretation of a linear functional programming language (LFP). We observe that we can not extend the standard GoI interpretation to an adequate interpretation of LFP, and we propose a new adequate GoI interpretation of LFP by modifying the standard GoI interpretation. We derive the modified interpretation from a realizability model of LFP. We also relate the interpretation of recursion to cyclic computation (the trace operator in the category of sets and partial maps) in the realizability model.

1 Introduction

Geometry of Interaction (GoI) introduced by Girard [11] models linear logic by bidirectional computation which is executed by passing data along edges of a proof net. The main purpose of the original GoI is to capture dynamics of cut-elimination of linear logic (β-reduction of the linear lambda calculus). The original GoI interprets a proof as a matrix of operator algebras, and the *execution formula* captures the cut-elimination process. Since we can describe the computation of GoI by a set of local transition rules, we can implement GoI by token machines [14, 20, 10]. These implementations are free from the notion of substitution, and local transition rules specifies them and we have a graphical presentation of local transition rules. As emphasised in [14], GoI interpretation provides a semantic tool to verify graph reductions.

GoI has some applications to studies of programming languages and lambda calculi such as optimization of reductions [14], a compilation of call-by-name PCF into a low level language [20], full completeness for ML-types in the polymorphic lambda calculus [2] and implicit complexity theory [4]. Among these studies, the following questions are fundamental:

- Can we extend GoI to a semantics of programming languages?
- To what extent does GoI interpretation capture behaviour of programming languages? For example, is GoI adequate for the polymorphic linear functional programming language `lily` in [8]?

There are two approaches to the first question. In [12], Girard gave an algorithm to compute fixed points of linear functions, from which we can calculate fixed points for

intuitionistic functions. The algorithm is similar to the computation of the least fixed points in domain theory. On the other hand, in the GoI interpretation of the call-by-name PCF [20], Mackie interpreted the fixed point operator by means of cyclic computation (Section 2 in [20]). This approach corresponds exactly to the coding of recursion in graph reduction, which is more efficient with respect to the use of memory, see [18]. However it is not discussed whether these interpretations are adequate or not.

Our aim in this paper is to give an adequate GoI interpretation of a linear functional programming language LFP which is essentially equal to `lily`. As we will observe, the standard GoI interpretation does not extend to an adequate interpretation of LFP. The main problem is that the standard GoI interpretations of LFP programs are not strict: for some LFP programs, their standard GoI interpretations do not evaluate their arguments. Therefore, so as to give an adequate GoI interpretation of LFP, we need to modify the standard GoI interpretation by imposing strictness on the interpretations of terms. In this paper, we give a modification of the standard GoI interpretation by extracting realizers from a realizability model of LFP.

Our main contributions are:

- We give a modified GoI interpretation of LFP in Mackie style: we interpret recursion by cyclic computation.
- We prove adequacy of the modified GoI interpretation.

This paper is organized as follows. In Section 2, We describe a linear functional programming language LFP. In Section 3, We recall the standard GoI interpretation and define modified GoI interpretation. In Section 4, we illustrate how we extract the modified GoI interpretation from a categorical realizability model. We capture a fixed point operator in the categorical model by means of the trace operator of the category of sets and partial maps which is given by cyclic computation. In Section 5, We prove adequacy of our modified GoI interpretation.

2 A linear functional programming language LFP and its operational semantics

We describe the syntax of a linear functional programming language LFP. This language is essentially equal to `lily` in [8]. LFP is also a fragment of $\mathrm{PILL}_{\mathrm{Y}}$ in [9].

Type $A := X \mid A \multimap A \mid !A \mid \forall X.A$
Term $M := x \mid M\ M \mid \lambda x^{A}.M \mid \texttt{let}\ !x\ \texttt{be}\ M\ \texttt{in}\ M \mid !M \mid M\ A \mid \Lambda X.M \mid \mu x^{A}.M$

$$\frac{\Theta \vdash \Gamma, A}{\Theta \mid \Gamma; x : A \vdash x : A} \qquad \frac{\Theta \vdash \Gamma, A}{\Theta \mid \Gamma, x : A; - \vdash x : A} \qquad \frac{\Theta \mid \Gamma; \Delta, x : A \vdash M : B}{\Theta \mid \Gamma; \Delta \vdash \lambda x^{A}.M : A \multimap B}$$

$$\frac{\Theta \mid \Gamma; \Delta \vdash M : A \multimap B \quad \Theta \mid \Gamma; \Delta' \vdash N : A}{\Theta \mid \Gamma; \Delta \# \Delta' \vdash M\ N : B} \qquad \frac{\Theta \mid \Gamma, x : A; - \vdash M : A}{\Theta \mid \Gamma; - \vdash \mu x^{A}.M : A}$$

$$\frac{\Theta \mid \Gamma; - \vdash M : A}{\Theta \mid \Gamma; - \vdash !M : !A} \qquad \frac{\Theta \mid \Gamma, x : A; \Delta \vdash M : B \quad \Theta \mid \Gamma; \Delta' \vdash N : !A}{\Theta \mid \Gamma; \Delta \# \Delta' \vdash \texttt{let}\ !x\ \texttt{be}\ N\ \texttt{in}\ M : B}$$

$$\frac{\Theta, X \mid \Gamma; \Delta \vdash M : A \quad \Theta \vdash \Gamma; \Delta}{\Theta \mid \Gamma; \Delta \vdash \Lambda X.M : \forall X.A} \qquad \frac{\Theta \mid \Gamma; \Delta \vdash M : \forall X.A \quad \Theta \vdash B}{\Theta \mid \Gamma; \Delta \vdash M\ B : A[B/X]}$$

where Θ is a finite sequence of type variables, and Γ and Δ are finite sequences of pairs of type variables and types. $\Delta\#\Delta'$ is a *merge* of Δ and Δ' [6]. We inductively define a relation "$\Delta\#\Delta'$ is a merge of finite lists Δ and Δ'" by (i) Δ is a merge of the empty sequence ε and Δ, (ii) Δ is a merge of Δ and ε, (iii) if Δ'' is a merge of Δ and Δ', then $x : A, \Delta''$ is a merge of $x : A, \Delta$ and Δ', (iv) if Δ'' is a merge of Δ and Δ', then $x : A, \Delta''$ is a merge of Δ and $x : A, \Delta'$. We write $|\Gamma|$ and $|\Delta|$ for the length of these sequences. $\Theta \vdash A$ means that A is a well formed type under Θ, and $\Theta \vdash \Gamma; \Delta$ means that every type A in $\Gamma; \Delta$ satisfies $\Theta \vdash A$. We write Ty for the set of closed types, and we write Term(A) for the set of closed terms for $A \in$ Ty. When we write Term(A), we assume that A is a closed type without mentioning it.

Operational semantics for LFP is the standard call-by-name evaluation strategy. For more about operational semantics of LFP, see [8].

$$V := \lambda x^{A}.M \mid \Lambda X.M \mid !M$$

$$\frac{}{V \Downarrow V} \qquad \frac{M \Downarrow \lambda x^{A}.L \quad L[N/x] \Downarrow V}{M\ N \Downarrow V} \qquad \frac{M[\mu x^{A}.M/x] \Downarrow V}{\mu x^{A}.M \Downarrow V}$$

$$\frac{M \Downarrow !L \quad N[L/x] \Downarrow V}{\texttt{let}\ !x\ \texttt{be}\ M\ \texttt{in}\ N \Downarrow V} \qquad \frac{M \Downarrow \Lambda X.N \quad N[A/X] \Downarrow V}{M\ A \Downarrow V}$$

When $M \Downarrow V$ is derivable, we write $M \Downarrow$. Otherwise, we write $M \Uparrow$.

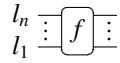
We informally define soundness and adequacy. We define adequacy by termination at !-types. As noted in [8], if we add a type $\mathbb{B}$ of Boolean values then adequacy defined for termination at !-types is equivalent to adequacy defined for termination at $\mathbb{B}$. In Section 3.3, we formally define these notions for the modified GoI interpretation.

Definition 1. *An* interpretation *of* LFP *is a map* $[\![-]\!]$ *that assigns a mathematical object to each term* M*. We suppose that there is a distinguished object* $\bot$ *among the mathematical objects. We say that the interpretation is* sound *when* $M \Downarrow V \Longrightarrow [\![M]\!] = [\![V]\!]$ *for every closed term* M*. We say that the interpretation is* adequate *when the interpretation is sound and* $[\![M]\!] = \bot \iff M \Uparrow$ *for every closed type* A *and a term* M *in* Term(!A).

