Linear Dependent Types

Zhaohui Luo
Royal Holloway
University of London

(Based on work with Yu Zhang at Chinese Academy of Sciences)

Linear types and dependent types


- Linear types (Girard 1987):
  - $A \multimap B$, …
- Dependent types (Martin-Löf 1970s):
  - $\Pi x{:}A.B[x]$, …
- How to combine them?
  - In most of existing work (Pfenning et al 2002, Krishnaswami et al 2015, Vákár 2015)
    - B[x] only when x is intuitionistic (non-linear).
    - Hence it is possible to separate intuitionistic Γ and linear Δ: Γ; Δ |- a : A
    - Δ depends on Γ, but not the other way around.
  - Except McBride's recent work (2016) – independent/different.
- This paper: LDTT, where types can depend on linear variables.

Motivations

- Reasoning
  - Linear functions and logic in the same language
  - Internal reasoning about linear functions
- Concurrency
  - Dependent session types
  - Curry-Howard for concurrent programs
- Natural language analysis
  - Dependent categorial grammars
  - Uniform analysis of syntax/semantics in modern TTs

Reasoning about linear functions

- Type theory
  - Language for both programming and reasoning
  - For f : Nat→Nat, x:Nat,

    $$f(x) = x$$

    is a proposition in the same language in which f is written.
- What if f : Nat–$\circ$Nat and x::Nat (a linear variable)?
  - f(x) = x is now a proposition/type depending on linear x …
  - This would not be allowed in previous work.
  - We would allow this, written as

    $$Eq_{Nat}(f\ x, x)$$

  - ➔ linear dependent types

Dependent session types

- Girard's claim:
  - Linear logic offers Curry-Howard for concurrent programs.
  - Attempts in 1990s (Abramsky, Bellin-Scott, …)
- Breakthrough: Caires and Pfenning (2010)
  - Linear formulas $\Leftrightarrow$ session types
  - Linearity: unique ownership of channel endpoints.
- Dependent session types?
  - cf, Toninho …, Gay at TYPES16, …
  - Truly dependent? E.g., ?x:Int.B(x) – linear Π-type

➔ linear dependent types

Dependent Categorial Grammars

- Lambek calculus (1958) & Lambek CGs
  - Syntactic analysis of natural language
  - "Ordered" system (linear + no-exchange)
  - Corresponding to Montague semantics (only →)

- Formal semantics of NLs in modern TTs
  - Ranta (1994) in Martin-Löf's type theory
  - Recent developments of MTT-semantics: leading to a full-blown better alternative to Montague semantics
  - See http://www.cs.rhul.ac.uk/home/zhaohui/lexsem.html
- Syntactic analysis corresponding to MTT-semantics?

  ➔ linear/ordered dependent types

LDTT – a Linear Dependent Type Theory

- Types: intuitionistic/linear Π-types, equality types
- Contextual regime
  - Contexts are sequences of two forms of entries:

    x:A, y::B[x], z:C[x,y], …
    - Intuitionistic variables x : A
    - Linear variables y :: B
  - Types dependent on linear variables
    - Example: x::A, f : A$\multimap$A $\vdash$ Eq_A(f x, x) type

Intuitionistic Π-types

$$\frac{\Gamma,\ x:A \vdash B\ type}{\Gamma \vdash \Pi x{:}A.B\ type} \qquad \frac{\Gamma,\ x:A \vdash b:B}{\Gamma \vdash \lambda x{:}A.b : \Pi x{:}A.B} \qquad \frac{\Gamma \vdash f : \Pi x{:}A.B \quad \overline{\Gamma} \vdash a : A}{\Gamma \vdash f(a) : [a/x]B}$$

- $\overline{\Gamma}$ – the intuitionistic part of Γ (A proven intuitionistically)
  - $\overline{\Gamma}$ = Γ \ $FV_{LD}(\Gamma)$ – removing the linear dependent variables
    - $FV_{LD}(\langle\rangle) = \varnothing$
    - $FV_{LD}(\Gamma,x{:}A) = FV_{LD}(\Gamma)$ if $FV(A) \cap FV_{LD}(\Gamma) = \varnothing$;
      $= FV_{LD}(\Gamma) \cup \{x\}$ otherwise
    - $FV_{LD}(\Gamma,x::A) = FV_{LD}(\Gamma) \cup \{x\}$
  - Example: $\Gamma \equiv x{:}A, y::B, z{:}C$
    $\overline{\Gamma} \equiv x{:}A, z{:}C$ if $y \notin FV(C)$
    $\overline{\Gamma} \equiv x{:}A,$ if $y \in FV(C)$

Linear Π-types

$$\frac{\Gamma,\ x{::}A \vdash B\ type}{\Gamma \vdash \Pi x{::}A.B\ type} \qquad \frac{\Gamma,\ x{::}A \vdash b : B}{\Gamma \vdash \lambda x{::}A.b : \Pi x{::}A.B} \qquad \frac{\Gamma \vdash f : \Pi x{::}A.B \quad \Delta \vdash a : A \quad Merge(\Gamma;\Delta)\downarrow}{Merge(\Gamma;\Delta) \vdash f\ a : [a/x]B}$$

- Merge(Γ;Δ) is only defined if
  - $\overline{\Gamma} \equiv \overline{\Delta}$ (the intuitionistic parts are the same)
  - $FV_{LD}(\Gamma) \cap FV_{LD}(\Delta) = \varnothing$ (Γ/Δ do not share linear dependent variables)
- When the above are the case, Merge is defined as:

(a) $Merge(\langle\rangle;\langle\rangle) = \langle\rangle$.

