Mixed Linear and Non-linear Recursive Types

Vladimir Zamdzhiev

Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France

Joint work with Michael Mislove and Bert Lindenhovius

ICFP'19
Berlin
21 August 2019

Linear Logic

- Introduced by Girard in 1987.
- Resource-sensitive logic.
- 30+ years of research.
- Very few linear languages that are convenient for programming.

Mixed Linear/Non-linear type systems

- Mixed linear/non-linear type systems have recently found applications in:
  - concurrency (session types for π-calculus);
  - quantum programming (substructural limitations imposed by quantum information);
  - circuit description languages (dealing with wires of string diagrams);
  - programming resource-sensitive data (file handlers, etc.).
- This talk: add recursive types to a mixed linear/non-linear type system in a way that is convenient for programming.
- Very detailed categorical treatment:
  - a new technique for solving recursive domain equations within **CPO**;
  - coherence theorems;
  - sound and adequate categorical models.

Long story short

- Syntax and operational semantics is mostly based on prior work[1].
- Main difficulty is on the denotational and categorical side.
- How can we copy/discard non-linear recursive types *implicitly*?
  - A list of file handlers should be *linear* – cannot copy/discard.
  - A list of natural numbers should be *non-linear* – can copy/discard at will (and implicitly).
- How do we design a linear/non-linear fixpoint calculus (LNL-FPC)?

[1] Rios and Selinger, QPL'17; Lindenhovius, Mislove and Zamdzhiev LICS'18

Syntax

Type variables	X, Y, Z		
Term variables	x, y, z		
Types	A, B, C	$::=$	$X \mid A + B \mid A \otimes B \mid A \multimap B \mid !A \mid \mu X.A$
Non-linear types	P, R	$::=$	$X \mid P + R \mid P \otimes R \mid !A \mid \mu X.P$
Type contexts	Θ	$::=$	$X_1, X_2, \ldots, X_n$
Term contexts	Γ, Σ	$::=$	$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$
Terms	m, n, p	$::=$	$x \mid \mathsf{left}_{A,B}\, m \mid \mathsf{right}_{A,B}\, m$
			$\mid \mathsf{case}\ m\ \mathtt{of}\ \{\mathsf{left}\ x \to n\ \mathsf{right}\ y \to p\}$
			$\mid \langle m, n \rangle \mid \mathsf{let}\ \langle x, y \rangle = m\ \mathtt{in}\ n \mid \lambda x^A.m \mid mn$
			$\mid \mathsf{lift}\ m \mid \mathsf{force}\ m \mid \mathtt{fold}_{\mu X.A} m \mid \mathtt{unfold}\ m$
Values	v, w	$::=$	$x \mid \mathsf{left}_{A,B}\, v \mid \mathsf{right}_{A,B}\, v \mid \langle v, w \rangle \mid \lambda x^A.m$
			$\mid \mathsf{lift}\ m \mid \mathtt{fold}_{\mu X.A} v$
Term Judgements	$\Theta; \Gamma \vdash m : A$		

Operational Semantics

$$\frac{}{x \Downarrow x} \qquad \frac{m \Downarrow v}{\text{left } m \Downarrow \text{left } v} \qquad \frac{m \Downarrow v}{\text{right } m \Downarrow \text{right } v}$$

$$\frac{m \Downarrow \text{left } v \qquad n[v/x] \Downarrow w}{\text{case } m \texttt{ of } \{\text{left } x \to n \mid \text{right } y \to p\} \Downarrow w} \qquad \frac{m \Downarrow \text{right } v \qquad p[v/y] \Downarrow w}{\text{case } m \texttt{ of } \{\text{left } x \to n \mid \text{right } y \to p\} \Downarrow w}$$

$$\frac{m \Downarrow v \qquad n \Downarrow w}{\langle m, n\rangle \Downarrow \langle v, w\rangle} \qquad \frac{m \Downarrow \langle v, v'\rangle \qquad n[v/x, v'/y] \Downarrow w}{\text{let } \langle x, y\rangle = m \text{ in } n \Downarrow w}$$

$$\frac{}{\lambda x.m \Downarrow \lambda x.m} \qquad \frac{m \Downarrow \lambda x.m' \qquad n \Downarrow v \qquad m'[v/x] \Downarrow w}{mn \Downarrow w}$$

$$\frac{}{\text{lift } m \Downarrow \text{lift } m} \qquad \frac{m \Downarrow \text{lift } m' \qquad m' \Downarrow v}{\text{force } m \Downarrow v} \qquad \frac{m \Downarrow v}{\texttt{fold } m \Downarrow \texttt{fold } v} \qquad \frac{m \Downarrow \texttt{fold } v}{\texttt{unfold } m \Downarrow v}$$