3 GoI interpretation of LFP

3.1 Graphical presentation of partial maps

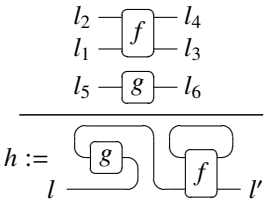
In this section, we give graphical presentation of partial maps, which correspond precisely to string diagrams in the category **Int**(**Pfn**) (see Section 4.3). Let $\mathbb{N}$ be the set of natural numbers. We present a partial map $f : \{l_1, \cdots, l_{n+m}\} \times \mathbb{N} \rightharpoonup \{l_1, \cdots, l_{n+m}\} \times \mathbb{N}$ as a diagram with labeled ports:



When the number of ports on the left hand side is n and the number of ports on the right hand side is m as above, we write $f \in \mathcal{D}(n, m)$. By the definition of $\mathcal{D}(n, m)$, we have $\mathcal{D}(n+1, m) = \mathcal{D}(n, m+1)$ and so on. This equation corresponds to a rearrangement of labeled ports of a diagram. For labels l_i and l_j, we write $f(l_i, n) \simeq (l_j, m)$ or $(l_i, n) \overset{f}{\mapsto} (l_j, m)$ when $f(l_i, n)$ is defined and is equal to (l_j, m). If we do not need to write

labels, especially when a diagram has at most one port, we do not write labels. As a special partial map, we write $\emptyset_{n,m} \in \mathcal{D}(n,m)$ for the partial map whose value is always undefined. When we can infer n and m from context, we omit them.

By combining diagrams, we can construct new partial maps. For example, h given in the following presents a partial map from $\{l, l'\} \times \mathbb{N}$ to $\{l, l'\} \times \mathbb{N}$. The value of h for (l', n) is calculated by the algorithm on the left hand side:



1. Input n to l_3.
2. If we get an output m from l_1 then input m to l_5.
3. If we get an output m from l_2 then input m to l_4.
4. If we get an output m from l_3 then h outputs m from l.
5. If we get an output m from l_4 then input m to l_2.
6. If we get an output m from l_5 then input m to l_1.
7. If we get an output m from l_6 then h outputs m from l.

If there is an infinite loop or no output, then $h(l', n)$ is undefined. The value of $h(l, n)$ is calculated by the same algorithm. Similar argument appears in the composition of strategies in game semantics called "parallel composition + hiding".

3.2 The standard GoI interpretation

For LFP without recursion, we give the standard GoI interpretation, which is a restriction of GoI interpretation of classical linear logic to intuitionistic linear logic. For the GoI interpretation of classical linear logic, see [16].

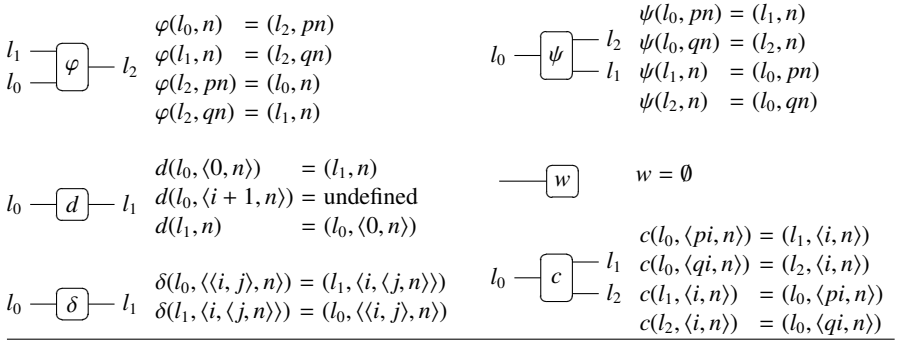


Fig. 1. Components for the standard GoI iterpretation

In the following, we fix two bijections: $\alpha : \mathbb{N}+\mathbb{N} \to \mathbb{N}$ and $\langle -, - \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We define maps $p, q : \mathbb{N} \to \mathbb{N}$ by $\alpha \circ \mathrm{inl}$ and $\alpha \circ \mathrm{inr}$ respectively. For the purpose of this paper, Any infinite set is sufficient, and the choice is a matter of taste. Generalisation of GoI interpretation to Figure 1 is the list of components for the standard GoI interpretation. We use maps p, q and $\langle -, - \rangle$ for tagging. They tell where a natural number came from. An output pn at l_2 of φ means that the input was a number n coming from l_0, and an

output qn at l_2 means that the input was a number n coming from l_1. A number $\langle i, n\rangle$ means the i-th copy of n. By the definition, ψ is a right inverse of φ, which corresponds

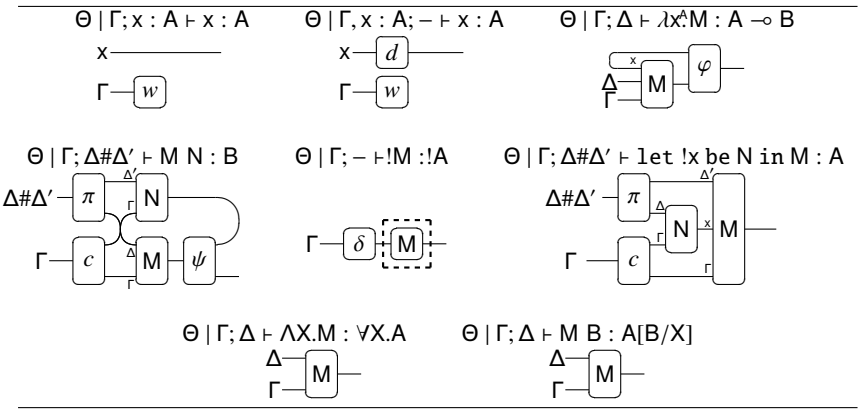
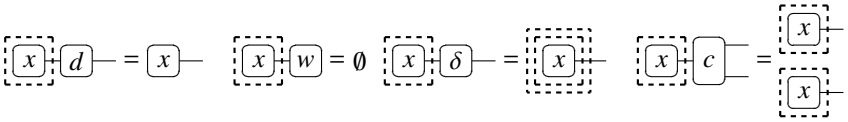


Fig. 2. The standard GoI interpretation of LFP without recursion

to the tensor cell of proof nets [13]. Components d, δ, w and c correspond to dereliction $!A \multimap A$, comultiplication $!A \multimap !!A$, weakening $!A \multimap \mathrm{I}$ and contraction $!A \multimap !A \otimes !A$ of linear logic respectively. In fact, we have



where we define dotted line box, which corresponds to ! of linear logic, as follows.

$$(l, \langle i, n\rangle) \overset{Tf}{\mapsto} (l', \langle j, m\rangle) \overset{\Delta}{\iff} (l, n) \overset{f}{\mapsto} (l', m) \wedge i = j$$

In Figure 2, we define the standard GoI interpretation of LFP without recursion. We interpret each term $\mathsf{\Theta \mid \Gamma; \Delta \vdash M : A}$ by an element in $\mathcal{D}(|\Gamma| + |\Delta|, 1)$. On the left hand side of the interpretation of M, the i-th port counted from the bottom corresponds to the i-th variable in Γ for $1 \le i \le |\Gamma|$, and the $|\Gamma| + j$-th port corresponds to the j-th variable in Δ for $1 \le j \le |\Delta|$. In the definition, πs are the appropriate permutations, and we write diagrams as if $|\Gamma|$ and $|\Delta|$ were 1. We can infer precise definitions from Figure 2.

We can inductively show soundness of the standard GoI interpretation. However, the standard GoI interpretation identifies certain terms that ought to be distinguished: the interpretation of $\mathtt{let\ !x\ be\ M\ in\ !}(\lambda \mathsf{x}^{\mathsf{A}}.\mathsf{x})$ for a term $\mathsf{M} \in \mathrm{Term}(!\mathsf{A})$ is equal to the interpretation of $!(\lambda \mathsf{x}^{\mathsf{A}}.\mathsf{x})$. Because of this equality, the standard GoI interpretation does not extend to an adequate interpretation of the whole LFP. In fact, we have $!(\lambda \mathsf{x}^{\mathsf{A}}.\mathsf{x}) \Downarrow$ as well as $\mathtt{let\ !x\ be}\ \mu \mathsf{x}^{\mathsf{A}}.\mathsf{x}\ \mathtt{in}\ !(\lambda \mathsf{x}^{\mathsf{A}}.\mathsf{x}) \Uparrow$. The reason of this undesirable equality is that the interpretation of a term is not strict on its arguments. This means that the interpretation

of a term does not necessarily evaluate its free variables. If a term $\Theta \mid \Gamma, \mathsf{x} : \mathsf{A}; \Delta \vdash \mathsf{M} : \mathsf{B}$ does not have free x, then the GoI interpretation $[\![\mathsf{M}]\!]$ of M ignores x, i.e. the port on the right hand side of $[\![\mathsf{M}]\!]$ has no path to the port for x on the left hand side of $[\![\mathsf{M}]\!]$.

