

## 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ある 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

$$\begin{aligned} \langle term ; t \rangle &::= \langle variable ; x \rangle \\ &| \text{ 'Fun' } \langle variable \rangle \langle term \rangle \\ &| \text{ 'App' } \langle term \rangle \langle term \rangle \end{aligned}$$

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、 $\alpha$  変換 (ラムダ項の間の関係かラムダ項の変換)・ $\beta$  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

$$\begin{aligned} &\frac{x_2 \notin \text{FV}(t_2)}{\text{Fun } x_1 t_1 \equiv \text{Fun } x_2 (t_1\{x_1 \leftarrow t_2\})} \text{ alpha} \\ &\frac{x \notin \text{FV}(t_2)}{\text{App } (\text{Fun } x t_1) t_2 \equiv t_2\{x \leftarrow t_1\}} \text{ computation} \\ &\frac{t_1 \equiv t_2}{\text{Fun } x t_1 \equiv \text{Fun } x t_2} \text{ conversion-fun} \\ &\frac{t_1 \equiv t_3}{\text{App } t_1 t_2 \equiv \text{App } t_3 t_2} \text{ conversion-app-1} \\ &\frac{t_2 \equiv t_3}{\text{App } t_1 t_2 \equiv \text{App } t_1 t_3} \text{ conversion-app-2} \end{aligned}$$

$\alpha, \beta$  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

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

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

$$\begin{aligned} \langle type ; T \rangle &::= \langle type\text{-}variable ; X \rangle \\ &| \text{'Arr'} \langle type \rangle \langle type \rangle \\ \langle term ; t \rangle &::= \langle term\text{-}variable ; x \rangle \\ &| \text{'Fun'} \langle variable \rangle \langle type \rangle \langle term \rangle \\ &| \text{'App'} \langle term \rangle \langle term \rangle \\ \langle context ; \Gamma \rangle &::= \text{empty} | \langle context \rangle , \langle term\text{-}variable \rangle : \langle type \rangle \end{aligned}$$

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  $\beta$  変換や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

$$\begin{aligned} &\frac{}{\Gamma, x : T \vdash x : T} \text{variable} \quad \frac{\Gamma \vdash x_2 : T_2}{\Gamma, x_1 : T_1 \vdash x_2 : T_2} \text{weakening} \\ &\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \text{Fun } x T_1 t : \text{Arr } T_1 T_2} \text{introduction} \quad \frac{\Gamma \vdash t_1 : \text{Arr } T_1 T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash \text{App } t_1 t_2 : T_2} \text{elimination} \end{aligned}$$

#### ラムダ項の等式

$$\begin{aligned} &\frac{}{\text{Fun } x_1 t_1 \equiv \text{Fun } x_2 (t_1 \{x_1 \leftarrow t_2\})} \text{alpha} \quad \frac{}{\text{App } (\text{Fun } x t_1) t_2 \equiv t_2 \{x \leftarrow t_1\}} \text{computation} \\ &\frac{t_1 \equiv t_2}{\text{Fun } x t_1 \equiv \text{Fun } x t_2} \text{conversion-intro} \quad \frac{t_1 \equiv t_3 \quad t_2 \equiv t_4}{\text{App } t_1 t_2 \equiv \text{App } t_3 t_4} \text{conversion-elim} \end{aligned}$$

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。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  $F(\Gamma, t) : \text{Some } T$  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 1

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。証明項というものを完全になくそうとした場合、証明を組むのは証明項ではなく証明木を作ることになる問題点としては、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。一応書いておいただけ。

項やコンテキストの定義

```

<term> ::= <variable>
| 'Prop'
| 'Type'
| 'Fun' <variable> <term> <term>
| 'Forall' <variable> <term> <term>
| 'Apply' <term> <term>
| 'Refined' <term> <term>

<context-snippet> ::= <variable> ':' <term>
| Hold <term>

<context> ::= 'empty' | <context> ',' <context-snippet>

