Internal Models of Linear Type Theories

Sam Speight and Niels van der Weide

This talk in 1 slide

- This talk is about the **semantics of the linear λ-calculus**
- The starting point is that there are many variations of linear λ-calculus (linear λ-calculus, linear system F, linear dependent type theory, ...)

This talk in 1 slide

- This talk is about the **semantics of the linear λ-calculus**
- The starting point is that there are many variations of linear λ-calculus (linear λ-calculus, linear system F, linear dependent type theory, ...)
- Each of these notions come with their notion of model
- These notions of models are ad-hoc extensions of LNL-models

This talk in 1 slide

- This talk is about the **semantics of the linear λ-calculus**
- The starting point is that there are many variations of linear λ-calculus (linear λ-calculus, linear system F, linear dependent type theory, ...)
- Each of these notions come with their notion of model
- These notions of models are ad-hoc extensions of LNL-models
- Our goal is to provide **a general framework for models of linear λ-calculi**

Linear Non-Linear Models

An **LNL model** is a symmetric monoidal adjunction as follows[1].

$$\begin{tikzcd} \mathcal{L} \arrow[rr, bend right, "\mathsf{M}"'] \arrow[rr, phantom, "\perp"] & & \mathcal{M} \arrow[ll, bend right, "\mathsf{L}"'] \end{tikzcd}$$

Here:

- $\mathcal{L}$ is a symmetric monoidal category
- $\mathcal{M}$ is a Cartesian monoidal category

LNL models give semantics to the linear simply typed λ-calculus

[1] "A Mixed Linear and Non-Linear Logic: Proofs, Terms and Models", Nick Benton

Linear Non-Linear Models

An **LNL model** is a symmetric monoidal adjunction as follows[1].

$$\begin{tikzcd} \mathcal{L} \arrow[r, bend right, "M"'] \arrow[r, phantom, "\perp"] & \mathcal{M} \arrow[l, bend right, "L"'] \end{tikzcd}$$

Here:

- $\mathcal{L}$ is a symmetric monoidal category
- $\mathcal{M}$ is a Cartesian monoidal category

LNL models give semantics to the linear simply typed λ-calculus
Note: we do not consider other notions of model in this talk (like Lafont and Seely model)

[1] "A Mixed Linear and Non-Linear Logic: Proofs, Terms and Models", Nick Benton

Variations of LNL models

There are many variations of LNL models

- **ECLNL models**[2]: formulated using **enriched categories**, and provide a language for string diagrams

[2] "Enriching a linear/non-linear lambda calculus: A programming language for string diagrams", Lindenhovius, Mislove, Zamdzhiev

[3] "Linear realizability and full completeness for typed lambda-calculi", Abramsky, Lenisa

[4] "Impredicativity in linear dependent type theory", Speight, Van der Weide

[5] "Models of linear dependent type theory", Lundfall

Variations of LNL models

There are many variations of LNL models

- **ECLNL models**[2]: formulated using **enriched categories**, and provide a language for string diagrams
- **LNL hyperdoctrines**[3]: formulated using **fibred categories**, provide an interpretation of **linear System F**

[2] "Enriching a linear/non-linear lambda calculus: A programming language for string diagrams", Lindenhovius, Mislove, Zamdzhiev

[3] "Linear realizability and full completeness for typed lambda-calculi", Abramsky, Lenisa

[4] "Impredicativity in linear dependent type theory", Speight, Van der Weide

[5] "Models of linear dependent type theory", Lundfall

Variations of LNL models

There are many variations of LNL models

- **ECLNL models**[2]: formulated using **enriched categories**, and provide a language for string diagrams
- **LNL hyperdoctrines**[3]: formulated using **fibred categories**, provide an interpretation of **linear System F**
- **Internal LNL models**: formulated using **internal categories**, useful in the context of **linear realizability**[4]

[2] "Enriching a linear/non-linear lambda calculus: A programming language for string diagrams", Lindenhovius, Mislove, Zamdzhiev

[3] "Linear realizability and full completeness for typed lambda-calculi", Abramsky, Lenisa

[4] "Impredicativity in linear dependent type theory", Speight, Van der Weide

[5] "Models of linear dependent type theory", Lundfall

Variations of LNL models

There are many variations of LNL models

- **ECLNL models**[2]: formulated using **enriched categories**, and provide a language for string diagrams
- **LNL hyperdoctrines**[3]: formulated using **fibred categories**, provide an interpretation of **linear System F**
- **Internal LNL models**: formulated using **internal categories**, useful in the context of **linear realizability**[4]
- **Linear comprehension categories**[5]: LNL hyperdoctrines with extra structure, provide an interpretation of **linear dependent type theory**

---

[2] "Enriching a linear/non-linear lambda calculus: A programming language for string diagrams", Lindenhovius, Mislove, Zamdzhiev

[3] "Linear realizability and full completeness for typed lambda-calculi", Abramsky, Lenisa

[4] "Impredicativity in linear dependent type theory", Speight, Van der Weide

[5] "Models of linear dependent type theory", Lundfall

We need a general framework!