(b) If $x \in FV_{LD}(\Gamma) \cup FV_{LD}(\Delta)$,

$$Merge(\Gamma, x{:}\bar{:}A;\ \Delta) = Merge(\Gamma;\ \Delta, x{:}\bar{:}A) = Merge(\Gamma;\Delta), x{:}\bar{:}A,$$

where $\bar{:}$ is either : or ::.

(c) $Merge(\Gamma, x{:}A;\ \Delta, x{:}A) = Merge(\Gamma;\Delta), x{:}A$.

- Example: Γ ≡ x:A, y_1::B_1, z:C

  Δ ≡ x:A, y_2::B_2, z:C

  Merge(Γ;Δ) ≡ x:A, z:C, y_1::B_1, y_2::B_2

Note: $y_1 \neq y_2$ and $y_1, y_2 \notin FV(C)$ for otherwise, Merge(Γ;Δ) would be undefined.

Equality Types

- Formation rule

$$\frac{\Gamma \vdash a : A \quad \Delta \vdash b : A \quad merge(\Gamma;\Delta)\downarrow}{merge(\Gamma;\Delta) \vdash Eq_A(a,b)\ type}$$

- merge(Γ;Δ) is defined only when var-sharing is OK:

  x?A∈Γ, x?B∈Δ ➔ A≡B and ? is both : or both ::

- merge(Γ;Δ) is defined as

(a) $merge(\Gamma;\langle\rangle) = \Gamma.$

(b) $$merge(\Gamma;\ x{:}A, \Delta) = \begin{cases} merge(\Gamma;\Delta) & \text{if } x \in FV(\Gamma) \\ merge(\Gamma, x{:}A;\ \Delta) & \text{otherwise} \end{cases}$$

- Examples:
  - x::A, f : A–◦A |– f x : A and x::A |– x : A ➔ x::A, f : A–◦A |– Eq_A(f x, x) type
  - x::A |– x : A and y::A |– y : A ➔ x::A, y::A |– Eq(x,y) type

Variable Typing

$$\frac{\Gamma,\ x\bar{:}A,\ \Gamma'\ valid \quad (\text{for all } y{::}\Gamma_y \in \Gamma,\ y \in D_\Gamma(x)) \quad \Gamma' \text{ intuitionistic}}{\Gamma,\ x\bar{:}A,\ \Gamma' \vdash x : A} \qquad (\bar{:} \in \{:, ::\})$$

where

- *Γ′ intuitionistic* means that it does not have linear ::-entries
- $D_\Gamma(x)$ is defined as:
  - $x \in D_\Gamma(x)$;
  - For any $y \in D_\Gamma(x)$, $FV(\Gamma_y) \subseteq D_\Gamma(x)$.

- Examples:
  - Judgements derivable intuitionistically are derivable.
  - x::A,y:B(x) |– x:A and x::A,y:B(x) |– y:B(x) are derivable since $x \in B(x)$.
  - x::A, x′::A, y:B(x) |– y : B(x) is *not* derivable if $x' \notin B(x)$.

Weak Linearity

- *Defn (essential occurrences)* Let *Γ |- a:A*. The multiset $E_\Gamma(a)$ of variables essentially occurring in *a* under *Γ* is inductively defined as follows (Eq-types omitted):
  - Variable typing: $E_{\Gamma,\ x\bar{:}A,\ \Gamma'}(x) = D_{\Gamma,\ x\bar{:}A,\ \Gamma'}(x)$
  - λ-typing: $E_\Gamma(\lambda x\bar{:}A.b) = E_{\Gamma,x\bar{:}A}(b)\backslash\{x\}$
  - Intuitionistic applications: $E_\Gamma(f(a)) = E_\Gamma(f) \cup E_{\overline{\Gamma}}(a)$
  - Linear applications: $E_{Merge(\Gamma;\Delta)}(f\ a) = E_\Gamma(f) \cup E_\Delta(a)$
- *Theorem (weak linearity)*

  In LDTT, every linear variable occurs essentially for exactly once in a well-typed term. Formally,

  Γ, y::B, Γ' |- a : A ➔ $y \in E_{\Gamma,y::B,\Gamma'}(a)$ only once.

Summary and Related/Future Work

- Extended abstract of the technical development:
  - Z. Luo and Y. Zhang. A Linear Dependent Type Theory. TYPES 2016. Novi Sad, 2016. (Available as http://www.cs.rhul.ac.uk/~zhaohui/TYPES16.pdf)
- Work on linearity in dependent types
  - Eg, (Pfenning et al, I&C02), (Krishnaswami et al, POPL15), (Vákár, FoSSaCS 15)
  - Lambek calculus with dependent types (Luo, TYPES 2015)
  - Types in all above are non-dependent on linear/Lambek variables
  - McBride 2016 (Walder Festschrift)
    - More general setting: considering "prices" $\rho \in \{0,1,w\}$ and intuitionistic/linear Π-types $(\rho x{:}A) \to B$.
    - Independent with the current work and comparison to be done.
- Examples of future work
  - Extension to other linear/Lambek type constructors
  - Implementation: type-checking algorithm done in Haskell.

October 2016