```

Refined が refinement type の型。コンテキストの Hold <term> は命題の仮定を表す。<variable> を  $x$ 、<term> を  $t$  や  $A$  や  $P$ 、<context> を  $\Gamma$  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

```

⊢ Γ
Γ ⊢ t1 : t2
Γ ⊢ 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。

コンテキストの well-define

$$\begin{array}{c}
\frac{}{\vdash \text{empty}} \text{ context empty} \\
\\
\frac{\vdash \Gamma \quad \Gamma \vdash A : \text{Type} \quad x \notin \text{FV}(\Gamma)}{\vdash \Gamma, x : A} \text{ context type} \\
\\
\frac{\vdash \Gamma \quad \Gamma \vdash P : \text{Prop}}{\vdash \Gamma, \text{Hold } P} \text{ context prop}
\end{array}$$

型付け (Type と Prop)

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

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

$$\frac{\Gamma \vdash A : \text{Type} \quad \Gamma \vdash P : \text{Forall } x A \text{Prop}}{\Gamma \vdash \text{Refined } A P : \text{Type}} \text{ refinement formation}$$

Prop から Type を作るのは禁止したいので型付けられないようにしてある

以下は  $\beta$  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ととりあえずつけた。ITT での equality rule みたいなのがないのでバグがありそう) 変な問題があっても直す気はない。

conversion

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

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

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

より一般の項の型付けは次のようになる。

typing

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

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

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

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

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

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

次に証明木とか証明周りのやつを書く。「証明項をもって証明のかわりとする」規則を入れてなかったので色々増える。

Proof

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

$$\frac{\Gamma \vdash \mathbf{Forall} \, x \, A \, P : \mathbf{Prop} \quad \Gamma, x : A \models \mathbf{Apply} \, P \, x}{\Gamma \models \mathbf{Forall} \, x \, A \, P} \text{ prop forall intro}$$

$$\frac{\Gamma \models \mathbf{Forall} \, x \, A \, P \quad \Gamma \vdash t : A}{\Gamma \models P\{x \leftarrow t\}} \text{ prop forall elim}$$

$$\frac{\Gamma \vdash x : \mathbf{Refined} \, A \, P}{\Gamma \models \mathbf{Apply} \, P \, x} \text{ prop refinement elim}$$

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。

## 4 型理論 1

証明項に対応するものを微妙に導入して、型システムが自動的に判定できる証明を増やした形。

項やコンテキストの定義

$$\begin{aligned}
 \langle term \rangle &::= \langle variable \rangle \\
 &| \text{ 'Prop' } \\
 &| \text{ 'Type' } \\
 &| \text{ 'Fun' } \langle variable \rangle \langle term \rangle \langle term \rangle \\
 &| \text{ 'For' } \langle variable \rangle \langle term \rangle \langle term \rangle \\
 &| \text{ 'App' } \langle term \rangle \langle term \rangle \\
 &| \text{ 'Ref' } \langle term \rangle \langle term \rangle \\
 &| \text{ 'Prf' } \langle term \rangle \\
 \langle context-snippet \rangle &::= \langle variable \rangle \text{ ':' } \langle term \rangle \\
 &| \text{ Hold } \langle term \rangle \\
 \langle context \rangle &::= \text{ 'empty' } | \langle context \rangle \text{ ',' } \langle context-snippet \rangle
 \end{aligned}$$

Proof は証明項を陽に扱うためのもの。命題  $P$  が示せるときに Proof  $P$  を証明項として扱ってよいとすることで簡単にならないだろうか？あるいは、(証明が必要となるところでのみ用いたい)

項やコンテキストの評価 ここでコンテキストや項の関係を定義していく。

コンテキストと項の関係

$\vdash \Gamma$  , コンテキストの well-def 性  
 $\Gamma \vdash t_1 : t_2$  , 項の型付け性  
 $\Gamma \vdash t$  , 項の証明可能性

コンテキストの well-def

$$\begin{aligned}
 &\frac{}{\vdash \text{empty}} \text{ context empty} \\
 &\frac{\vdash \Gamma \quad x \notin \text{FV}(\Gamma)}{\vdash \Gamma, x : \text{Sort}} \text{ context type} \quad \frac{\vdash \Gamma \quad \Gamma \vdash A : \text{Sort} \quad x \notin \text{FV}(\Gamma)}{\vdash \Gamma, x : A} \text{ context term} \\
 &\frac{\vdash \Gamma \quad \Gamma \vdash P : \text{Prop}}{\vdash \Gamma, \text{Hold } P} \text{ context prop}
 \end{aligned}$$

コンテキスト 自明

$$\begin{aligned}
 &\frac{\vdash \Gamma \quad x \notin \text{FV}(\Gamma)}{\Gamma, x : \text{Sort} \vdash x : \text{Sort}} \text{ variable-type} \quad \frac{\vdash \Gamma \quad \Gamma \vdash A : \text{Type} \quad x \notin \text{FV}(\Gamma)}{\Gamma, x : A \vdash x : A} \text{ variable-term} \\
 &\frac{\vdash \Gamma, \_ \quad \Gamma \vdash t : A_2}{\Gamma, \_ \vdash t : A_2} \text{ weakning}
 \end{aligned}$$



formation