3.3 Modified GoI interpretation

As noted at the end of previous section, the standard GoI interpretation does not extend to an adequate interpretation of LFP. In this section, in order to obtain adequacy, we modify the standard GoI interpretation and extend it to the whole language. Idea of our modification is to impose strictness on the interpretations of terms.

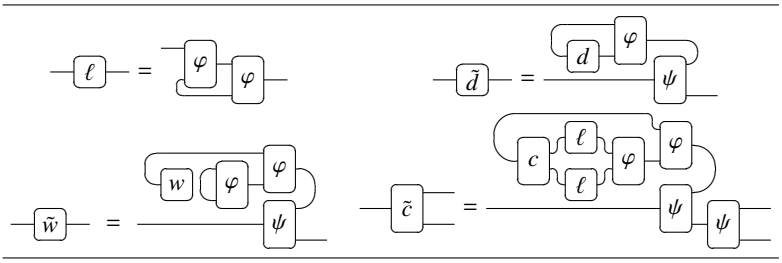
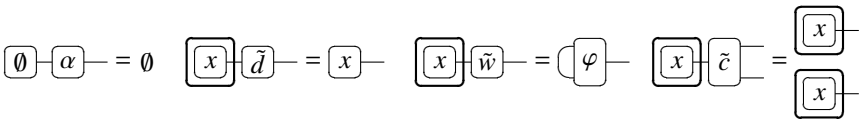


Fig. 3. Basic components for the modified GoI interpretation of LFP

Figure 3 is a list of components for the modified GoI interpretation. $\tilde{d}, \tilde{w}$ and $\tilde{c}$ correspond to d, w and c of the standard GoI interpretation. Note that $\tilde{w} \in \mathcal{D}(1,1)$ whereas $w \in \mathcal{D}(1,0)$. Informally, we "lift" elements $x \in \mathcal{D}(0,1)$ by concatenating ℓ and x. Let α be one of $\tilde{d}, \tilde{w}$ and $\tilde{c}$. Then the concatenation of α and $\emptyset$ is equal to $\emptyset$, and the concatenation of α and a lifting of Tx for $x \in \mathcal{D}(0,1)$ is equal to the following:



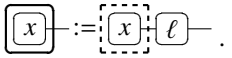
where  . So as to show the equalities, readers are not required to compute the partial functions represented by the above diagrams. We can show the above 4 equalities by means of graph rewriting using the equations noted in Section 3.2.

Figure 4 is the definition of the modified GoI interpretation of LFP. As for the standard GoI interpretation, we interpret a term $\Theta \mid \Gamma; \Delta \vdash \mathsf{M} : \mathsf{A}$ by a partial map in $\mathcal{D}(|\Gamma| + |\Delta|, 1)$, and cells named π in Figure 4 are the appropriate permutations. Note that cyclic computation appears in the interpretation of $\mu \mathsf{x}^{\mathsf{A}}.\mathsf{M}$ which starts from c in front of M going back to δ. We write the modified GoI interpretation of M by $[\![\mathsf{M}]\!]$. The interpretation $[\![\mathsf{M}]\!]$ induces a map from $\mathcal{D}(0,1)^{|\Gamma;\Delta|}$ to $\mathcal{D}(0,1)$, which are strict on their arguments: the value of the map at $(\boldsymbol{x}, \boldsymbol{y})$ is $\emptyset$ when each component of $\boldsymbol{x}$ is $\emptyset$ or surrounded by a thick line box, and at least one of $\boldsymbol{x}$ is $\emptyset$.

We formally define soundness and adequacy for the modified GoI interpretation.

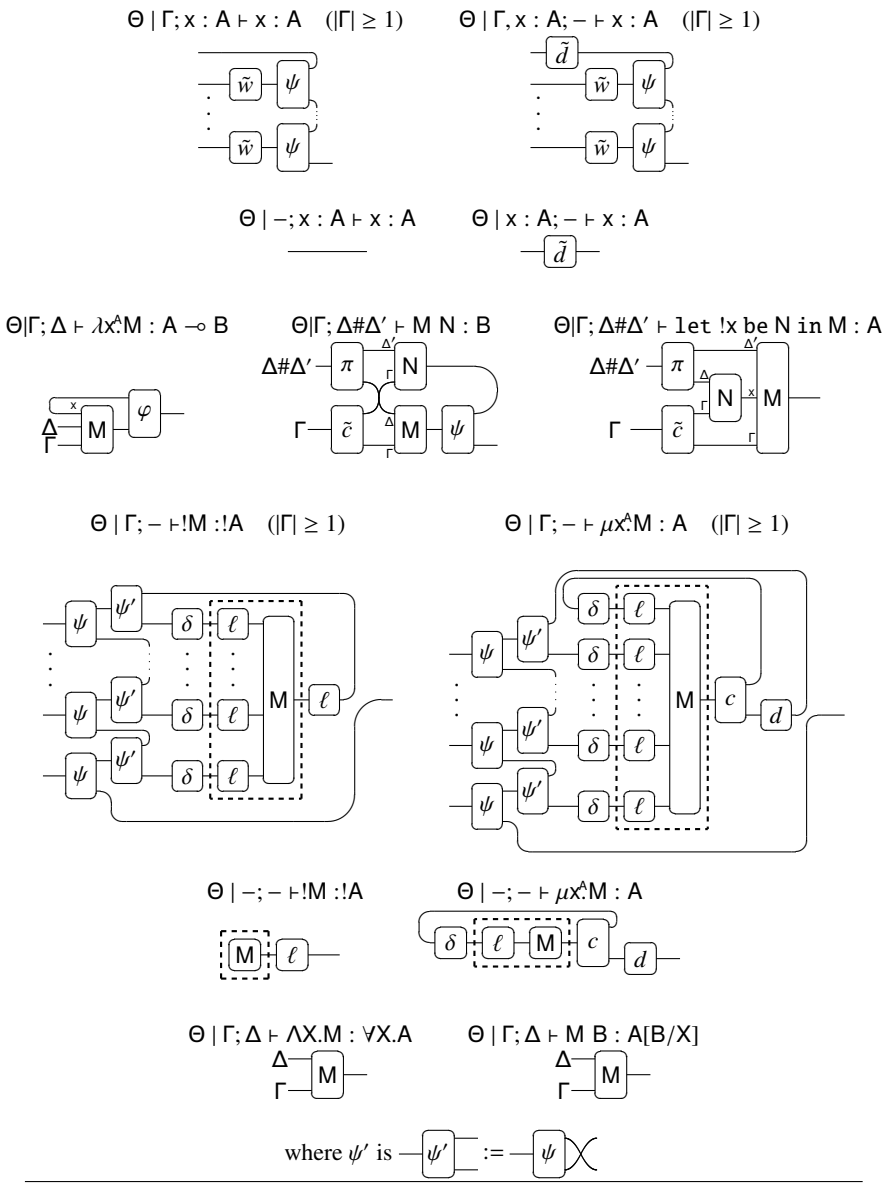


Fig. 4. Modified GoI interpretation of LFP

Definition 2. *We say that the modified GoI interpretation is* sound *when* $\mathsf{M}\Downarrow\mathsf{V}$ *implies* $[\![\mathsf{M}]\!] = [\![\mathsf{V}]\!]$ *for every closed term* M*. We say that the modified GoI interpretation is* adequate *when the interpretation is sound and* $[\![\mathsf{M}]\!] = \emptyset \iff \mathsf{M} \Uparrow$ *for every closed type* A *and a term* M *in* $\mathrm{Term}(!\mathsf{A})$*.*

Theorem 1. *The modified GoI interpretation* $[\![-]\!]$ *is adequate.*

Proof. We will prove this theorem in Section 5.

4 Deriving the modification from a realizability interpretation

Before showing the adequacy of the modified GoI interpretation, we sketch how we derived the modification. Our technique stems from a realizability interpretation. We construct a realizability model of LFP from the ω-cpo $\mathcal{D}(0,1)$. We interpret each LFP term as an equivalence class of a partial equivalence relation (per) on $\mathcal{D}(0,1)$. We can derive our modified GoI interpretation by inductively extracting elements from the equivalence classes. For lack of space, we just outline the construction of the realizability model and illustrate several extractions of the modified GoI interpretation.