Term level recursion

In FPC, term recursion is induced by the isorecursive type structure. The same is true for LNL-FPC.

Theorem

The term-level recursion operator from[2] *is now a derived rule. For a given term* $\Phi, z :!A \vdash m : A$, *define:*

$$\begin{aligned} \alpha_m^z &\equiv \mathit{lift}\ \mathit{fold}\ \lambda x^{!\mu X.(!X \multimap A)}.(\lambda z^{!A}.m)(\mathit{lift}\ (\mathit{unfold}\ \mathit{force}\ x)x) \\ \mathit{rec}\ z^{!A}.m &\equiv (\mathit{unfold}\ \mathit{force}\ \alpha_m^z)\alpha_m^z \end{aligned}$$

[2] Lindenhovius, Mislove, Zamdzhiev: Enriching a Linear/Non-linear Lambda Calculus: A Programming Language for String Diagrams. LICS 2018

Example: functorial function

```
rec fact. λ n.
 case unfold n of
    left u -> succ zero
    right n’ -> mult(n, (force fact) n’)
```

Remark

The above program is written in the formal syntax without syntactic sugar. Note: implicit rules for copying and discarding.

Models of Intuitionistic Linear Logic

A model of ILL[3] is given by the following data:

- A cartesian closed category $\mathbf{C}$ with finite coproducts.
- A symmetric monoidal closed category $\mathbf{L}$ with finite coproducts.
- A symmetric monoidal adjunction:

$$\begin{tikzcd} \mathbf{C} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathbf{L} \arrow[l, bend left, "G"] \end{tikzcd}$$

[3]Nick Benton. *A mixed linear and non-linear logic: Proofs, terms and models*. CSL'94

Models of LNL-FPC

Definition

A **CPO**-*LNL model* is given by the following data:

1. A **CPO**-symmetric monoidal closed category $(\mathbf{L}, \otimes, \multimap, I)$, such that:
   - 1a. $\mathbf{L}$ has an initial object 0, such that the initial morphisms $e : 0 \to A$ are embeddings;
   - 1b. $\mathbf{L}$ has ω-colimits over embeddings;
   - 1c. $\mathbf{L}$ has finite **CPO**-coproducts, where $(- + -) : \mathbf{L} \times \mathbf{L} \to \mathbf{L}$ is the coproduct functor.
2. A **CPO**-symmetric monoidal adjunction $\begin{tikzcd} \mathbf{CPO} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathbf{L} \arrow[l, bend left, "G"] \end{tikzcd}$.

Theorem

In every **CPO**-*LNL model* $\mathbf{L}$ *is* **CPO**-*algebraically compact.*

Concrete Models

- Simplest non-trivial model: $\mathbf{CPO} \underset{U}{\overset{(-)_\perp}{\underset{\longleftarrow}{\overset{\longrightarrow}{\perp}}}} \mathbf{CPO}_{\perp!}$.
- A *class* of concrete models based on (enriched) presheaves into $\mathbf{CPO}_{\perp!}$. Concrete models for:
  - Quantum programming.
  - Circuit description languages.
  - String diagram description languages.
  - Petri net description languages.

A new technique for solving recursive domain equations

Problem

How to interpret the non-linear recursive types within **CPO**.

Definition

Let $T : \mathbf{A} \to \mathbf{B}$ be a **CPO**-functor between **CPO**-categories **A** and **B**. A morphism f in **A** is called a *pre-embedding with respect to* T if Tf is an embedding in **B**.

