1 動機

次のような性質を持つ型理論が欲しい。

- より自然に property に関する subtyping が使える
  - 2 が自然数でもあり偶数でもある。Coq の場合は 2 と 2 が偶数であることの証明の組が偶数として型付けされる。
  - 部分集合が本当に部分集合になり、キャストが簡単（書かなくていい）
  - 結果として型付けの一意性はないと思うけど、それでもいい
- 証明項を真に区別する必要がない or 証明項を扱うことができない
  - 群が等しいとは群の演算が等しいこと、証明項まで等しいこととみなしたくない
  - 証明項を構成することもできるが、それの存在を覚えておくだけぐらいでいい
  - あと関数の外延性などの axiom をいい感じにしたい
- 構造に関する部分型（？）も使えると楽
  - 環は群の部分型とみなしたい（キャストを明示的に書きたくない）
  - これをやると部分空間の扱いが絶対にめんどくさい
  - 公称型みたいな感じで扱った方がいいかも
- 等式をもっと簡単に扱いたい、　well-definedness をもっと簡単に
  - 例として、商群からの写像の扱いが Coq ではめんどくさい
  -（部分集合系が扱えると良いなあ）

2 ラムダ計算復習

ラムダ計算の復習を書いておく。Church 流（？）でコンストラクタみたいなやつをいちいち書く。少なくとも正確ではないし、多分こう考えたら普通の奴と同じになるだろう、みたいなのがいっぱいある。$\langle hoge; t\rangle$ とかいたら、$\langle hoge\rangle$ の元を a_i のように表すということ。

■型なしラムダ計算　ラムダ項の定義としては

項の定義

$\langle term\ ;\ t\rangle ::= \langle variable\ ;\ x\rangle$
　| ‘Fun’ $\langle variable\rangle\ \langle term\rangle$
　| ‘App’ $\langle term\rangle\ \langle term\rangle$

これについては型理論はないが、ラムダ項にどういうものが定義されていたかと言うと、α 変換（ラムダ項の間の関係かラムダ項の変換）・ β 変換（ラムダ項の間の関係かラムダ項の変換）・ reduction （ラムダ項の間の関係）・値（ラムダ項の性質）・正規形（ラムダ項の性質）などがあった。合流性やらが成り立つのだった。項 $t_1\{x \leftarrow t_2\}$ のような代入が定義されている。項の間の同値関係 $t_1 \equiv t_2$ を次の規則から生成する。

ラムダ項の等式

$$\frac{x_2 \notin \mathrm{FV}(t_2)}{\mathtt{Fun}\, x_1\, t_1 \equiv \mathtt{Fun}\, x_2\, (t_1\{x_1 \leftarrow t_2\})}\ \text{alpha}$$

$$\frac{x \notin \mathrm{FV}(t_2)}{\mathtt{App}\, (\mathtt{Fun}\, x\, t_1)\, t_2 \equiv t_2\{x \leftarrow t_2\}}\ \text{computation}$$

$$\frac{t_1 \equiv t_2}{\mathtt{Fun}\, x\, t_1 \equiv \mathtt{Fun}\, x\, t_2}\ \text{conversion-fun}$$

$$\frac{t_1 \equiv t_3}{\mathtt{App}\, t_1\, t_2 \equiv \mathtt{App}\, t_3\, t_2}\ \text{conversion-app-1}$$

$$\frac{t_2 \equiv t_3}{\mathtt{App}\, t_1\, t_2 \equiv \mathtt{App}\, t_1\, t_3}\ \text{conversion-app-2}$$

α, β 変換はこの同値と非常にうまくいくことが（定理としてであって定義ではなく）わかる。conversion-fun は関数型言語で作るときには人によっては使わないかも（スタックマシンへのコンパイルの場合は、値としての関数は変に reduction せず写すなどするため？）また、これに

$$\frac{}{t \equiv \mathtt{Fun}\, x\, (\mathtt{App}\, t\, x)}\ \text{computation-eta}$$

を付け加えることもあるが、外延的ラムダ計算という別の体系になる。

■単純型付きラムダ計算　ラムダ計算に型をつける。このとき、型に加えて新しくコンテキストというものが定義される。

項、型、コンテキストの定義

$\langle type\ ;\ T\rangle ::= \langle type\text{-}variable\ ;\ X\rangle$
　|　‘`Arr`’ $\langle type\rangle\ \langle type\rangle$

$\langle term\ ;\ t\rangle ::= \langle term\text{-}variable\ ;\ x\rangle$
　|　‘`Fun`’ $\langle variable\rangle\ \langle type\rangle\ \langle term\rangle$
　|　‘`App`’ $\langle term\rangle\ \langle term\rangle$

$\langle context\ ;\ \Gamma\ \rangle ::= \text{empty} \mid \langle context\rangle\ ,\ \langle term\text{-}variable\rangle : \langle type\rangle$

