工学部専門科目「計算と論理」配布資料
多相ラムダ計算 (System F)

五十嵐 淳
京都大学 大学院情報学研究科 通信情報システム専攻
`cal19@fos.kuis.kyoto-u.ac.jp`
`http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/cal/`

November 12, 2019

1 System F: 多相ラムダ計算

教科書 Poly.v で見たような**多相関数** (*polymorphic function*)(型引数をとり，引数によって型が変化する関数) によって単純型付ラムダ計算を拡張した計算体系が**多相ラムダ計算** (*polymorphic λ-calculus*) である．多相ラムダ計算は System F とも呼ばれている．[1]

System F の (単純型付ラムダ計算に対しての) 本質的な拡張は項 (プログラム) への多相関数[2]

$$\texttt{fun}\ \boldsymbol{X} : \texttt{Type}\ \texttt{=>}\ \boldsymbol{M}$$

と型適用 $\boldsymbol{M}\ \boldsymbol{T}$ の導入，そして多相型 $\forall \boldsymbol{X} : \texttt{Type}, \boldsymbol{T}$ の導入である．

(ふつうの) 関数の β 簡約と同様に型抽象が型に適用された項は簡約によって計算が進む．

$$(\texttt{fun}\ \boldsymbol{X} : \texttt{Type}\ \texttt{=>}\ \boldsymbol{M}[\boldsymbol{X}])\ \boldsymbol{T} \longrightarrow \boldsymbol{M}[\boldsymbol{T}]$$

関数本体中の (型の) 仮引数 $\boldsymbol{X}$ が実引数 $\boldsymbol{T}$ で置き換わるところは β 簡約と全く同じである．例えば，与えられた関数 `f` を `x` に二回適用する多相高階関数

$$\boldsymbol{M} \stackrel{\mathrm{def}}{=} \texttt{fun X : Type => fun f : X -> X => fun x : X => f (f x)}$$

について，

$$\begin{array}{rcl} \boldsymbol{M}\ \texttt{nat} & \longrightarrow & \texttt{fun f : nat -> nat => fun x : nat => f (f x)} \\ \boldsymbol{M}\ \texttt{bool} & \longrightarrow & \texttt{fun f : bool -> bool => fun x : bool => f (f x)} \end{array}$$

となり，(機能は同じだが) 自然数，真偽値に特化した関数が得られる．

---

[1] アメリカの計算機科学者ジョン・レイノルズ (John Reynolds) とフランスの論理学者ジャン・イヴ・ジラール (Jean-Yves Girard) が 1970 年代初めに独立に考え出した．System F の名前はジラールによる．

[2] 型抽象とも呼ぶ

型付け関係は単純型付ラムダ計算と同様に

$$\boldsymbol{\Gamma} \vdash \boldsymbol{M} : \boldsymbol{T}$$

という形で表されるが，型環境には変数の型宣言 $\boldsymbol{x} : \boldsymbol{T}$ だけでなく，型変数の宣言 $\boldsymbol{X} : \texttt{Type}$ も並ぶことになる．また (ふつうの) 関数や関数適用のための規則は変わらない．

多相関数についての型付け規則は

$$\frac{\boldsymbol{\Gamma},\ \boldsymbol{X}\ :\ \texttt{Type} \vdash \boldsymbol{M}\ :\ \boldsymbol{T} \qquad \boldsymbol{X} \notin \mathit{dom}(\boldsymbol{\Gamma})}{\boldsymbol{\Gamma} \vdash \texttt{fun}\ \ \boldsymbol{X}\ :\ \texttt{Type}\ \ \texttt{=>}\ \ \boldsymbol{M}\ :\ \forall \boldsymbol{X}\ :\ \texttt{Type}\,\texttt{,}\boldsymbol{T}} \tag{T-TFun}$$

