演習 III 第 1 期

Quantitative Type Theory

稲吉悠羽

April 22, 2025

依存型と線形型とは

依存型

λ 抽象を項に依存する項, 多相関数を型に依存する項と思ったとき, 依存型は項に依存する型.

- よく例に挙げられるのは, サイズ付きのリスト型

  リスト (項) がどの長さを持つかによって, 対応する型が変わる → 型が項に依存している

- 他に重要な例として, Martin-Löf 型理論における Identity type

  項の等しさの証拠を依存型の項に持たせる → 定理証明支援系で等式の証明を作るのに有用

線形型

プログラム (型検査) の中でちょうど 1 回しか使われない項であることを保証する型.

線形型って依存型で実現できないの?

だって, 依存型は項の性質を表すのに使えるじゃん !!

結論としては, できない !!

(少なくともあまり綺麗にはできなさそう)

線形型の何回使ったかという性質は型を超えたメタレベル (型検査を動かす側) の情報が必要なので, 依存型という型システムのなかの存在ではその挙動をとらえきれない

さらに依存型で型の中に項が出現するが, ここでの使用も線形型の使用カウントへ考慮する必要がある
→ リソース管理が複雑に... 既存の理論ではうまく表現できない

新しい理論の必要性!!

QTT の誕生

他にも同様の目的の理論として, Linear Dependent Type Theory などがある

もともとの目的

ランタイムに必要な情報 (computational use) と型検査で必要な情報 (type formation) を区別してメモリの効率を上げたい. このために, シークエント計算において左辺にある命題を使える回数を考える線形論理を用いることにした. 使用回数は命題を全く使わない 0, ちょうど 1 回つかう 1, そして制限なく使える multiple ω として区別される. 文脈の結合や分配に対応するためにはこれらの要素に対する半環を考えれば十分であった.

McBride のアイディア

上記のような発想をもとに体系を形式化するのに重要だったのは, 半環の 0 をランタイムに消されるが型検査には使える情報として扱えるようにすることだった. 実行時に消せるかどうかを分析しつつ, 体系の複雑性が上がりすぎないようにするため, 型付け規則ではある項が「使われたか (present)」それとも「使われなかったか (erased)」だけをちょうど追跡するようにした.

Section 2 : 構文論

内容

- McBride(2016) によるアイディアに必要な数量的な注釈と半環について
- BNF 記法による項や型の文法
- 型付けの推論規則 (双方向型付け)
- 構文論的性質に関する補題 (弱化や代入など)

ポイント

- 文脈に対する注釈
- Dependent Tensor Product Types
- McBride(2016) による推論規則の問題点の指摘&修正

Section 3 : 圏論的意味論

- 依存型の圏論的意味論に用いる Category with Families (CwF) の導入
- 上記を resource annotation への対応のために拡張した QCwF の導入
- Type Formers のための拡張の方針
- QTT の意味論と健全性 (証明の概要)

Section 4 : Realisability

- BCI-代数の拡張 Linear Combinatory Algebra(LCA) と 線形含意
- R-Linear Comibinatory Algebra (R-LCA) の導入
- Category of Assemblies の導入
- R-LCA から QCwF の構成

構文

semiring elements $\rho, \pi \in R$

precontexts $\Gamma ::= \diamond \mid \Gamma, x \overset{\rho}{:} S$

pretypes

$$S, T, U ::= (x \overset{\pi}{:} S) \to T \mid (x \overset{\pi}{:} S) \otimes T \mid I \mid \mathsf{Bool} \mid \mathsf{El}(M) \mid \mathsf{Set}$$

preterms

$$\begin{aligned} M, N, O ::= {} & x \mid \lambda x \overset{\pi}{:} S.M^T \mid \mathsf{App}_{(x \overset{\pi}{:} S)T}(M, N) \mid (M, N)_{x \overset{\pi}{:} S.T} \mid * \mid \mathsf{fst}_{x \overset{\pi}{:} S.T}(M) \\ & \mid \mathsf{snd}_{x \overset{\pi}{:} S.T}(M) \mid \mathsf{let}_{x \overset{\pi}{:} S.T}(x, y) = M \text{ in } N \mid \mathsf{let}_I * = M \text{ in } N \mid \mathsf{true} \mid \mathsf{false} \\ & \mid \mathsf{ElimBool}_{(z)T}(M_t, M_f, N) \mid (x \overset{\pi}{:} M) \to N \end{aligned}$$

型判断

McBride による新しい体系では, 型判断は

$$x_1 \overset{\rho_1}{:} S_1, x_2 \overset{\rho_2}{:} S_2, \ldots, x_n \overset{\rho_n}{:} S_n \vdash M : T$$

のようになる. ここで ρ_i は computational usage を表す. 0 ならばそれが実行時には使われないということを示している. 項の型付けは実際には, 上式の右辺において $M \overset{\sigma}{:} T$ となったものを考える. この σ は semiring の 0,1 のいずれかである. この制限は McBride(2016) の問題を踏まえたものであり, 体系で代入を認めるために必要である.