$$\frac{\Gamma \vdash A_1 : \text{Type} \quad \Gamma, x : A_1 \vdash A_2 : \text{Type}}{\Gamma \vdash \text{For } x A_1 A_2 : \text{Type}} \text{ forall formation(type-type)}$$

$$\frac{\Gamma \vdash A_1 : \text{Type} \quad \Gamma, x : A_1 \vdash A_2 : \text{Prop}}{\Gamma \vdash \text{For } x A_1 A_2 : \text{Prop}} \text{ forall formation(type-prop)}$$

$$\frac{\Gamma \vdash A_1 : \text{Prop} \quad \Gamma, x : A_1 \vdash A_2 : \text{Prop}}{\Gamma \vdash \text{For } x A_1 A_2 : \text{Prop}} \text{ forall formation(prop-prop)}$$

$$\frac{\Gamma \vdash A : \text{Type} \quad \Gamma \vdash P : \text{For } x A \text{Prop}}{\Gamma \vdash \text{Ref } A P : \text{Type}} \text{ refinement formation}$$

Prop の項から Type つくるの禁止 (型で制限)

$\beta$  同値について

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

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

下二つは application と conversion があればいらないかもしれない。必要になったら足す感じで。

introduction と elimination

$$\frac{\Gamma, x : A_1 \vdash t : A_2 \quad \Gamma \vdash \text{For } x A_1 A_2 : \text{Type}}{\Gamma \vdash \text{Fun } x A_1 t : \text{For } x A_1 A_2} \text{ for intro} \quad \frac{\Gamma \vdash t_1 : \text{For } x A_1 A_2 \quad \Gamma \vdash t_2 : A_1}{\Gamma \vdash \text{App } t_1 t_2 : A_2\{x \leftarrow t_2\}} \text{ for elim}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash \text{Ref } A P : \text{Type} \quad \Gamma \models \text{App } P t}{\Gamma \vdash t : \text{Ref } A P} \text{ ref intro} \quad \frac{\Gamma \vdash t : \text{Ref } A P}{\Gamma \vdash t : A} \text{ ref elim}$$

proof term について

$$\overline{\Gamma, \text{Hold } P \models P} \text{ assumption} \quad \frac{\Gamma \models P}{\Gamma, \_ \models P} \text{ weakning}$$

$$\frac{\Gamma \vdash P : \text{Prop} \quad \Gamma \vdash t : P}{\Gamma \models P} \text{ implicit proof} \quad \frac{\Gamma \vdash P : \text{Prop} \quad \Gamma \models P}{\Gamma \vdash \text{Prf } P : P} \text{ explicit proof}$$

$$\frac{\Gamma \vdash t : \text{Ref } A P}{\Gamma \models \text{App } P t} \text{ refinement inversion}$$

$$\frac{\Gamma \models P_1 \quad P_1 \equiv_{\beta} P_2}{\Gamma \models P_2} \text{ proof conversion}$$

Proof conversion は多分いらない。explicit proof と conversion によって  $\text{Proof } P_1$  が  $\text{Proof } P_1 : P_2$  と型付けされる。

具体例を出したい。

- $A : \text{Prop}, B : \text{Prop}, \text{Hold}(\text{for } x A B), \text{Hold } A \models B$
- $A, B : \text{Type}, f : A \rightarrow B$  とする。  $P, Q : A, B$  上の述語に対して「任意の  $x : A$  について、  $P(x)$  が成り立つなら  $Q(f(x))$  が成り立つ」とする。このとき  $f : \text{Ref } A P \rightarrow \text{Ref } B Q$  と型付けられるはず。

一つ目は省略（カーリーハワードと Proof を使う）コンテキストとしては、

$$\Gamma := A : \text{Type}, B : \text{Type}, f : A \rightarrow B, P : A \rightarrow \text{Prop}, Q : B \rightarrow \text{Prop}$$

をまず考え、これに命題に対応する項がくつつく。  $K := \text{For } x A ((\text{App } P x) \rightarrow (\text{App } Q (\text{App } f x)))$  なる  $\text{Prop}$  型の項が「任意の  $x : A$  について、  $P(x)$  が成り立つなら ...」に対応する。改めて  $\Gamma \leftarrow \Gamma, \text{Hold } K$  と置き直して、  $\Gamma \vdash f : \text{Ref } A P \rightarrow \text{Ref } B Q$  を示す。eta-conversion が必要になっちゃった。

$$\frac{\frac{\Gamma_1 \vdash f : A \rightarrow B \quad \Gamma_1 \vdash x : A}{\Gamma_1 \vdash \text{App } f x : B} \text{ for intro} \quad \frac{\vdots}{\Gamma_1 \vdash \text{Ref } B Q : \text{Type}} \text{ hello} \quad \frac{\vdots}{\Gamma_1 \models \text{App } Q (\text{App } f x)} X_1}{\frac{\Gamma_1 := \Gamma, x : \text{Ref } A P \vdash \text{App } f x : \text{Ref } B Q}{\Gamma \vdash \text{Fun } x (\text{App } f x) : \text{Ref } A P \rightarrow \text{Ref } B Q} \text{ for elim} \quad \text{eta-conversion}} \Gamma \vdash f : \text{Ref } A P \rightarrow \text{Ref } B Q$$

$X_1$  は app 省略して

$$\frac{\frac{\frac{\overline{\Gamma_1 \models K} \text{ assumption}}{\Gamma_1 \vdash \text{Prf } K : K} \text{ exp prf}}{\Gamma_1 \vdash ((\text{Prf } K) x) : (P x) \rightarrow (Q (f x))} \text{ for elim} \quad \frac{\frac{\overline{\Gamma_1 \vdash x : \text{Ref } A P} \text{ var term}}{\Gamma_1 \models P x} \text{ ref inv}}{\Gamma_1 \vdash \text{Prf } (P x) : P x} \text{ exp prf}}{\frac{\Gamma_1 \vdash ((\text{Prf } K) x) (\text{Prf } (P x)) : \text{App } Q (\text{App } f x)}{\Gamma_1 \models \text{App } Q (\text{App } f x)} \text{ imp prf}} \text{ for elim}$$

$\_ \vdash \_$  が必要になったときのほとんどは自動的に推論できそう