4.1 Interpretation of the untyped linear lambda calculus

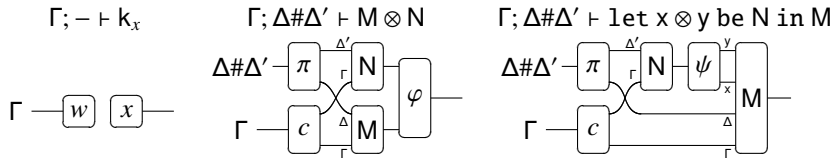
As a preparation, we give a syntax for diagrams representing partial maps. We add the tensor to the untyped linear lambda calculus given in [22]. We need the tensor to describe the monoidal product, which is a fundamental structure of linear categories, of the realizability model that we are going to construct.

$$\mathsf{M} := \mathsf{x} \mid \mathsf{k}_x \mid \lambda\mathsf{x}.\mathsf{M} \mid !\mathsf{M} \mid \mathsf{M}\ \mathsf{M} \mid \mathtt{let}\ !\mathsf{x}\ \mathtt{be}\ \mathsf{M}\ \mathtt{in}\ \mathsf{M} \mid \mathsf{M}\otimes\mathsf{M} \mid \mathtt{let}\ \mathsf{x}\otimes\mathsf{y}\ \mathtt{be}\ \mathsf{M}\ \mathtt{in}\ \mathsf{M}$$

$$\frac{}{\Gamma;\mathsf{x}\vdash\mathsf{x}}\quad\frac{}{\Gamma,\mathsf{x};-\vdash\mathsf{x}}\quad\frac{}{\Gamma;-\vdash\mathsf{k}_x}\quad\frac{\Gamma;\Delta\vdash\mathsf{M}\quad\Gamma;\Delta'\vdash\mathsf{N}}{\Gamma;\Delta\#\Delta'\vdash\mathsf{M}\ \mathsf{N}}\quad\frac{\Gamma;\Delta,\mathsf{x}\vdash\mathsf{M}}{\Gamma;\Delta\vdash\lambda\mathsf{x}.\mathsf{M}}\quad\frac{\Gamma;-\vdash\mathsf{M}}{\Gamma;-\vdash!\mathsf{M}}$$

$$\frac{\Gamma,\mathsf{x};\Delta\vdash\mathsf{M}\quad\Gamma;\Delta'\vdash\mathsf{N}}{\Gamma;\Delta\#\Delta'\vdash\mathtt{let}\ !\mathsf{x}\ \mathtt{be}\ \mathsf{N}\ \mathtt{in}\ \mathsf{M}}\quad\frac{\Gamma;\Delta\vdash\mathsf{M}\quad\Gamma;\Delta'\vdash\mathsf{N}}{\Gamma;\Delta\#\Delta'\vdash\mathsf{M}\otimes\mathsf{N}}\quad\frac{\Gamma;\Delta,\mathsf{x},\mathsf{y}\vdash\mathsf{M}\quad\Gamma;\Delta'\vdash\mathsf{N}}{\Gamma;\Delta\#\Delta'\vdash\mathtt{let}\ \mathsf{x}\otimes\mathsf{y}\ \mathtt{be}\ \mathsf{N}\ \mathtt{in}\ \mathsf{M}}$$

where k_x is a constant symbol, and x runs through $\mathcal{D}(0,1)$. We assign each term to a partial map. For k_x, $\mathsf{M}\otimes\mathsf{N}$ and $\mathtt{let}\ \mathsf{x}\otimes\mathsf{y}\ \mathtt{be}\ \mathsf{N}\ \mathtt{in}\ \mathsf{M}$, assignments are as follows:



For other terms, we can associate partial maps as in Figure 2. For example, the partial map associated to $-;\mathsf{x}\vdash\lambda\mathsf{k}.\mathsf{k}\ \mathsf{x}$ is equal to ℓ in Figure 3. We identify a term and its associated partial map. There are expected equations.

Proposition 1. *As partial maps, the following equations hold.*

$$\begin{aligned}
&(\lambda\mathsf{x}.\mathsf{M})\ \mathsf{N} && = \mathsf{M}[\mathsf{N}/\mathsf{x}] \\
&\mathtt{let}\ !\mathsf{x}\ \mathtt{be}\ !\mathsf{M}\ \mathtt{in}\ \mathsf{P} && = \mathsf{P}[\mathsf{M}/\mathsf{x}] \quad (\mathsf{M}\ \textit{is closed}) \\
&\mathtt{let}\ \mathsf{x}\otimes\mathsf{y}\ \mathtt{be}\ \mathsf{M}\otimes\mathsf{N}\ \mathtt{in}\ \mathsf{P} && = \mathsf{P}[\mathsf{M}/\mathsf{x},\mathsf{N}/\mathsf{y}] \quad (\mathsf{M}\ \textit{and}\ \mathsf{N}\ \textit{are closed})
\end{aligned}$$

4.2 Admissible pers and strict morphisms

We regard $\mathcal{D}(0,1)$ as an ω-cpo by the inclusion order of graph relations of partial maps. The empty map $\emptyset$ is the least element in $\mathcal{D}(0,1)$. We construct a realizability model on the ω-cpo $\mathcal{D}(0,1)$. Like in [9], we consider admissible pers on $\mathcal{D}(0,1)$ which is a subclass of partial equivalence relations.

Definition 3. *For a pointed ω-cpo D, an* admissible per *R on D is a per on D such that $(\bot,\bot) \in R$, and R is closed under least upper bounds (lub) of ω-chains, i.e. for any ascending chain $(x_1,y_1) \le (x_2,y_2) \le \cdots$ in R, the lub $(\bigvee_i x_i, \bigvee_i y_i)$ is also in R. We write $|R|$ for $\{x \mid (x,x) \in R\}$. For $x \in |R|$, we write $[x]_R$ for the equivalence class of x.*

In the following, we simply write x for a closed term k_x. For example, for $x, y \in \mathcal{D}(0,1)$, $x\ y$ means $\mathsf{k}_x\ \mathsf{k}_y$. We use italics for elements in $\mathcal{D}(0,1)$ and sans serifs for syntactic variables of the untyped linear lambda calculus.

Definition 4. *We define a category* **Adm**. *Objects are admissible pers on $\mathcal{D}(0,1)$. For objects X and Y, we define $\sim_{X,Y}$ to be a partial equivalence relation on $\mathcal{D}(0,1)$ such that $r \sim_{X,Y} s$ if and only if $\forall (x,x') \in X.\ (r\ x, s\ x') \in Y$. A morphism $\boldsymbol{f} : X \to Y$ is an equivalence class of $\sim_{X,Y}$. We call a representative r of $\boldsymbol{f}$ a* realizer of $\boldsymbol{f}$*, and we say that r* realizes $\boldsymbol{f}$*. When $\boldsymbol{f}[\emptyset]_X = [\emptyset]_Y$, we say a morphism $\boldsymbol{f} : X \to Y$ is* strict*. We define $\mathbf{Adm}_\bot$ to be a subcategory of* **Adm** *consisting of* **Adm***-objects and strict* **Adm***-morphisms. We use bold face for morphisms in* **Adm** *and* $\mathbf{Adm}_\bot$.