半環

型の推論規則では台集合 R と二項演算 $(+), (\cdot)$ からなり, 定数 $0, 1 \in R$ について $(R, +, 0)$ が可換モノイド, $(R, \cdot, 1)$ がモノイドで, $(+), (\cdot)$ が分配法則を満たし, $0\rho = \rho 0 = 0$ となるものを考える. 必ずしも $\{0, 1, \omega\}$ には限定されていない.

$(+)$: 複数の文脈における使用を累積させる

ElimBool は依存 if 文のような構文. 条件節の文脈 Γ_2 と then/else 節 の文脈 Γ_1 における使用を合わせたものが全体の使用にあたる. M_t, M_f のどちらの項も同じ文脈から導かれているので, 片方しか評価されないこともわかる.

$$\frac{0\Gamma_1, z \overset{0}{:} \mathsf{Bool} \vdash T \quad \Gamma_1 \vdash M_t \overset{\sigma}{:} T[\mathsf{true}/z] \quad \Gamma_1 \vdash M_f \overset{\sigma}{:} T[\mathsf{false}/z] \quad \Gamma_2 \vdash N \overset{\sigma}{:} \mathsf{Bool} \quad 0\Gamma_1 = 0\Gamma_2}{\Gamma_1 + \Gamma_2 \vdash \mathsf{ElimBool}_{(z)T}(M_t, M_f, N) \overset{\sigma}{:} T[N/z]} \tag{B-ELIM}$$

$(\cdot)$: ネストした使用のカウント

関数内で各 π 回使われる変数があり, 関数自体を σ 回使うとすると, 文脈の中で変数は計 $\pi\sigma$ 回使われる.

$$\frac{\Gamma, x \overset{\sigma\pi}{:} S \vdash M \overset{\sigma}{:} T}{\Gamma \vdash \lambda x \overset{\pi}{:} S.M^T \overset{\sigma}{:} (x \overset{\pi}{:} S) \to T} \tag{LAM}$$

R への要請

半環 R が

1. positive: つまり $\rho + \pi = 0$ ならば, $\rho = 0$ かつ $\pi = 0$ である
2. zero-product property : $\rho\pi = 0$ ならば, $\rho = 0$ または $\pi = 0$ である

という性質をもつことを, 代入を認めるために要求する.

文脈に対する演算

基本的には, 各項単位での演算をするものとして解釈する. 文脈の和 $\Gamma_1 + \Gamma_2$ は $0\Gamma_1 = 0\Gamma_2$ の場合にのみ定義される. これは usage annotation 以外の文脈の要素が演算の適用によって変化することがないということである.

0 fragment は「静的型検査として」完全に従来の型理論と同じ. この場合には型検査と実行は切り離して考えている. Atkey(2018) で項の評価規則は明示的に与えられておらず, 抽象的な扱いに留められている. Section3 で考える意味論は静的な範囲しか扱わない. Section4 ではすこし実行時の計算のことも考え始める.

ルールが適応された文脈で Γ_2 が π 倍されている. これは M の中で π 回使われる変数 x に代入される N をその都度導出することに対応していそう.

$$\frac{\Gamma_1 \vdash M \overset{\sigma}{:} (x \overset{\pi}{:} S) \to T, \quad \Gamma_2 \vdash N \overset{\sigma'}{:} S, \quad 0\Gamma_1 = 0\Gamma_2, \quad \sigma' = 0 \Leftrightarrow (\pi = 0 \vee \sigma = 0)}{\Gamma_1 + \pi\Gamma_2 \vdash \mathsf{App}_{(x \overset{\pi}{:} S)T}(M, N) \overset{\sigma}{:} T[N/x]} \text{(APP)}$$

→ 適用前に一度だけ評価する値呼びではなく, そのまま項を代入して出現ごとに評価する名前呼びに相当する.
依存型の体系では, 型の決定に項の評価が関係しており, 値呼びだと型検査が難しくなるので, 名前呼びで考えるのが一般的.

補足: 依存型の型構成子

今考えたい型の全体である, 型宇宙を U とします.
依存型の体系には次のように構成される型があります.