と与えられる．関数全体に与えられる型の違いはあるが，T-Fun 規則とよく似ている．この $\boldsymbol{X}$ は後で多相関数が型引数に適用された時に具体的な型に置き換えられるものである．実際，型適用の型付け規則は

$$\frac{\boldsymbol{\Gamma} \vdash \boldsymbol{M}\ :\ \forall \boldsymbol{X}\ :\ \texttt{Type}\,\texttt{,}\boldsymbol{T}[\boldsymbol{X}]}{\boldsymbol{\Gamma} \vdash \boldsymbol{M}\ \boldsymbol{S}\ :\ \boldsymbol{T}[\boldsymbol{S}]} \tag{T-TApp}$$

のようになり，多相関数型 $\forall \boldsymbol{X}\ :\ \texttt{Type}\,\texttt{,}\boldsymbol{T}[\boldsymbol{X}]$ を持つ多相関数 $\boldsymbol{M}$ 型引数 $\boldsymbol{S}$ を与えると，$\boldsymbol{X}$ を $\boldsymbol{S}$ に置き換えたような型 (すなわち $\boldsymbol{T}[\boldsymbol{S}]$) が得られる．

例えば，上で定義した二回適用する関数の例は以下のように型付け関係を導出することができる．

$$\begin{aligned}
\boldsymbol{M} &\stackrel{\text{def}}{=} \texttt{fun X : Type => fun f : X -> X => fun x : X => f (f x)} \\
\boldsymbol{\Gamma} &\stackrel{\text{def}}{=} \texttt{X : Type, f : X -> X, x : X -> X}
\end{aligned}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\boldsymbol{\Gamma} \vdash \texttt{f : X -> X}}\ \textsc{T-Var} \qquad \dfrac{\dfrac{}{\boldsymbol{\Gamma} \vdash \texttt{f : X -> X}}\ \textsc{T-Var} \quad \dfrac{}{\boldsymbol{\Gamma} \vdash \texttt{x : X}}\ \textsc{T-Var}}{\boldsymbol{\Gamma} \vdash \texttt{f x : X}}\ \textsc{T-App}}{\boldsymbol{\Gamma} \vdash \texttt{f (f x) : X}}\ \textsc{T-App}}{\texttt{X : Type, f : X -> X} \vdash \texttt{fun x : X => f (f x) : X -> X}}\ \textsc{T-Fun}}{\texttt{X : Type} \vdash \texttt{fun f : X -> X => fun x : X => f (f x) : (X -> X) -> X -> X}}\ \textsc{T-Fun}$$

$$\dfrac{\dfrac{\cdots}{\vdash \boldsymbol{M}\ \texttt{: }\forall\texttt{X : Type, (X -> X) -> X -> X}}\ \textsc{T-TFun}}{\vdash \boldsymbol{M}\ \texttt{nat : (nat -> nat) -> nat -> nat}}\ \textsc{T-TApp}$$

以下に (自然数も真偽値も再帰関数も含まない) 純粋な System F の定義をまとめる．

1.1 型，ラムダ項，型環境，判断の構文

$$
\begin{array}{llcl}
\text{(types)} & \boldsymbol{S}, \boldsymbol{T} & ::= & \boldsymbol{X} \\
 & & | & \boldsymbol{S}\ \texttt{->}\ \boldsymbol{T} \\
 & & | & \forall \boldsymbol{X}\ \texttt{:}\ \texttt{Type}\,\texttt{,}\boldsymbol{T} \\
\\
\text{(terms)} & \boldsymbol{M}, \boldsymbol{N} & ::= & \boldsymbol{x} \\
 & & | & \texttt{fun}\ \ \boldsymbol{x}\ \texttt{:}\ \boldsymbol{T}\ \texttt{=>}\ \boldsymbol{M} \\
 & & | & \boldsymbol{M_1}\ \boldsymbol{M_2} \\
 & & | & \texttt{fun}\ \ \boldsymbol{X}\ \texttt{:}\ \texttt{Type}\ \texttt{=>}\ \boldsymbol{M} \\
 & & | & \boldsymbol{M}\ \boldsymbol{T} \\
\\
\text{(type environments)} & \boldsymbol{\Gamma} & ::= & \bullet \\
 & & | & \boldsymbol{\Gamma},\ \boldsymbol{x}\ \texttt{:}\ \boldsymbol{T} \\
 & & | & \boldsymbol{\Gamma},\ \boldsymbol{X}\ \texttt{:}\ \texttt{Type}
\end{array}
$$