Adm as well as $\mathbf{Adm}_\bot$ is a model of intuitionistic linear logic called *linear category*: A linear category is a symmetric monoidal closed category with a comonad called *linear exponential comonad* [5, 7]. An intuition of **Adm** is the category of ωcpos and continuous maps, and an intuition of $\mathbf{Adm}_\bot$ is the category of pointed ωcpos and strict continuous maps. We list the unit object, the linear implication and the linear exponential comonad of **Adm** and $\mathbf{Adm}_\bot$ respectively.

$$
\begin{aligned}
\mathbf{Adm} \quad \mathrm{I} &= \{(\emptyset,\emptyset)\} \\
X \otimes Y &= \{(x \otimes y, x' \otimes y') \mid (x,x') \in X, (y,y') \in Y\} \\
X \multimap Y &= \{(r,s) \mid \forall (x,x') \in X.\ (r\ x, s\ x') \in Y\} \\
!X &= \{(!x, !x') \mid (x,x') \in X\}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{Adm}_\bot \quad \dot{\mathrm{I}} &= \{(\emptyset,\emptyset)\} \cup \{(\lambda \mathsf{x}.\mathsf{x}, \lambda \mathsf{x}.\mathsf{x})\} \\
X \dot{\otimes} Y &= \{(x \otimes y, x' \otimes y') \mid (x,x') \in X, (y,y') \in Y\} \cup \\
&\qquad \left\{ (x \otimes y, x' \otimes y') \;\middle|\; \begin{array}{l} (x,x' \in |X|) \wedge (y,y' \in |Y|) \wedge \\ (x \in [\emptyset]_X \vee y \in [\emptyset]_Y) \wedge (x' \in [\emptyset]_X \vee y' \in [\emptyset]_Y) \end{array} \right\} \\
X \dot{\multimap} Y &= \{(r,s) \mid \forall (x,x') \in X.\ (r\ x, s\ x') \in Y \wedge r\ \emptyset \in [\emptyset]_Y\} \\
\dot{!}X &= \{(\emptyset,\emptyset)\} \cup \{(\lambda \mathsf{k}.\mathsf{k}\ !x, \lambda \mathsf{k}.\mathsf{k}\ !x') \mid (x,x') \in X\}
\end{aligned}
$$

It is not hard to show that both **Adm** and $\mathbf{Adm}_\bot$ provide models of the polymorphic linear lambda calculus whose linear exponential comonad ! is idempotent, i.e. !! is isomorphic to !. Readers can consult [2, 9]. In [9], Birkedal et al. showed that $\mathbf{Adm}_\bot$ on a certain domain provides a model of the polymorphic linear lambda calculus with recursion, and they discussed relationship between **Adm** and $\mathbf{Adm}_\bot$. We can find a similar relationship for **Adm** and $\mathbf{Adm}_\bot$ on $\mathcal{D}(0,1)$ (Lemma 1).

Extraction of realizers We illustrate extraction of realizers of the dereliction $\dot{\boldsymbol{d}} : {¡}X \to X$ and the weakening $\dot{\boldsymbol{w}} : {¡}X \to \dot{\mathrm{I}}$ of $\mathbf{Adm}_\perp$. They are given by the following:

$$\dot{\boldsymbol{d}}[\emptyset]_{¡X} = [\emptyset]_X \quad \dot{\boldsymbol{d}}[\lambda \mathsf{k}.\mathsf{k}\ !x]_{¡X} = [x]_X \quad \dot{\boldsymbol{w}}[\emptyset]_{¡X} = [\emptyset]_{\mathrm{I}} \quad \dot{\boldsymbol{w}}[\lambda \mathsf{k}.\mathsf{k}\ !x]_{¡X} = [\lambda \mathsf{x}.\mathsf{x}]_{\mathrm{I}}\,.$$

$\dot{\boldsymbol{d}}$ and $\dot{\boldsymbol{w}}$ are realized by $\dot{d}$ and $\dot{w}$ in the following:

$$\dot{d} := \lambda \mathsf{x}.\mathsf{x}\ (\lambda \mathsf{x}'.\mathtt{let}\ !\mathsf{y}\ \mathtt{be}\ \mathsf{x}'\ \mathtt{in}\ \mathsf{y}) \qquad \dot{w} := \lambda \mathsf{x}.\mathsf{x}\ (\lambda \mathsf{x}'.\mathtt{let}\ !\mathsf{y}\ \mathtt{be}\ \mathsf{x}'\ \mathtt{in}\ \lambda \mathsf{z}.\mathsf{z})$$

By uncurrying $\dot{d}$, we obtain $\mathsf{x}\ (\lambda \mathsf{x}'.\mathtt{let}\ !\mathsf{y}\ \mathtt{be}\ \mathsf{x}'\ \mathtt{in}\ \mathsf{y})$, whose corresponding diagram is exactly $\tilde{d}$ in Figure 3. Similarly, we obtain $\tilde{w}$ in Figure 3 by uncurrying $\dot{w}$. Extraction of $\tilde{\delta}$ and the modified GoI interpretation of $!\mathsf{M}$ are more complicated. Still, we can extract realizers by elementary calculation. First, we give a realizer of these terms in a form of untyped terms, then we write down the corresponding diagrams. In the end, by ad hoc optimization, we obtain simpler realizers of these terms which are the modified GoI interpretations of them.

It is not automatic to find $\dot{d}$ and $\dot{w}$. However, if you write down a proof of $\mathbf{Adm}_\perp$ being a linear category, then you must have given an algorithm constructing realizers of $\dot{\boldsymbol{d}}$ and $\dot{\boldsymbol{w}}$ (unless you did not use excluded middle in the proof). Then, you can extract realizers $\dot{d}$ and $\dot{w}$ by the algorithm.

4.3 Interpretation of recursion in Mackie style

We give an interpretation of recursion in $\mathbf{Adm}_\perp$ and illustrate extraction of a realizer of $\mu \mathsf{x}^{\mathsf{A}}.\mathsf{M}$ in Mackie style. At the level of realizers, we interpret recursion by means of the trace operator in $\mathbf{Int}(\mathbf{Pfn})$ (See the following definitions). Before that, we briefly recall several necessary notions: traced symmetric monoidal categories, compact closed categories and $\mathbf{Int}$-construction. For further details, see [19, 17].

Definition 5. *A symmetric monoidal category $\mathbb{C}$ is* traced *when $\mathbb{C}$ has an operator $\mathrm{tr}^A_{X,Y} : \mathbb{C}(X \otimes A, Y \otimes A) \to \mathbb{C}(X, Y)$ subject to*

- $\mathrm{tr}^A_{X',Y'}((k \otimes \mathrm{id}_A) \circ f \circ (h \otimes \mathrm{id}_A)) = k \circ \mathrm{tr}^A_{X,Y}(f) \circ h$
- $\mathrm{tr}^X_{X,X}(\sigma_{X,X}) = \mathrm{id}_X$
- $\mathrm{tr}^A_{Z\otimes X, Z\otimes Y}(\mathrm{id}_Z \otimes f) = \mathrm{id}_Z \otimes \mathrm{tr}^A_{X,Y}(f)$
- $\mathrm{tr}^A_{X,Y}(\mathrm{tr}^B_{X\otimes A, Y\otimes A}(f)) = \mathrm{tr}^B_{X,Y}(\mathrm{tr}^A_{X\otimes B, Y\otimes B}((\mathrm{id}_Y \otimes \sigma_{A,B}) \circ f \circ (\mathrm{id}_X \otimes \sigma_{B,A})))$

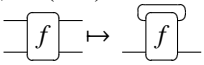
Definition 6. *A* compact closed category $\mathbb{C}$ *is a symmetric monoidal category with a function $* : ob(\mathbb{C}) \to ob(\mathbb{C})$ and morphisms $\epsilon_X : X^* \otimes X \to \mathrm{I}$ and $\eta_X : \mathrm{I} \to X \otimes X^*$ satisfying $(X \otimes \epsilon_X) \circ \alpha \circ (\eta_X \otimes X) = \mathrm{id}_X$ and $(\epsilon_X \otimes X^*) \circ \alpha^{-1} \circ (X^* \otimes \eta_X) = \mathrm{id}_{X^*}$ where $\alpha_{X,Y,Z} : (X \otimes Y) \otimes Z \to X \otimes (Y \otimes Z)$ is the coherence isomorphism.*

Definition 7. *Let $\mathbb{C}$ be a traced symmetric monoidal category $\mathbb{C}$. We define a category $\mathbf{Int}(\mathbb{C})$. Objects are pairs (X_+, X_-) of $\mathbb{C}$-objects, and a morphism $f : (X_+, X_-) \to (Y_+, Y_-)$ is a $\mathbb{C}$-morphism $f : X_+ \otimes Y_- \to Y_+ \otimes X_-$.*

As is known, every compact closed category has a canonical structure of a traced symmetric monoidal category. On the other hand, for a traced symmetric monoidal category $\mathbb{C}$, **Int**($\mathbb{C}$) is a compact closed category whose monoidal product $(X_+, X_-) \otimes (Y_+, Y_-)$ is $(X_+ \otimes Y_+, X_- \otimes Y_-)$.

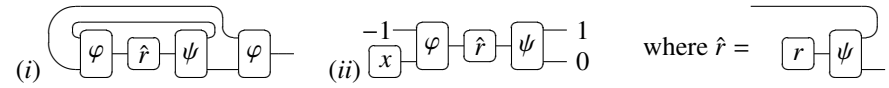
The following theorem shown in [17, 21] relates a trace operator on a linear category to an interpretation of recursion.