Π -types

codomain が domain の要素に応じて変わりうる function type.
dependent function type や dependent product type ともいう.
型 A と型族 $B : A \to U$ について, dependent function type は $\Pi_{(x:A)}B(x) : U$ とかける. B が constant だと通常の function type になる: $\Pi_{(x:A)}B(x) \equiv A \to B$

Σ -types

dependent sum type や dependent pair type ともいう.
$\Sigma_{(x:A)}B(x) : U$ と表記する. B が定数関数だと, $(\Sigma_{x:A}B) \equiv A \times B$ となる.

直感的な解釈として, $\langle A \rangle$ は A という資源が高々 (ちょうど?)1 回使えることを表す.

$$\frac{}{\langle A \rangle \vdash A}\ \langle \mathsf{Id} \rangle$$

$A \otimes B$ は直感的には A, B が同時にしか消費できない資源であることを表している.

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ (\otimes\text{-I}) \qquad \frac{\Gamma \vdash A \otimes B \quad \Delta, \langle A \rangle, \langle B \rangle \vdash C}{\Gamma, \Delta \vdash C}\ (\otimes\text{-E})$$

疑問点

この型に対応する項は $*$ と $(M, N)_{x:^{\pi}S.T}$ であり, pair type ぽい.
しかしなぜ tensor というのか? → 線形論理の tensor 演算子に対応するから

tensor を使う理由

同時に消費することしたほうが, 数量的な制約に関する規則に都合がよいから.
やろうと思えば要素ごとに射影して別々に使えるようにもできるが, 複雑性が増す.
→ この背景を踏まえると, erased/present で除去規則が異なることに納得がいく

erased ($\sigma = 0$)

実行時の消費のことを考えなくてよいので, projection operator を使える!
また, I type の除去則はない. 実行時に消えるので型検査時にわざわざ明示的に除去する必要もない.

$$\frac{\Gamma \vdash M \overset{0}{:} (x \overset{\pi}{:} S) \otimes T}{\Gamma \vdash \mathsf{fst}_{x \overset{\pi}{:} S.T}(M) \overset{0}{:} A} \qquad \frac{\Gamma \vdash M \overset{0}{:} (x \overset{\pi}{:} S) \otimes T}{\Gamma \vdash \mathsf{snd}_{x \overset{\pi}{:} S.T}(M) \overset{0}{:} T[\mathsf{fst}(M)/x]}$$

present ($\sigma = 1$)

同時に資源を消費する必要がある. また, 数量的制限を守るために I type も明示的に消費する必要がある.

$$\frac{0\Gamma_1, z \overset{0}{:} (x \overset{\pi}{:} S) \otimes T \vdash U \quad \Gamma_1 \vdash M \overset{\sigma}{:} (x \overset{\pi}{:} S) \otimes T \quad \Gamma_2, x \overset{\sigma\pi}{:} S, y \overset{\sigma}{:} T \vdash N \overset{\sigma}{:} U[(x,y)/z] \quad 0\Gamma_1 = 0\Gamma_2}{\Gamma_1 + \Gamma_2 \vdash \mathsf{let}_{x \overset{\pi}{:} S.T}(x,y) = M \text{ in } N \overset{\sigma}{:} U[M/z]}$$

$$\frac{0\Gamma_1, x \overset{0}{:} I \vdash U \quad \Gamma_1 \vdash M \overset{\sigma}{:} I \quad \Gamma_2 \vdash N \overset{\sigma}{:} U[*/x] \quad 0\Gamma_1 = 0\Gamma_2}{\Gamma_1 + \Gamma_2 \vdash \mathsf{let}_I * = M \text{ in } N \overset{\sigma}{:} U[M/x]}$$

補題 2.3: Zero needs nothing

$\Gamma \vdash M \overset{0}{:} S$ ならば $0\Gamma = \Gamma$

補題 2.4: 弱化

$$\frac{\Gamma, \Gamma' \vdash \mathcal{J} \quad 0\Gamma \vdash U}{\Gamma, x \overset{0}{:} U, \Gamma' \vdash \mathcal{J}}$$

ただし, $\mathcal{J}$ は次のいずれかであるとする: $S, S \equiv T, M \overset{\sigma}{:} S, M \equiv N \overset{\sigma}{:} S$

補題 2.5: 代入では, 文脈と項の双方に対して同時に代入をする規則が認められている.
簡約規則は与えられていないため, 保存や進行といった性質は導けない.