Definition

Let $\mathbf{CPO}_{pe}$ be the full-on-objects subcategory of **CPO** of all cpo's with pre-embeddings with respect to the functor $F : \mathbf{CPO} \to L$.

Example

Every embedding in **CPO** is a pre-embedding, but not vice versa. The empty map $\iota : \varnothing \to X$ is a pre-embedding (w.r.t to F in our model), but not an embedding.

Denotational Semantics (Types)

Main ideas:

- Provide a standard interpretation for all types $\llbracket \Theta \vdash A \rrbracket : \mathbf{L_e}^{|\Theta|} \to \mathbf{L}_e$.
- A closed type is interpreted as $\llbracket A \rrbracket \in \mathrm{Ob}(\mathbf{L}_e) = \mathrm{Ob}(\mathbf{L})$.
- Provide a non-linear interpretation for non-linear types $(\!|\Theta \vdash P|\!) : \mathbf{CPO}_{pe}^{|\Theta|} \to \mathbf{CPO}_{pe}$.
- A closed non-linear type admits an interpretation as $(\!|P|\!) \in \mathrm{Ob}(\mathbf{CPO}_{pe}) = \mathrm{Ob}(\mathbf{CPO})$.
- **Theorem:** For any closed non-linear type P, there exists an isomorphism

$$\alpha^P : \llbracket P \rrbracket \cong F(\!|P|\!)$$

which satisfies some important coherence conditions.

Copying and discarding

Definition

We define morphisms, called discarding ($\diamond$), copying ($\triangle$) and promotion ($\Box$):

$$\diamond^{\Psi} := \llbracket\Psi\rrbracket \xrightarrow{\alpha} F(\!|\Psi|\!) \xrightarrow{F1} F1 \xrightarrow{u^{-1}} I;$$

$$\triangle^{\Psi} := \llbracket\Psi\rrbracket \xrightarrow{\alpha} F(\!|\Psi|\!) \xrightarrow{F\langle \mathrm{id},\mathrm{id}\rangle} F\left((\!|\Psi|\!) \times (\!|\Psi|\!)\right) \xrightarrow{m^{-1}} F(\!|\Psi|\!) \otimes F(\!|\Psi|\!) \xrightarrow{\alpha^{-1}\otimes\alpha^{-1}} \llbracket\Psi\rrbracket \otimes \llbracket\Psi\rrbracket;$$

$$\Box^{\Psi} := \llbracket\Psi\rrbracket \xrightarrow{\alpha} F(\!|\Psi|\!) \xrightarrow{F\eta} !F(\!|\Psi|\!) \xrightarrow{!\alpha^{-1}} !\llbracket\Psi\rrbracket,$$

where Ψ is a closed non-linear type or non-linear term context.

Proposition

The triple $\left(\llbracket\Psi\rrbracket, \triangle^{\Psi}, \diamond^{\Psi}\right)$ *forms a cocommutative comonoid in* $\mathbf{L}$.

Denotational Semantics (Terms)

- A term $\Gamma \vdash m : A$ is interpreted as a morphism $[\![\Gamma \vdash m : A]\!] : [\![\Gamma]\!] \to [\![A]\!]$ in $\mathbf{L}$ in the standard way.
- The interpretation of a non-linear value $[\![\Phi \vdash v : P]\!]$ commutes with the substructural operations of ILL (shown by providing a non-linear interpretation $(\!|\Phi \vdash v : P|\!)$ within **CPO**).
- Soundness: If $m \Downarrow v$, then $[\![m]\!] = [\![v]\!]$.
- Adequacy: For models that satisfy some additional axioms, the following is true: for any $\cdot \vdash m : P$ with P non-linear, then $m \Downarrow$ iff $[\![m]\!] \neq \perp$.

Conclusion and Future Work

- Introduced LNL-FPC: the linear/non-linear fixpoint calculus;
- Implicit weakening and contraction rules (copying and deletion of non-linear variables);
- New results about parameterised initial algebras;
- New technique for solving recursive domain equations in **CPO**;
- Detailed semantic treatment of mixed linear/non-linear recursive types;
- Sound and adequate models;
- How to implicitly deal with lambda abstractions?

Thank you for your attention!