項に対しては相変わらず単純に代入が定義された。また、型があるけれども β 変換やらは普通に定義される。項の上の $\equiv$ も同じように定義してしまって構わない。コピペで持ってくる。一方で型付けと言う $\Gamma \vdash t : T$ なる新しい要素が増えた。

型付け規則

$$\frac{}{\Gamma, x : T \vdash x : T}\ \text{variable} \qquad \frac{\Gamma \vdash x_2 : T_2}{\Gamma, x_1 : T_1 \vdash x_2 : T_2}\ \text{weakning}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \mathtt{Fun}\, x\, T_1\, t : \mathtt{Arr}\, T_1\, T_2}\ \text{introduction} \qquad \frac{\Gamma \vdash t_1 : \mathtt{Arr}\, T_1\, T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash \mathtt{App}\, t_1\, t_2 : T_2}\ \text{elimination}$$

ラムダ項の等式

$$\frac{}{\mathtt{Fun}\, x_1\, t_1 \equiv \mathtt{Fun}\, x_2\, (t_1\{x_1 \leftarrow t_2\})}\ \text{alpha} \qquad \frac{}{\mathtt{App}\, (\mathtt{Fun}\, x\, t_1) t_2 \equiv t_2\{x \leftarrow t_2\}}\ \text{computation}$$

$$\frac{t_1 \equiv t_2}{\mathtt{Fun}\, x\, t_1 \equiv \mathtt{Fun}\, x\, t_2}\ \text{conversion-intro} \qquad \frac{t_1 \equiv t_3 \quad t_2 \equiv t_4}{\mathtt{App}\, t_1\, t_2 \equiv \mathtt{App}\, t_3\, t_4}\ \text{conversion-elim}$$

この体系はいい意味で強くて、型保存性や強正規化定理などがある。おそらく「 $\Gamma \vdash t_1 : T, t_1 \equiv t_2$ なら $\Gamma \vdash t_2 : T$ 」が成り立ちそうだ。これが成り立つ場合なら、同値性と型付け可能性は別々に定義してしまって構わないと思ってよいが、後でこの二つが強く関係してくるような体系ができる。

type-check は decidable らしい。型を計算することもできて、 $F(\Gamma, t)$ なる (option 型) を返す関数が作れて $F(\Gamma, t) : \mathrm{Some}\, T$ なら $\Gamma \vdash t : T$ であり、そうでなければ $\Gamma \vdash t : T$ なる T がない？

■単純型付きラムダ計算の拡張達　あとで書く。積、和、 Unit 、 Void を足すことで Curry-Howard がいい感じに。自然数、真偽値、リストを足すことでプログラムっぽくなる。全部 Inductive に作れる。

3 型理論 0

ただ単に証明木 + refinement type を扱えるようにしただけの体系を考える。subtyping はない。問題点としては、もしシステムとして実装するなら証明木を対話的に組んでいく必要があってめんどくさそう。なので、この方向は使わない（問題点を示すためのもの）

項やコンテキストの定義

$\langle term \rangle ::= \langle variable \rangle$
| ‘`Prop`’
| ‘`Type`’
| ‘`Fun`’ $\langle variable \rangle$ $\langle term \rangle$ $\langle term \rangle$
| ‘`Forall`’ $\langle variable \rangle$ $\langle term \rangle$ $\langle term \rangle$
| ‘`Apply`’ $\langle term \rangle$ $\langle term \rangle$
| ‘`Refined`’ $\langle term \rangle$ $\langle term \rangle$

$\langle context\text{-}snippet \rangle ::= \langle variable \rangle$ ‘`:`’ $\langle term \rangle$
| Hold $\langle term \rangle$

$\langle context \rangle ::=$ ‘`empty`’ | $\langle context \rangle$ ‘`,`’ $\langle context\text{-}snippet \rangle$

`Refined` が refinement type の型。コンテキストの Hold $\langle term \rangle$ は命題の仮定を表す。$\langle variable \rangle$ を x 、$\langle term \rangle$ を t や A や P 、$\langle context \rangle$ を Γ のように書く。t と A は同じ項だが、気持ちとしては項と型の分け方である。また `Sort` は `Prop` か `Type` を表す。

コンテキストと項の関係について。直観的にはコンテキストの well-def 性、型付け可能性、証明可能、を表す。

関係一覧

$\vdash \Gamma$
$\Gamma \vdash t_1 : t_2$
$\Gamma \vDash t$

これが関係の全部。自由変数とか subst とかはめんどくさいので書いてない。 conversion rule も定義していないが、 $M_1 \equiv_\beta M_2$ みたいなのが定義されていてほしい。Type : Type を避けるためにいろいろ書いている？コンテキストの well-defined 性は次のようになる。

$$\frac{}{\vdash empty} \text{ context empty}$$

$$\frac{\vdash \Gamma \quad \Gamma \vdash A : \texttt{Type} \quad x \notin \mathrm{FV}(\Gamma)}{\vdash \Gamma, x : A} \text{ context type}$$

$$\frac{\vdash \Gamma \quad \Gamma \vdash P : \texttt{Prop}}{\vdash \Gamma, \mathrm{Hold}\, P} \text{ context prop}$$

