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 t 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

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

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

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

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。Curry-Howard 対応で見ておもしろくなるものをラムダ計算に付け足す。

項、型、コンテキストの定義

$\langle type \rangle ::= \langle type\text{-}variable \rangle$
| ‘`Arr`’ $\langle type \rangle$ $\langle type \rangle$
| ‘`Prod`’ $\langle type \rangle$ $\langle type \rangle$
| ‘`Sum`’ $\langle type \rangle$ $\langle type \rangle$
| ‘`Unit`’
| ‘`Void`’

$\langle term \rangle ::= \langle term\text{-}variable \rangle$
| ‘`Fun`’ $\langle variable \rangle$ $\langle type \rangle$ $\langle term \rangle$ | ‘`App`’ $\langle term \rangle$ $\langle term \rangle$
| ‘`Pair`’ $\langle term \rangle$ $\langle term \rangle$ | ‘`Pjl`’ $\langle term \rangle$ | ‘`Pjr`’ $\langle term \rangle$
| ‘`Inl`’ $\langle term \rangle$ | ‘`InE`’ $\langle term \rangle$ | ‘`Case`’ $\langle term \rangle$ $\langle variable \rangle$ $\langle term \rangle$ $\langle variable \rangle$ $\langle term \rangle$
| ‘`T`’

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

型付け規則

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

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \texttt{Fun}\, x\, T_1\, t : \texttt{Arr}\, T_1\, T_2}\ \text{Arr-intro} \qquad \frac{\Gamma \vdash t_1 : \texttt{Arr}\, T_1\, T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash \texttt{App}\, t_1\, t_2 : T_2}\ \text{Arr-elim}$$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma \vdash t_2 : T_2}{\Gamma \vdash \texttt{Pair}\, t_1\, t_1 : \texttt{Prod}\, T_1\, T_2}\ \text{Prod-intro} \qquad \frac{\Gamma \vdash t : \texttt{Prod}\, T_1\, T_2}{\Gamma \vdash \texttt{Pjl}\, t : T_1}\ \text{Prod-elim1} \qquad \frac{\Gamma \vdash t : \texttt{Prod}\, T_1\, T_2}{\Gamma \vdash \texttt{Pjr}\, t : T_2}\ \text{Prod-elim2}$$

$$\frac{\Gamma \vdash x : T_1}{\Gamma \vdash \texttt{Inl}\, t : \texttt{Sum}\, T_1\, T_2}\ \text{Sum-intro1} \qquad \frac{\Gamma \vdash x : T_2}{\Gamma \vdash \texttt{Inr}\, t : \texttt{Sum}\, T_1\, T_2}\ \text{Sum-intro2}$$

$$\frac{\Gamma \vdash t : \texttt{Sum}\, T_1\, T_2 \quad \Gamma, x_1 : T_1 \vdash t_1 : T \quad \Gamma, x_2 : T_2 \vdash t_2 : T}{\Gamma \vdash \texttt{Case}\, t\, x_1\, t_1\, x_2\, t_2 : T}\ \text{Sum-elim}$$

$$\frac{}{\Gamma \vdash \texttt{T} : \texttt{Unit}}\ \text{Unit-intro} \qquad \frac{}{\Gamma \vdash}\ \text{Unit-elim}$$

ラムダ項の等式

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

$$\frac{t_1 \equiv t_2}{\mathtt{Fun}\,x\,t_1 \equiv \mathtt{Fun}\,x\,t_2}\ \text{Arr-conv} \qquad \frac{t_1 \equiv t_3}{\mathtt{App}\,t_1\,t_2 \equiv \mathtt{App}\,t_3\,t_2}\ \text{Arr-conv} \qquad \frac{t_2 \equiv t_3}{\mathtt{App}\,t_1\,t_2 \equiv \mathtt{App}\,t_1\,t_3}\ \text{Arr-conv}$$

$$\frac{}{\mathtt{Pj1}\,(\mathtt{Pair}\,t_1 t_2) \equiv t_1}\ \text{Prod-comp1} \qquad \frac{}{\mathtt{Pj2}\,(\mathtt{Pair}\,t_1 t_2) \equiv t_2}\ \text{Prod-comp2}$$

$$\frac{}{\mathtt{Case}\,(\mathtt{Inl}\,t)\,x_1\,t_1\,x_2\,t_2 \equiv t_1\{x_1 \leftarrow t\}}\ \text{Sum-comp1} \qquad \frac{}{\mathtt{Case}\,(\mathtt{Inr}\,t)\,x_1\,t_1\,x_2\,t_2 \equiv t_2\{x_2 \leftarrow t\}}\ \text{Sum-comp1}$$

あと cov がいろいろ

この場合、 Coq と比較すると Product Type の induction が対応しないように見えるため、eliminater に対応するものに変えることが考えられる。こっちの方が match に近くて書きやすい。

項、型、コンテキストの定義

⟨*type*⟩ ::= ⟨*type-variable*⟩
| ‘`Arr`’ ⟨*type*⟩ ⟨*type*⟩
| ‘`Prod`’ ⟨*type*⟩ ⟨*type*⟩
| ‘`Sum`’ ⟨*type*⟩ ⟨*type*⟩
| ‘`Unit`’
| ‘`Void`’

⟨*term*⟩ ::= ⟨*term-variable*⟩
| ‘`Fun`’ ⟨*variable*⟩ ⟨*type*⟩ ⟨*term*⟩
| ‘`App`’ ⟨*term*⟩ ⟨*term*⟩
| ‘`Prod-c`’ ⟨*term*⟩ ⟨*term*⟩
| ‘`Prod-e`’ ⟨*term*⟩
| ‘`Sum-c1`’ ⟨*term*⟩
| ‘`Sum-c2`’ ⟨*term*⟩
| ‘`Sum-e`’ ⟨*term*⟩
| ‘`Unit-c`’
|

⟨*context-snippet*⟩ ::= ⟨*term-variable*⟩ : ⟨*type*⟩

⟨*context*⟩ ::= empty | ⟨*context*⟩ , ⟨*context-snippet*⟩

型付け規則

$$\frac{}{\Gamma, x : T \vdash x : T}\ \text{variable}$$

$$\frac{\Gamma \vdash x_2 : T_2}{\Gamma, x_1 : T_1 \vdash x_2 : T_2}\ \text{weakning}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \texttt{Fun}\, x\, T_1\, t : \texttt{Arr}\, T_1\, T_2}\ \text{Arr-introduction}$$

$$\frac{\Gamma \vdash t_1 : \texttt{Arr}\, T_1\, T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash \texttt{App}\, t_1\, t_2 : T_2}\ \text{Arr-elimination}$$