Theorem 2. *Let $\mathbb{C}$ be a linear category. If $\mathbb{C}$ is traced, and the linear exponential comonad ! is idempotent, then $\mathbb{C}$ has an operator $(-)^\dagger : \mathbb{C}(!A \otimes !X, X) \to \mathbb{C}(!A, X)$ that satisfies $f = f \circ (!A \otimes (!f^\dagger \circ \delta)) \circ c$ for every $f : !A \otimes !X \to X$ where c is the contraction $!A \to !A \otimes !A$ of $\mathbb{C}$, and δ is the comultiplication $!X \to !!X$ of $\mathbb{C}$.*

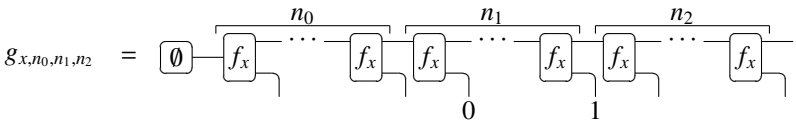
In this paper, we are interested in the traced symmetric monoidal category **Pfn** of sets and partial maps and the compact closed category **Int**(**Pfn**). The monoidal product of **Pfn** is the set theoretic coproduct (i.e. disjoint union), and the trace operator of **Pfn** is given by cyclic computation, see [1]. Diagrams in this paper are exactly string diagrams in **Int**(**Pfn**), and **Int**(**Pfn**)$((\mathbb{N}, \mathbb{N})^{\otimes n}, (\mathbb{N}, \mathbb{N})^{\otimes m})$ is equal to $\mathcal{D}(n, m)$ where $(\mathbb{N}, \mathbb{N})^{\otimes n}$ is the n-fold monoidal product of $(\mathbb{N}, \mathbb{N})$. As noted in the above, **Int**(**Pfn**) is also a traced monoidal category. Its trace operator $\mathrm{tr}^{(\mathbb{N},\mathbb{N})}_{(\mathbb{N},\mathbb{N}),(\mathbb{N},\mathbb{N})}$ is depicted as  . Observe that $\mathrm{tr}^{(\mathbb{N},\mathbb{N})}_{(\mathbb{N},\mathbb{N}),(\mathbb{N},\mathbb{N})}(f)$ is calculated by cyclic computation.

Proposition 2. **Adm** *is a traced symmetric monoidal category.*

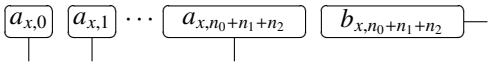
Proof. Let r be a realizer of $\boldsymbol{f} : X \otimes A \to Y \otimes A$. We define $\mathrm{tr}(r) \in \mathcal{D}(0, 1)$ by the diagram (i) below. We show that $\mathrm{tr}(r)$ realizes a morphism from X to Y.



Note that the trace operator $\mathrm{tr}^{(\mathbb{N},\mathbb{N})}_{(\mathbb{N},\mathbb{N}),(\mathbb{N},\mathbb{N})}$ of **Int**(**Pfn**) appears in $\mathrm{tr}(r)$. For $x \in \mathcal{D}(0, 1)$, let $f_x \in \mathcal{D}(1, 2)$ be the diagram (ii) in the above. Then, $n \overset{\mathrm{tr}(r)\ x}{\mapsto} m$ if and only if either $(0, n) \overset{f_x}{\mapsto} (0, m)$, or there exists $k \geq 0$ and $i_0, \cdots, i_k \in \{-1, 1\}$ such that $(0, n) \overset{f_x}{\mapsto} (i_0, x_0)$, $(-i_p, x_p) \overset{f_x}{\mapsto} (i_{p+1}, x_{p+1})$ for $0 \leq p \leq k - 1$ and $(-i_k, x_k) \overset{f_x}{\mapsto} (0, m)$. This is equivalent to the existence of $n_0, n_2 \geq 0$, $n_1 \geq 1$ and $i, j \in \{0, 1\}$ such that the following partial map g_{x,n_0,n_1,n_2} sends (i, n) to (j, m).



Since r is a realizer of $\boldsymbol{f} : X \otimes A \to Y \otimes A$, the partial map g_{x,n_0,n_1,n_2} is equal to



for some $a_{x,n} \in |Y|$ and $b_{x,n} \in |A|$. Note that we have $a_{x,0} \leq a_{x,1} \leq \cdots$. Therefore,

$$\begin{aligned} n \overset{\mathrm{tr}(r)\ x}{\longmapsto} m &\iff \exists n_0, n_1, n_2 \geq 0.\ \exists i, j \in \{0,1\}.\ g_{x,n_0,n_1,n_2}(i,n) \simeq (j,m) \\ &\iff \exists k \geq 0.\ a_{x,k}(n) \simeq m \\ &\iff (\bigvee_{k\geq 0} a_{x,k})(n) \simeq m \end{aligned}$$

By induction on a natural number n, we can show that if $(x, x') \in X$ then $(a_{x,n}, a_{x',n}) \in Y$ and $(b_{x,n}, b_{x',n}) \in A$. Therefore, $(\mathrm{tr}(r)\ x, \mathrm{tr}(r)\ x') = (\bigvee_{n\geq 0} a_{x,n}, \bigvee_{n\geq 0} a_{x',n}) \in Y$, i.e. $\mathrm{tr}(r)$ realizes a morphism from X to Y. Let s be another realizer of $\boldsymbol{f}$, and we define $a'_{x,n}$ and $b'_{x,n}$ in $\mathcal{D}(0,1)$ similarly. Then, we can inductively show that $(a_{x,n}, a'_{x,n}) \in Y$ and $(b_{x,n}, b'_{x,n}) \in A$ for each $x \in |X|$ and $n \geq 0$. Therefore, $\mathrm{tr}(r)$ and $\mathrm{tr}(s)$ realize the same morphism. We define $\mathrm{tr}^A_{X,Y}(\boldsymbol{f})$ to be the morphism realized by $\mathrm{tr}(r)$. The axioms in Definition 5 follow from the fact that **Int**(**Pfn**) is a compact closed category, which is in particular a traced symmetric monoidal category.

Corollary 1. **Adm** *has an operator* $(-)^\dagger : \mathbf{Adm}(!A \otimes !X, X) \to \mathbf{Adm}(!A, X)$ *that satisfies* $\boldsymbol{f} = \boldsymbol{f} \circ (!A \otimes (!\boldsymbol{f}^\dagger \circ \boldsymbol{\delta})) \circ \boldsymbol{c}$ *for every* $\boldsymbol{f} : !A \otimes !X \to X$ *where* $\boldsymbol{c}$ *is the contraction* $!A \to !A \otimes !A$ *of* **Adm***, and* $\boldsymbol{\delta}$ *is the comultiplication* $!X \to !!X$ *of* **Adm**.

Proof. Since **Adm** is traced and the linear exponential comonad of **Adm** is idempotent, the statement holds by Remark 5.1 in [17]. For the proof, see [17, 21].

The following lemma relates **Adm** with $\mathbf{Adm}_\perp$. We omit the proof. Explicitly, LX in the following Lemma is given by $\{(\emptyset, \emptyset)\} \cup \{(\lambda \mathsf{k}.\mathsf{k}\ x, \lambda \mathsf{k}.\mathsf{k}\ y) \mid (x, y) \in X\}$, and the unit $\boldsymbol{h} : X \to ULX$ of the adjunction is given by $\boldsymbol{h}[x]_X = [\lambda \mathsf{k}.\mathsf{k}\ x]_{LX}$. In fact, the linear exponential comonad $\dot{!}$ of $\mathbf{Adm}_\perp$ is the induced linear exponential comonad $L!U$.

Lemma 1. *The forgetful functor* $U : \mathbf{Adm}_\perp \to \mathbf{Adm}$ *has a left adjoint functor* $L : \mathbf{Adm} \to \mathbf{Adm}_\perp$ *which is strong monoidal.*

Proposition 3. $\mathbf{Adm}_\perp$ *has an operator* $(-)^\ddagger : \mathbf{Adm}_\perp(\dot{!}A \dot{\otimes} \dot{!}X, X) \to \mathbf{Adm}_\perp(\dot{!}A, X)$ *that satisfies* $\boldsymbol{f} = \boldsymbol{f} \circ (\dot{!}A \dot{\otimes} (\dot{!}\boldsymbol{f}^\ddagger \circ \dot{\boldsymbol{\delta}})) \circ \dot{\boldsymbol{c}}$ *for every* $\boldsymbol{f} : \dot{!}A \dot{\otimes} \dot{!}X \to X$ *where* $\dot{\boldsymbol{c}}$ *is the contraction* $\dot{!}A \to \dot{!}A \dot{\otimes} \dot{!}A$ *of* $\mathbf{Adm}_\perp$*, and* $\dot{\boldsymbol{\delta}}$ *is the comultiplication* $\dot{!}X \to \dot{!}\dot{!}X$ *of* $\mathbf{Adm}_\perp$.