McBride による規則の代入に関する問題

McBride は型判断の右辺も半環の任意の要素を認めたが, この場合代入操作によって半環 $\{0, 1, \omega\}$ が閉じなくなってしまう.
問題は APP 規則において適切に文脈を分割できないケースが生じることである.
次のような型判断を考える.

$$\dfrac{\dfrac{\dfrac{}{f \overset{\omega}{:} (x \overset{1}{:} A) \to A, x \overset{0}{:} A \vdash f \overset{\omega}{:} (x \overset{1}{:} A) \to A}\text{VAR} \quad \dfrac{}{f \overset{0}{:} (x \overset{1}{:} A) \to A, x \overset{\omega}{:} A \vdash x \overset{\omega}{:} A}\text{VAR}}{f \overset{\omega}{:} (x \overset{1}{:} A) \to A, x \overset{\omega}{:} A \vdash fx \overset{\omega}{:} A}\text{APP}}{f \overset{\omega}{:} (x \overset{1}{:} A) \to A \vdash \lambda x \overset{\omega}{:} A.fx \overset{\omega}{:} (x \overset{\omega}{:} A) \to A}\text{LAM}$$

$$\dfrac{\vdash 0\Gamma, x \overset{\sigma}{:} S, 0\Gamma'}{0\Gamma, x \overset{\sigma}{:} S, 0\Gamma' \vdash x \overset{\sigma}{:} S}(\text{VAR}) \quad T, \quad \dfrac{\Gamma_1 \vdash M \overset{\sigma}{:} (x \overset{\pi}{:} S) \to T \quad \Gamma_2 \vdash N \overset{\sigma'}{:} S, \quad 0\Gamma_1 = 0\Gamma_2, \quad \sigma' = 0 \Leftrightarrow (\pi = 0 \vee \sigma = 0)}{\Gamma_1 + \pi\Gamma_2 \vdash \mathsf{App}_{(x \overset{\pi}{:} S)T}(M, N) \overset{\sigma}{:} T[N/x]}(\text{APP}) \quad \dfrac{\Gamma, x \overset{\sigma\pi}{:} S \vdash M \overset{\sigma}{:} T}{\Gamma \vdash \lambda x \overset{\pi}{:} S.M^T \overset{\sigma}{:} (x \overset{\pi}{:} S) \to T}\text{LAM}$$

全スライドからは次の型判断が示された.

$$f \overset{\omega}{:} (x \overset{1}{:} A) \to A \vdash \lambda x \overset{\omega}{:} A.fx \overset{\omega}{:} (x \overset{\omega}{:} A) \to A$$

usage annotation を消せば, $f \to \lambda x.fx$ という η 展開をしているだけだが, 引数の usage が大きくなっている.
しかし, 文脈 $\Gamma \equiv y \overset{\omega}{:} A, f \overset{\omega}{:} (x \overset{1}{:} A) \to A$ において $g = \lambda x \overset{\omega}{:} A.fx$ とおくと $\Gamma \vdash g(gy) \overset{\omega}{:} A$ だが, $\Gamma \not\vdash f(fy) \overset{\omega}{:} A$ となり, β 簡約で $(\lambda x : A.t)u$ という関数適用を代入 $t[u/x]$ に置き換えると型の導出可能性が変わってしまう.

何が f と g の違いを生んだのか?

λ 抽象がある場合は LAM 規則により引数をいったん左辺の文脈に送ることができる. McBride の体系では, APP 規則で引数として適用される項の注釈を $1 \to \omega$ に緩和することができたので, η 展開をしたとき引数の注釈を前スライドのように変換できた.
一方で, λ 抽象のない f そのものの状態では, f の型付けをするのに VAR 規則しか使えないので変数の注釈の変化は起きえない.

- 同様の問題を起こしそうなのは, APP 規則の $\Gamma_2 \vdash N \overset{\sigma'}{:} S$ 部分で σ' が $1 \to \omega$ となるケースだけなので, 型判断の右辺の注釈を $0, 1$ に制限することで Atkey は体系で代入が可能になるようにした.
- 注釈に関する部分型関係を用いた変換 sub usaging が陽に認められていなくても APP 規則により暗黙にできてしまっていたことにより問題が生じた.
- そもそも右辺には数量の注釈をつけないという方法により, 問題を回避する QTT の亜種もある. そちらでは陽に sub usaging の規則があり, 数量的制限の意味もリソースの利用回数の上限と下限を与えるものになっている.
- APP 規則で適用される項と λ 抽象の引数の注釈が一致する場合のみ認め, 暗黙の sub usaging を排除するほうが自然に思えるが, そうしない理由があまりわかっていない.