Type と Prop に型付けされる項は次のようになる（ただし、 large elimination に相当するものをのぞく）

$$\frac{\Gamma \vdash A_1 : \texttt{Type} \quad \Gamma, x : A_1 \vdash A_2 : \texttt{Type}}{\Gamma \vdash \texttt{Forall}x\texttt{of}A_1\texttt{to}A_2 : \texttt{Type}} \text{ forall formation(type)}$$

$$\frac{\Gamma \vdash A_1 : \texttt{Type} \quad \Gamma, x : A_1 \vdash A_2 : \texttt{Prop}}{\Gamma \vdash \texttt{Forall}x\texttt{of}A_1\texttt{to}A_2 : \texttt{Prop}} \text{ forall formation(prop)}$$

$$\frac{\Gamma \vdash A : \texttt{Type} \quad \Gamma \vdash P : \texttt{Forall}x\texttt{of}A\texttt{toProp}}{\Gamma \vdash \texttt{Refined}A\texttt{with}P : \texttt{Type}} \text{ refinement formation}$$

以下は β 同値に関するもの。application や conversion により 下二つはいらないかもしれないし、逆に強すぎるかもしれない。(subject reduction がうまくいくのかわからなかったのでとりあえずつけた。 ITT での equality rule みたいなのがないのでバグがありそう)

$$\frac{\Gamma \vdash x : A_1 \quad A_1 \equiv_\beta A_2 \quad \Gamma \vdash A_1 : \texttt{Type}}{\Gamma \vdash x : A_2} \text{ conversion}$$

$$\frac{\Gamma \vdash A_1 : \texttt{Type} \quad A_1 \equiv_\beta A_2}{\Gamma \vdash A_2 : \texttt{Type}} \text{ conversion(type)}$$

$$\frac{\Gamma \vdash A_1 : \texttt{Prop} \quad A_1 \equiv_\beta A_2}{\Gamma \vdash A_2 : \texttt{Prop}} \text{ conversion(prop)}$$

より一般の項の型付けは次のようになる。

$$\frac{\vdash \Gamma \quad \Gamma \vdash A : \texttt{Type}}{\Gamma, x : A \vdash x : A} \text{ start}$$

$$\frac{\vdash \Gamma, x : A_1 \quad \Gamma \vdash t : A_2}{\Gamma, x : A_1 \vdash t : A_2} \text{ weakning}$$

$$\frac{\Gamma, x : A_1 \vdash t : A_2 \quad \Gamma \vdash \texttt{Forall}x\texttt{of}A_1\texttt{to}A_2 : \texttt{Type}}{\Gamma \vdash \texttt{Fun}x\texttt{of}A_1\texttt{mapsto}t : \texttt{Forall}x\texttt{of}A_1\texttt{to}A_2} \text{ forall introduction}$$

$$\frac{\Gamma \vdash t_1 : \texttt{Forall}x\texttt{of}A_1\texttt{to}A_2 \quad \Gamma \vdash t_2 : A_1}{\Gamma \vdash \texttt{Apply}t_1\texttt{and}t_2 : A_2\{x \leftarrow t_2\}} \text{ forall elimination}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash \texttt{Refined}A\texttt{with}P : \texttt{Type} \quad \Gamma \vDash (\texttt{Apply}P\texttt{and}t)}{\Gamma \vdash t : \texttt{Refined}A\texttt{with}P} \text{ refinement introduction}$$

$$\frac{\Gamma \vdash t : \texttt{Refined}A\texttt{with}P}{\Gamma \vdash t : A} \text{ refinement elimination}$$

次に証明木とか証明周りのやつを書く。

$$\frac{\Gamma \vdash P_1 : \texttt{Prop} \quad P_1 \equiv_\beta P_2 \quad \Gamma \vDash P_1}{\Gamma \vDash P_2} \text{ prop conversion}$$

$$\frac{\Gamma \vdash \texttt{Forall}x\texttt{of}A\texttt{to}P : \texttt{Prop} \quad \Gamma, x : A \vDash \texttt{Apply}P\texttt{and}x}{\Gamma \vDash \texttt{Forall}x\texttt{of}A\texttt{to}P} \text{ prop forall intro}$$

$$\frac{\Gamma \vDash \texttt{Forall}x\texttt{of}A\texttt{to}P \quad \Gamma \vdash t : A}{\Gamma \vDash P\{x \leftarrow t\}} \text{ prop forall elim}$$

$$\frac{\Gamma \vdash x : \texttt{Refined}A\texttt{with}P}{\Gamma \vDash \texttt{Apply}P\texttt{and}x} \text{ prop refinement elim}$$

これで終わり。この型理論を検討することはないと思うが、例を後で挙げたい。Prop と Type で別々の型付けが必要になることがありめんどくさい。