- On the previous slide, we saw many variations of LNL models
- Each of these notions follows the same idea: we have an **adjunction** between a **symmetric monoidal** and a **Cartesian monoidal** category
- **Goal of this talk**: provide a general notion of LNL model that generalises each of these variations

But what should such a general framework look like?

Let's recall LNL models
An LNL model is a symmetric monoidal adjunction as follows.

$$\begin{tikzcd} \mathcal{L} \arrow[r, bend right, "\mathsf{M}"'] \arrow[r, phantom, "\perp"] & \mathcal{M} \arrow[l, bend right, "\mathsf{L}"'] \end{tikzcd}$$

Here:

- $\mathcal{L}$ is a symmetric monoidal category
- $\mathcal{M}$ is a Cartesian monoidal category

But what should such a general framework look like?

Let's recall LNL models
An LNL model is a **symmetric monoidal adjunction** as follows.

$$\begin{tikzcd} \mathcal{L} \arrow[r, bend right, "\mathsf{M}"'] \arrow[r, phantom, "\perp"] & \mathcal{M} \arrow[l, bend right, "\mathsf{L}"'] \end{tikzcd}$$

Here:

- $\mathcal{L}$ is a **symmetric monoidal** category
- $\mathcal{M}$ is a **Cartesian monoidal** category

Key notions behind LNL models

LNL models are formulated using the following notions:

- Adjunctions
- Symmetric monoidal
- Cartesian monoidal

Key notions behind LNL models

LNL models are formulated using the following notions:

- Adjunctions
- Symmetric monoidal
- Cartesian monoidal

Main: each of these notions are 2-categorical!
So: we can **formulate the notion of LNL model 2-categorically**

Some 2-categorical notions: adjunctions

Definition

Let C be a 2-category. An **adjunction** consists of morphisms $l : x \to y$ and $r : y \to x$ as follows

$$\begin{tikzcd} x \arrow[r, shift left=2, "l"] \arrow[r, phantom, "\perp"] & y \arrow[l, shift left=2, "r"] \end{tikzcd}$$

We also have 2-cells η and ε satisfying the triangle equations.

$$\begin{tikzcd} x \arrow[r, "l"] \arrow[rr, bend right, "\mathrm{id}"'] \arrow[r, phantom, shift right=3, "\Rightarrow \eta"] & y \arrow[r, "r"] & x \end{tikzcd}$$

$$\begin{tikzcd} y \arrow[r, "r"] \arrow[rr, bend right, "\mathrm{id}"'] \arrow[r, phantom, shift right=3, "\Leftarrow \varepsilon"] & x \arrow[r, "l"] & x \end{tikzcd}$$

Some 2-categorical notions: Cartesian objects

Definition

Let C be a 2-category with strict finite products. We say that x is **Cartesian**[6] whenever we have adjunctions as follows

$$\begin{tikzcd} x \times x \arrow[r, shift right, "m"'] \arrow[r, phantom, "\perp"] & x \arrow[l, shift right, "\Delta"'] \arrow[r, shift left, "!"] \arrow[r, phantom, "\perp"] & 1 \arrow[l, shift left, "u"] \end{tikzcd}$$

[6] "A 2-categorical approach to change of base and geometric morphisms. I", Carboni, Kelly, Street

Some 2-categorical notions: Cartesian objects

Definition

Let C be a 2-category with strict finite products. We say that x is **Cartesian**[6] whenever we have adjunctions as follows

$$
\begin{tikzcd}
x \times x \arrow[r, shift right=1.5ex, "m"'] \arrow[r, phantom, "\perp"] & x \arrow[l, shift right=1.5ex, "\Delta"'] \arrow[r, shift left=1.5ex, "!"] \arrow[r, phantom, "\perp"] & 1 \arrow[l, shift left=1.5ex, "u"]
\end{tikzcd}
$$

Note: the generalisation is based on the fact that we can give a "global" definition of limits using adjoints

[6] "A 2-categorical approach to change of base and geometric morphisms. I", Carboni, Kelly, Street

Some 2-categorical notions: pseudomonoids

Definition

Let C be a 2-category with strict finite products. A **symmetric pseudomonoid**[7] consists of an object x together with 1-cells $u : 1 \to x$ and $m : x \times x \to x$ and invertible 2-cells witnessing unitality, associativity, and symmetry (the diagrams are in the appendix). These 2-cells are required to satisfy analogous laws as symmetric monoidal categories.

[7] "Monoidal bicategories and Hopf algebroids", Day, Street

Some 2-categorical notions: pseudomonoids

Definition

Let C be a 2-category with strict finite products. A **symmetric pseudomonoid**[7] consists of an object x together with 1-cells $u : 1 \to x$ and $m : x \times x \to x$ and invertible 2-cells witnessing unitality, associativity, and symmetry (the diagrams are in the appendix). These 2-cells are required to satisfy analogous laws as symmetric monoidal categories.