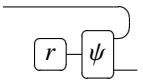
Proof. Let $\boldsymbol{g}$ be the transpose of $L(!UA \otimes !UX) \overset{\cong}{\longrightarrow} L!UA \dot{\otimes} L!UX \overset{f}{\longrightarrow} X$. We define $\boldsymbol{f}^\ddagger : \dot{!}A \to X$ for $\boldsymbol{f} : \dot{!}A \dot{\otimes} \dot{!}X \to X$ to be $L!UA \overset{L(\boldsymbol{g}^\dagger)}{\longrightarrow} LUX \longrightarrow X$. By diagram chasing, we can check $\boldsymbol{f} = \boldsymbol{f} \circ (\dot{!}A \dot{\otimes} (\dot{!}\boldsymbol{f}^\ddagger \circ \dot{\boldsymbol{\delta}})) \circ \dot{\boldsymbol{c}}$.

By Proposition 3, we can interpret recursion in $\mathbf{Adm}_\perp$ by $(-)^\ddagger$. As in the proof, $\ddagger$ is constructed from $\dagger$, and $\dagger$ is constructed from the trace operator of **Adm**. Since the trace of **Adm** is calculated by taking trace of a realizer of $\boldsymbol{f}$ in **Int**(**Pfn**), the trace operator of **Int**(**Pfn**) (cyclic computation) appears in the interpretation of recursion.

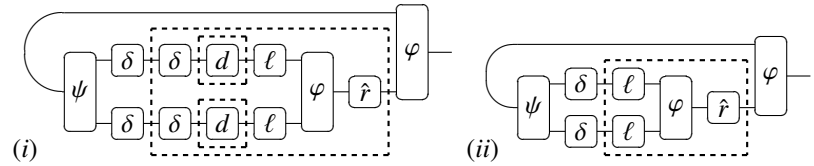
Extraction of realizers We calculate a realizer of $\boldsymbol{f}^\ddagger : \dot{!}A \to X$ for $\boldsymbol{f} : \dot{!}A \dot{\otimes} \dot{!}X \to X$. We take a realizer r of $\boldsymbol{f}$. Let $\boldsymbol{g} : !UA \otimes !UX \to UX$ be an **Adm**-morphism obtained by the transpose of $L(!UA \otimes !UX) \overset{\cong}{\longrightarrow} L!UA \dot{\otimes} L!UX \overset{f}{\longrightarrow} X$. We define h and M by:

$$h := \ !UA \otimes !UX \longrightarrow !!UA \otimes !!UX \longrightarrow !(!UA \otimes !UX) \overset{!\boldsymbol{g}}{\longrightarrow} !UX$$

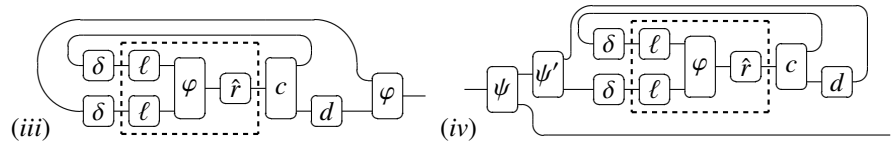
$\mathsf{M} := \lambda \mathsf{x}.\mathtt{let}\ \mathsf{p} \otimes \mathsf{q}\ \mathtt{be}\ \mathsf{x}\ \mathtt{in}\ \mathtt{let}\ !\mathsf{s}\ \mathtt{be}\ \mathsf{p}\ \mathtt{in}\ \mathtt{let}\ !\mathsf{t}\ \mathtt{be}\ \mathsf{q}\ \mathtt{in}\ !(r\ ((\lambda \mathsf{k}.\mathsf{k}\ !\mathsf{s}) \otimes (\lambda \mathsf{k}.\mathsf{k}\ !\mathsf{t})))$.

Then M realizes h. The diagram of M is (*i*), where $\hat{r}$ is 

.



Since the diagrams (*i*) and (*ii*) realize the same morphism, we can optimize (*i*) by (*ii*). Then $\boldsymbol{g}^{\dagger}$ is realized by the following diagram (*iii*).



We write s for the diagram (*iii*). Then $\lambda \mathsf{k}.\mathsf{k}\ s$ realizes $\boldsymbol{f}^{\dagger}$. By uncurrying the diagram of $\lambda \mathsf{k}.\mathsf{k}\ s$ and arranging the uncurried diagram, we obtain a diagram (*iv*) in the above. By replacing $\hat{r}$ and φ in the dotted line box with the modified GoI interpretation of M, we obtain the modified GoI interpretation of $\mu \mathsf{x}^{\wedge}.\mathsf{M}$ in Figure 4 for $|\Gamma| = 1$.

5 Proof of soundness and adequacy

We show Theorem 1 by means of logical relations. Theorem 1 follows from Proposition 4 and Proposition 5 in this section.

For a closed type A, we define $\mathcal{R}(\mathsf{A})$ to be the set of relations R between $\mathcal{D}(0,1)$ and $\mathrm{Term}(\mathsf{A})$ such that

- $\forall \mathsf{M} \in \mathrm{Term}(\mathsf{A}).\ (\emptyset, \mathsf{M}) \in R$.
- If $(x_i, \mathsf{M}) \in R$ for an ascending chain $\{x_i\}_{i \in \mathbb{N}}$, then $(\bigvee_i x_i, \mathsf{M}) \in R$.
- If $(x, \mathsf{M}) \in R$ and $\mathsf{M} \Downarrow \mathsf{V} \iff \mathsf{N} \Downarrow \mathsf{V}$, then $(x, \mathsf{N}) \in R$.

For $\tau : \Theta \to \mathrm{Ty}$ and $\{\rho(\mathsf{X}) \in \mathcal{R}(\tau(\mathsf{X}))\}_{\mathsf{X} \in \Theta}$, we define $\langle \mathsf{A} \rangle_{\tau,\rho} \in \mathcal{R}(\mathsf{A}[\tau(\Theta)/\Theta])$ for $\Theta \vdash \mathsf{A}$:

$$
\begin{aligned}
\langle \mathsf{X} \rangle_{\tau,\rho} &= \rho(\mathsf{X}) \\
\langle \mathsf{A} \multimap \mathsf{B} \rangle_{\tau,\rho} &= \{(x, \mathsf{M}) \mid \forall (y, \mathsf{N}) \in \langle \mathsf{A} \rangle_{\tau,\rho}.\ (x\ y, \mathsf{M}\ \mathsf{N}) \in \langle \mathsf{B} \rangle_{\tau,\rho}\} \\
\langle !\mathsf{A} \rangle_{\tau,\rho} &= \begin{array}{l} \{(\emptyset, \mathsf{M}) \mid \mathsf{M} \in \mathrm{Term}(!\mathsf{A}[\tau(\Theta)/\Theta])\} \\ \cup \{(\lambda \mathsf{k}.\mathsf{k}\ !x, \mathsf{N}) \mid \mathsf{N} \Downarrow !\mathsf{M} \wedge (x, \mathsf{M}) \in \langle \mathsf{A} \rangle_{\tau,\rho}\} \end{array} \\
\langle \forall \mathsf{X}.\mathsf{A} \rangle_{\tau,\rho} &= \textstyle\bigcap_{\mathsf{B} \in \mathrm{Ty}, R \in \mathcal{R}(\mathsf{B})} \{(x, \mathsf{M}) \mid (x, \mathsf{N}\ \mathsf{B}) \in \langle \mathsf{A} \rangle_{\tau[\mathsf{B}/\mathsf{X}], \rho[R/\mathsf{X}]}\}
\end{aligned}
$$

For $\Theta \mid \Gamma; \Delta \vdash \mathsf{M} : \mathsf{A}$, since $[\![\mathsf{M}]\!] \in \mathcal{D}(|\Gamma| + |\Delta|, 1)$, we have a map from $\mathcal{D}(0,1)^{|\Gamma|} \times \mathcal{D}(0,1)^{|\Delta|}$ to $\mathcal{D}(0,1)$. We write its value at $(\boldsymbol{x}, \boldsymbol{y})$ by $[\![\mathsf{M}]\!](\boldsymbol{x}, \boldsymbol{y})$.

