Preliminaries

myuon

2017 年 10 月 12 日

目次

概要

他のノートを読むにあたって前提とされていることの一部をここにまとめておく.

1 Categorical Semantics

type theory とはその名の通り “型” とは何かということを定めたものである. 以下では項の文法としてラムダ計算を取り扱うことにして, あまり type theory と型付きラムダ計算を区別せずに話をするが, 必ずしも type theory は型付きラムダ計算を指すわけではない.

1.1 Informal Definitions

type theory の定義や定式化はものによって様々なので formal な定義を与えるのは難しい. よってここでは, 出来る限り標準的な ($\neq$ 一般的な)type theory について言えることを述べることにする. 以下の定義が上手く機能しないようなものを考えるときは適宜定義を修正して考えることになる.

定義 1. type theory T とは, type, term, そしてそれらを関係づける rule からなるものである. それぞれ次のような代表的な定め方がある:

- type formation rule: 型コンストラクタが満たすべきルールを $\vdash A : \mathrm{Type}$ のようにして記述したもの. 例えば, function type constructor $\to$ が満たすべきルールは,

$$\vdash A : \mathrm{Type}, \vdash B : \mathrm{Type} \Longrightarrow \vdash A \to B : \mathrm{Type}$$

  のようになる. ただし, 型は型コンストラクタによって自由生成されたもの (特別な制約がないもの) が多いので, そのような場合には $A ::= A_1 \to A_2$ のように BNF で書いてしまっても十分なことが多い.
- term introduction/elimination rule: 項コンストラクタが満たすべきルールを, 各型コンストラクタに対してそれを導入する (結論に用いられる) 規則とそれを除去する (前提に用いられる) 規則からなる (両方あるとは限らない). これらはいずれも項コンストラクタの導入規則であることに注意. 例として, function type constructor$\to$ の intro/elim ルールは, 次のような

abs/app ルールである:

$$\text{abs(funI)}\ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

$$\text{app(funE)}\ \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

- conversion rule: 項の equality を external に与えるようなルールで, これが実際にラムダ計算に構造を与える. type system によっては項の equality を internal に定義できる (実際に type system 上で equal であることを証明できる) 場合があるが, ここでいう conversion rule はそのようなものは含まない. あくまで external に与えられたルールを指していう. 例として, function type $\to$ に関するルールとして次の β/η ルールがある:

$$(\beta) \quad \vdash (\lambda x.\, M)N = M[N/x]$$
$$(\eta) \quad \vdash (\lambda x.\, Mx) = M$$

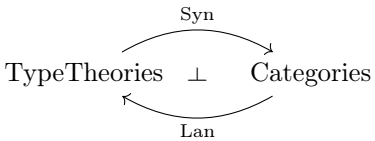
- computation rule: 項の計算・評価を行う際に用いられるルールで, これは rule と言っているが項を別の項へ変換する関数である. 主にラムダ計算をプログラミング言語として見る場合に定義される. 例として, function type $\to$ に関する computation rule として次のような β-reduction ルールがある:

$$(\lambda x.\, M)N \rightsquigarrow M[N/x] \qquad (\beta)$$

定義 2. type theory T に対して, 次のようにして与えた category を T の **syntactic category** とよび, Syn(T) とかく.

- object: T の context $\Gamma = \{x_1 : A_1, \ldots, x_n : A_n\}$
- hom(Γ,$\{\vec{y_i} : \vec{B_i}\}$): 項の集合 $\{t_1, \ldots t_n\}$ であって, 各 i に対して typing judgement $\Gamma \vdash_T t_i : B_i$ が存在するもの.

定義 3. 適当な type theory のなす category TypeTheories と適当な category のなす category Categories が与えられているとする. **categorical semantics** とは, 次の形の随伴 (圏同値) のことである:



ここで, Syn は syntactic category(を与える functor), Lan は internal language のことである.

定義 4. type theory T の (categorical) **モデル (model)** とは, category $\mathbb{C}$ と morphism Syn(T) $\to \mathbb{C}$ のことをいう. この morphism を T の $\mathbb{C}$ での **interpretation** とよぶ. また, 単に $\mathbb{C}$ のことをモデルということもある.