Note:

- We can define a 2-category of symmetric pseudomonoids, and hence we can talk a symmetric monoidal adjunctions
- Every Cartesian object gives rise to a symmetric pseudomonoid

[7] "Monoidal bicategories and Hopf algebroids", Day, Street

Internal LNL Adjunctions

Definition

An **internal LNL adjunction** in a 2-category C with strict finite products consists of

- a symmetric pseudomonoid $\mathcal{L}$
- a Cartesian object $\mathcal{M}$
- 1-cells $\mathsf{L} : \mathcal{M} \to \mathcal{L}$ and $\mathsf{M} : \mathcal{L} \to \mathcal{M}$

These 1-cells are required to form an adjunction $\mathsf{L} \dashv \mathsf{M}$ in the 2-category of symmetric pseudomonoids

2-category theory is nice! Generality

Our notion of internal LNL adjunction encapsulates notions that we discussed before

- **LNL models**: internal LNL adjunctions in the 2-category of categories

2-category theory is nice! Generality

Our notion of internal LNL adjunction encapsulates notions that we discussed before

- **LNL models**: internal LNL adjunctions in the 2-category of categories
- **ECLNL models**: special case of internal LNL adjunctions in the 2-category of enriched categories

2-category theory is nice! Generality

Our notion of internal LNL adjunction encapsulates notions that we discussed before

- **LNL models**: internal LNL adjunctions in the 2-category of categories
- **ECLNL models**: special case of internal LNL adjunctions in the 2-category of enriched categories
- **LNL hyperdoctrines**: internal LNL adjunctions in the 2-category of fibred categories

2-category theory is nice! Generality

Our notion of internal LNL adjunction encapsulates notions that we discussed before

- **LNL models**: internal LNL adjunctions in the 2-category of categories
- **ECLNL models**: special case of internal LNL adjunctions in the 2-category of enriched categories
- **LNL hyperdoctrines**: internal LNL adjunctions in the 2-category of fibred categories
- **Internal LNL models**: internal LNL adjunctions in the 2-category of internal categories

2-category theory is nice! Constructions

The following theorem is nice

Theorem

Every pseudofunctor F that preserves finite products also preserves LNL adjunctions.

It follows from the facts that

- All pseudofunctors preserve adjunctions
- Pseudofunctors that preserve finite products can be lifted to symmetric pseudomonoids
- Pseudofunctors that preserve finite products preserve Cartesian objects

Externalisation

This theorem gives us a nice way to construct LNL hyperdoctrines

- **Externalisation**: every internal category gives rise to a split fibration
- One can show that externalisation preserves finite products
- Hence, every internal LNL model induces an LNL hyperdoctrine

What about higher categorical models?

We conjecture that our notions can be used for bicategorical models as follows.

- There is a 2-category of pseudo double categories
- Every pseudo double category gives rise to a bicategory
- **Conjecture**: applying this construction to internal LNL adjunctions of pseudo double categories gives bicategorical models of linear logic

For ∞-categorical models[8]: does not fall in our framework. The issue is that monoidal structures for ∞-categories cannot be captured 2-categorically.

[8] "∞-Categorical models of linear logic", Harington, Mimram

Conclusion

- There are many variations of the notion of LNL model, each with their own applications
- We gave a general framework using 2-categorical notions: **internal LNL adjunctions**
- A wide class of pseudofunctors preserve internal adjunctions, which gives us a very nice method for constructing LNL adjunctions
- An application: externalisation of internal categories

Conclusion

- There are many variations of the notion of LNL model, each with their own applications
- We gave a general framework using 2-categorical notions: **internal LNL adjunctions**
- A wide class of pseudofunctors preserve internal adjunctions, which gives us a very nice method for constructing LNL adjunctions
- An application: externalisation of internal categories

Future work:

- Bicategorical models from double categorical models
- Using relative adjunctions instead

Pseudomonoids: unitality

$$\begin{tikzcd}
X \arrow[r, "\langle !, \mathrm{id} \rangle"] \arrow[drr, "\mathrm{id}"'] & 1 \times X \arrow[r, "u \times \mathrm{id}"] \arrow[dr, phantom, "\Leftarrow \mathrm{l}"] & X \times X \arrow[d, "m"] \\
 & & X
\end{tikzcd}$$

$$\begin{tikzcd}
X \arrow[r, "\langle \mathrm{id}, ! \rangle"] \arrow[drr, "m"'] & X \times 1 \arrow[r, "\mathrm{id} \times u"] \arrow[dr, phantom, "\Leftarrow \mathrm{r}"] & X \times X \arrow[d, "m"] \\
 & & X
\end{tikzcd}$$

Pseudomonoids: associativity

$$\begin{tikzcd}
(x \times x) \times x \arrow[rr, "\simeq"] \arrow[dd, "m \times \mathrm{id}"'] & & x \times (x \times x) \arrow[d, "\mathrm{id} \times m"] \\
& \arrow[phantom, "\Downarrow a"] & x \times x \arrow[d, "m"] \\
x \times x \arrow[rr, "m"'] & & x
\end{tikzcd}$$

Pseudomonoids: symmetry

$$
\begin{tikzcd}
X \times X \arrow[rr, "\langle \pi_2, \pi_1 \rangle"] \arrow[dr, "m"'] & \arrow[d, phantom, "\Rightarrow s"] & X \times X \arrow[dl, "m"] \\
& X &
\end{tikzcd}
$$