Lemma 2. *For* $\Theta \mid \Gamma; \Delta \vdash \mathsf{M} : \mathsf{A}$ *and* $\mathbf{P} \in \mathrm{Term}(\Gamma)$ *and* $\mathbf{Q} \in \mathrm{Term}(\Delta)$*, we have* $[\![\mathsf{M}]\!]\,([\![!\mathbf{P}]\!], [\![\mathbf{Q}]\!]) = [\![\mathsf{M}[\mathbf{P}/\Gamma, \mathbf{Q}/\Delta]]\!]$.

Proposition 4 (Soundness). *If* $\mathsf{M} \Downarrow \mathsf{V}$ *then* $[\![\mathsf{M}]\!] = [\![\mathsf{V}]\!]$.

Proof. By induction on the derivation of $\mathsf{M} \Downarrow \mathsf{V}$. In the induction step of $\mu \mathsf{x}^{\mathsf{A}}.\mathsf{M}$, we can show that the modified GoI interpretation is equal to the least fixed point of a continuous map on $\mathcal{D}(0,1)$ by an argument similar to the proof of Proposition 2. Then, the induction step of $\mu \mathsf{x}^{\mathsf{A}}.\mathsf{M}$ follows from the induction hypothesis. Other cases follow from Lemma 2.

Proposition 5. *For a term* $\Theta \mid \Gamma; \Delta \vdash \mathsf{M} : \mathsf{A}$*, a map* $\tau : \Theta \to \mathrm{Ty}$*, relations* $\rho(\mathsf{X}) \in \mathcal{R}(\tau(\mathsf{X}))$ *for* $\mathsf{X} \in \Theta$*, pairs* $(\xi_0(\mathsf{x}), \xi_1(\mathsf{x})) \in \langle !\mathsf{B} \rangle_{\tau,\rho}$ *for* $(\mathsf{x} : \mathsf{B}) \in \Gamma$ *and pairs* $(\delta_0(\mathsf{x}), \delta_1(\mathsf{x})) \in \langle \mathsf{C} \rangle_{\tau,\rho}$ *for* $(\mathsf{x} : \mathsf{C}) \in \Delta$*, the following holds:*

- *If* $\xi_1(\mathsf{x}) \Downarrow !\xi_1'(\mathsf{x})$ *for every* $(\mathsf{x} : \mathsf{B})$ *in* Γ *then*

$$([\![\mathsf{M}]\!](\xi_0(\Gamma), \delta_0(\Delta)), \mathsf{M}[\tau(\Theta)/\Theta, \xi_1'(\Gamma)/\Gamma, \delta_1(\Delta)/\Delta]) \in \langle \mathsf{A} \rangle_{\tau,\rho}.$$

- *If there exists* $\mathsf{x} : \mathsf{B}$ *in* Γ *such that* $\xi_1(\mathsf{x}) \Uparrow$ *then* $[\![\mathsf{M}]\!](\xi_0(\Gamma), \delta_0(\Delta)) = \emptyset$.

Proof. By induction on M. In the induction step of $\mu \mathsf{x}^{\mathsf{A}}.\mathsf{M}$, we argue as in the proof of Proposition 2. We can calculate the interpretation of $\mu \mathsf{x}^{\mathsf{A}}.\mathsf{M}$ by the lub of an ω-chain. Then, the induction step of $\mu \mathsf{x}^{\mathsf{A}}.\mathsf{M}$ follows from the closedness of $\langle \mathsf{A} \rangle_{\tau,\rho}$ under lubs. We can directly show the other cases by the induction hypothesis.

Related works The GoI interpretation in this paper is based on the traced symmetric monoidal category **Pfn**. In [1], Abramsky, Haghverdi and Scott captured a categorical background of GoI interpretation by *GoI situation*. In [15], Haghverdi introduced *unique decomposition category* so as to capture the original GoI interpretation. The standard GoI interpretation in this paper is based on a GoI situation on the unique decomposition category **Pfn**.

We can describe our modified GoI interpretation by data of a GoI situation on the traced symmetric monoidal category **Pfn**, and we used the unique decomposition structure of **Pfn** to show that **Adm** is traced. However, we do not know to which class of unique decomposition categories we can generalise our argument. Since we used the ω-cpo structure on $\mathcal{D}(0,1) = \mathbf{Pfn}(\mathbb{N}, \mathbb{N})$ in the definition of **Adm** and the proof of Proposition 2, we might need to require some additional domain theoretic structures on a unique decomposition category. We also want to explore relationship between trace operators on linear categories and fixed point operators on cartesian closed categories from our concrete example.

At this point, we do not know how our modified GoI interpretation and the category **Adm** are related to the combinatory algebra called $\mathcal{B}$ in [23] and call-by-value game semantics [3].

Acknowledgement I thank Masahito Hasegawa and Sin-ya Katsumata for discussions and advice. The author is supported by JSPS Research Fellowships for Young Scientists.

References

1. Samson Abramsky, Esfandiar Haghverdi, and Philip J. Scott. Geometry of interaction and linear combinatory algebras. *Math. Struct. in Comput. Sci.*, 12(5):625–665, 2002.

2. Samson Abramsky and Marina Lenisa. A fully complete per model for ml polymorphic types. In Peter Clote and Helmut Schwichtenberg, editors, *CSL*, volume 1862 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 2000.
3. Samson Abramsky and Guy McCusker. Call-by-value games. In *Selected Papers from the11th International Workshop on Computer Science Logic*, CSL '97, pages 1–17, London, UK, 1998. Springer-Verlag.
4. Patrick Baillot and Marco Pedicini. Elementary complexity and geometry of interaction. *Fundam. Inf.*, 45(1-2):1–31, 2001.
5. Andrew Barber and Gordon Plotkin. Dual intuitionistic linear logic, 1997. Technical Reprot ECS-LFCS-96-347, LFCS, Universit of Edinburgh.
6. Andrew Graham Barber. *Linear Type Theories, Semantics and Action Calculi*. PhD thesis, University of Edinburgh, 1997.
7. Gavin M. Bierman. What is a categorical model of intuitionistic linear logic? In *TLCA '95: Proceedings of the Second International Conference on Typed Lambda Calculi and Applications*, pages 78–93, London, UK, 1995. Springer-Verlag.
8. Gavin M. Bierman, Andrew M. Pitts, and Claudio V. Russo. Operational properties of lily, a polymorphic linear lambda calculus with recursion. *Electr. Notes Theor. Comput. Sci.*, 41(3), 2000.
9. Lars Birkedal, Rasmus E. Møgelberg, and Rasmus L. Petersen. Domain-theoretical models of parametric polymorphism. *Theor. Comput. Sci.*, 388(1-3):152–172, 2007.
10. Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal lambda-machines. *Electr. Notes Theor. Comput. Sci.*, 3, 1996.
11. Jean-Yves Girard. Geometry of Interaction I: Interpretation of System F. In R. Ferro et al., editor, *Logic Colloquium '88*. North-Holland, 1989.
12. Jean-Yves Girard. Geometry of Interaction II: Deadlock-free Algorithms. In *Conference on Computer Logic '88*, volume 417 of *LNCS*, pages 76–93. Springer, 1990.
13. Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.
14. Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In *POPL*, pages 15–26, 1992.
15. Esfandiar Haghverdi. *A Categorical Approach to Linear Logic, Geometry of Proofs and Full Completeness*. PhD thesis, University of Ottawa, 2000.
16. Esfandiar Haghverdi and Philip J. Scott. A categorical model for the geometry of interaction. *Theor. Comput. Sci.*, 350(2-3):252–274, 2006.
17. Masahito Hasegawa. On traced monoidal closed categories. *Mathematical Structures in Computer Science*, 19(2):217–244, 2009.
18. Simon P. Jones. *The Implementation of Functional Programming Languages*. Prentice Hall, 1987.
19. André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.
20. Ian Mackie. The geometry of interaction machine. In *POPL*, pages 198–208, 1995.
21. Paul-André Melliès. Functorial boxes in string diagrams. In Zoltán Ésik, editor, *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 1–30. Springer, 2006.
22. Alex K. Simpson. Reduction in a linear lambda-calculus with applications to operational semantics. In Jürgen Giesl, editor, *RTA*, volume 3467 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2005.
23. Jaap van Oosten. A combinatory algebra for sequential functionals of finite type. In S.B. Cooper and J.K. Truss, editors, *Models and Computatbility*. Cambridge University Press, 1999.