型なしラムダ計算のまとめ。

項の定義

`<term>`	::=		t , M
	|	`<variable>`	x
	|	`Fun <variable> <term>`	
	|	`App <term> <term>`	

右の奴はメタ変数で表すときにどう書くか

$\langle variable \rangle$ は人が勝手に決めてよいが、等しいかどうか判定できないといけない。具体的には `isEquiv : variable -> variable -> Bool` があっていい性質を満たすことが必要。これを $=$ で書いて場合分けに使う。ラムダ項に対して以下のものが定義されていた。定理の記号は [!] とし、計算は停止しないもの、決定は停止するものに使う。（形式化されているっぽく書いているがこの言語に特に意味論はない。）

- 自由変数 (`freevalueOf`) : `term -> list<variable>`
  := 項の中にある束縛されていない変数を（リストにして）返す
- 単純な代入 (`simpleSubst`) : `term -> variable -> term`
  := 項 M の中に自由変数としてあらわれる変数 x を N に置き換える
  with notation $M\{x \leftarrow N\}$ as $\texttt{simpleSubst}\, M\, x\, N$
- α 同値関係 (`isAlphaEquiv`) ： `term -> term -> Prop`
  := 次で生成される関係

$$\frac{(y \notin \texttt{FV}(M))}{\texttt{Fun}\, x\, M =_\alpha \texttt{Fun}\, y\, (M\{x \leftarrow y\})} \text{ alpha}$$

$$\frac{M_1 =_\alpha M_2}{\texttt{Fun}\, x\, M_1 =_\alpha \texttt{Fun}\, x\, M_2} \text{ Fun conversion}$$

$$\frac{M_1 =_\alpha M_1}{\texttt{App}\, M_1\, N =_\alpha \texttt{App}\, M_1\, N} \text{ App conversion 1}$$

$$\frac{N_1 =_\alpha N_2}{\texttt{App}\, M\, N_1 =_\alpha \texttt{App}\, M\, N_2} \text{ App conversion 2}$$

  with notation $M =_\alpha N$ as $\texttt{isAlphaEquiv}\, M\, N$
- α 同値決定 (`isAlphaEquiv?`) : `term -> term -> Bool`
  := 次のように帰納的に定義
  M_1 と M_2 をとりこの構造に着目し以下のように場合分けする。
  - x_1 と x_2 なら $x_1 = x_2$ に帰着する。
  - $\texttt{Fun}\, x_1\, M_1$ と $\texttt{Fun}\, x_2\, M_2$ なら $M_1 =_\alpha M_2\{x_2 \leftarrow x_1\}$ に帰着する。
  - $\texttt{App}\, M_1\, M_2$ と $\texttt{App}\, N_1\, N_2$ なら $M_1 =_\alpha N_1$ かつ $M_2 =_\alpha N_2$ に帰着する。
  - それ以外は `false` とする。

! α 同値関係は α 同値決定で計算できる。

- 捕縛を回避した代入 (`subst`) : `term -> variable -> term`
  := 項 M の中に自由変数としてあらわれる変数 x を N に置き換えるが、 N が M の他の自由変数を束縛しないようにするもの
  with notation $M[x \leftarrow N]$ as $\texttt{subst}\, M\, x\, N$

- β 変換関係 (`isBetaConversion`) : `term -> term -> Prop`
  := 次で生成される関係

$$\frac{M =_\alpha N}{M =_\beta N}\ \text{from alpha}$$

$$\frac{}{\texttt{App}\,(\texttt{Fun}\,x\,M)\,N \to_\beta M[x \leftarrow N]}\ \text{beta}$$

$$\frac{M \to_\beta N}{\texttt{Fun}\,x\,M \to_\beta \texttt{Fun}\,x\,N}\ \text{Fun conversion}$$

$$\frac{M_1 \to_\beta M_1}{\texttt{App}\,M_1\,N \to_\beta \texttt{App}\,M_1\,N}\ \text{App conversion 1}$$

$$\frac{N_1 \to_\beta N_2}{\texttt{App}\,M\,N_1 \to_\beta \texttt{App}\,M\,N_2}\ \text{App conversion 2}$$

  with notation $M \to_\beta N$ as `isBetaConversion` $M\,N$
- 正規形 (`isNormal`) : `term -> Prop`
  := β 変換関係においてこれ以上変換できないとき
- 正規形決定 (`isNormal?`) : `term -> Bool`
  := 次のように帰納的に定義
  - x なら true
  - `Fun` $x\,M$ なら M に帰着
  - `App` $M\,N$ なら M と N に帰着

！正規形は正規形決定で決定できる

- β 変換 (`conversion`) : `term -> term`
  := 次のように帰納的に定義
  - x ならそのまま
  - `Fun` $x\,M$ なら `Fun` x (`conversion` M)
  - `App` (`Fun` $x\,M$) N なら $M[x \leftarrow N]$
  - それ以外は `App` $M\,N$ と書けるが、
    * M が正規形なら `App` M (`conversion` N)
    * そうでなければ `App` (`conversion` M) N

  注意：このような具体的な関数を作るのは評価戦略を決定することに対応しているため、人によっては β 変換の定義は異なる。

！β 変換は β 変換関係と compatible

- β 簡約関係 (`isBetaReduce`) : `term -> term -> Prop`
  := β 変換関係の推移閉包
- β 簡約、正規形の計算 (`normalize`) : `term -> term`
  := 以下のように再帰的に定義
  - 正規形ならそのまま
  - 正規形でないなら `conversion` したものを `normalize`

！正規形計算は停止しないことがあるが、停止すれば正規形が返ってくる。

- β 同値：`term -> term -> Prop`
  := β 変換の同値閉包。
- β 同値計算：`term -> term -> Bool`
  := 項をそれぞれ `normalize` して α 同値か比べる。

！β 同値計算は停止すれば β 同値の判定になる。

！β 簡約関係は合流性を持つ

項の間の同値関係 $t_1 \equiv t_2$ を次の規則から生成する。これは β 同値関係を含む外延的ラムダ計算と呼ばれる体系になる。

ラムダ項の等式

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

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

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

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

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

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