T と Syn(T) を自然に同一視して, T から $\mathbb{C}$ への変換を interpretation と呼ぶことも多い. この場合は T の context を object へ, T の judgement を morphism へ対応させるような変換である.

1.2 Typed Lambda Calculus

定義 5. ラムダ計算 λ の **equational theory** とは, λ で証明可能な等式のなす集合のことである: $\mathcal{E} = \{\vdash_\lambda M = N;\ M \text{ と } N \text{ は } \lambda \text{ で well-formed}\}$.

ラムダ計算での等しさの証明可能性とは, 例えば項の equality が conversion rule によって与えられている場合は conversion rule が生成する等式全体になる. 通常 equational theory は, 同値関係の定義である (refl), (sym), (trans) と,

$$\text{(subst)} \quad \vdash M = N \Longrightarrow \vdash L[M/x] = L[N/x]$$

を含めるようにして定義する. 通常 (subst) は, 項のコンストラクタごとに専用の equation を入れる必要がある.

また, 型付きラムダ計算の場合は, well-formed の条件に well-typed(型付け可能である) を要請する.

1.3 Properties

定義 6. ラムダ計算 λ で証明可能な命題 φ が **sound** とは, λ の任意のモデル $\mathbb{C}$(と morphism $m : \mathrm{Syn}(\lambda) \to \mathbb{C}$) に対し, $m(\bar{\varphi})$ が $\mathbb{C}$ で真となることである. ただし, φ の $\mathrm{Syn}(\lambda)$ での対応する命題を $\bar{\varphi}$ とかいた. また, この逆が成り立つ時, φ は **complete** という.

上の φ として equational theory(項の equality) をとったものをよく使うので, もう一度述べなおしておく.

定義 7. ラムダ計算 λ の equational theory $\mathcal{E}$ が **sound** とは, $\vdash_\lambda M = N \in \mathcal{E}$ のとき, λ の任意のモデル $\mathbb{C}$ と interpretation $[\![-]\!] : \lambda \to \mathbb{C}$ に対して $[\![M]\!] = [\![N]\!]$ が $\mathbb{C}$ で成り立つことである. また, この逆を **complete** という.

上はラムダ計算だけでなくその equational theory を指定して初めて意味をなす命題であることに注意.

また, 個々の interpretation に関して sound/complete という言い方もする.

定義 8. interpretation $[\![-]\!] : \lambda \to \mathbb{C}$ が **sound** とは, $\vdash M = N \in \mathcal{E}$ のとき, $[\![M]\!] = [\![N]\!]$ が $\mathbb{C}$ で成り立つことである. また, この逆を **complete** という.

equational theory が sound ならば, interpretation(model) は sound である.

1.4 Term Model

定義 9. product type constructor $\times$ と term constructor $\mathrm{pr}_i(-)$ をもつような型付きラムダ計算 λ に対して, 次のように category $\mathbb{C}_\lambda$ を定める:

- object: λ の型
- hom(A,B): typing judgement $x : A \vdash M : B$ を, λ の equational theory で割った同値類

$[\![-]\!]_\lambda : \mathrm{Syn}(\lambda) \to \mathbb{C}_\lambda$ を, $[\![\{x_1 : A_1, \ldots, x_n : A_n\}]\!]_\lambda = A_1 \times \cdots \times A_n$, $[\![\{x_1 : A_1, \ldots, x_n : A_n\} \vdash t : B]\!]_\lambda = [z : A_1 \times \cdots \times A_n \vdash t[\mathrm{pr}_i(z)/x_i] : B]$ と定める. これが λ のモデルになる時, このモデルを **term model** という.

実際には, これが category になりモデルを与えることは今考えているラムダ計算の定義に戻って示す必要がある.

補題 10. $[\![M]\!] = [\![N]\!]$ が *term model* $\mathbb{C}_\lambda$ で成り立つならば, $\vdash_\lambda M = N$ である.

term model の interpretation $[\![-]\!]_\lambda$ は必ず complete になる. また, term model は通常最も構造の少ないモデルであるので, 存在すれば equational theory も自然と complete になる.

参考文献

[1] nLab “type theory", https://ncatlab.org/nlab/show/type+theory