1.2 簡約

$$(\texttt{fun}\ \ \boldsymbol{x}\ \texttt{:}\ \boldsymbol{T}\ \texttt{=>}\ \boldsymbol{M}[\boldsymbol{x}])\ \boldsymbol{N} \longrightarrow \boldsymbol{M}[\boldsymbol{N}] \tag{R-Beta}$$

$$(\texttt{fun}\ \ \boldsymbol{X}\ \texttt{:}\ \texttt{Type}\ \texttt{=>}\ \boldsymbol{M}[\boldsymbol{X}])\ \boldsymbol{T} \longrightarrow \boldsymbol{M}[\boldsymbol{T}] \tag{R-TBeta}$$

$$\frac{\boldsymbol{M} \longrightarrow \boldsymbol{M'}}{\texttt{fun}\ \ \boldsymbol{x}\ \texttt{:}\ \boldsymbol{T}\ \texttt{=>}\ \boldsymbol{M} \longrightarrow \texttt{fun}\ \ \boldsymbol{x}\ \texttt{:}\ \boldsymbol{T}\ \texttt{=>}\ \boldsymbol{M'}} \tag{RC-Fun}$$

$$\frac{\boldsymbol{M} \longrightarrow \boldsymbol{M'}}{\boldsymbol{M}\ \boldsymbol{N} \longrightarrow \boldsymbol{M'}\ \boldsymbol{N}} \tag{RC-App1}$$

$$\frac{\boldsymbol{N} \longrightarrow \boldsymbol{N'}}{\boldsymbol{M}\ \boldsymbol{N} \longrightarrow \boldsymbol{M}\ \boldsymbol{N'}} \tag{RC-App2}$$

$$\frac{\boldsymbol{M} \longrightarrow \boldsymbol{M'}}{\texttt{fun}\ \ \boldsymbol{X}\ \texttt{:}\ \texttt{Type}\ \texttt{=>}\ \boldsymbol{M} \longrightarrow \texttt{fun}\ \ \boldsymbol{X}\ \texttt{:}\ \texttt{Type}\ \texttt{=>}\ \boldsymbol{M'}} \tag{RC-TFun}$$

$$\frac{\boldsymbol{M} \longrightarrow \boldsymbol{M'}}{\boldsymbol{M}\ \boldsymbol{T} \longrightarrow \boldsymbol{M'}\ \boldsymbol{T}} \tag{RC-TApp}$$

1.3 型付け規則

$$\frac{(\boldsymbol{x}\ \texttt{:}\ \boldsymbol{T} \in \boldsymbol{\Gamma})}{\boldsymbol{\Gamma} \vdash \boldsymbol{x}\ \texttt{:}\ \boldsymbol{T}} \tag{T-Var}$$

$$\frac{\boldsymbol{\Gamma},\ \boldsymbol{x} : \boldsymbol{S} \vdash \boldsymbol{M} : \boldsymbol{T} \qquad (\boldsymbol{x} \notin dom(\boldsymbol{\Gamma}))}{\boldsymbol{\Gamma} \vdash \texttt{fun}\ \boldsymbol{x} : \boldsymbol{S}\ \texttt{=>}\ \boldsymbol{M} : \boldsymbol{S}\ \texttt{->}\ \boldsymbol{T}} \tag{T-Fun}$$

$$\frac{\boldsymbol{\Gamma} \vdash \boldsymbol{M} : \boldsymbol{S}\ \texttt{->}\ \boldsymbol{T} \qquad \boldsymbol{\Gamma} \vdash \boldsymbol{N} : \boldsymbol{S}}{\boldsymbol{\Gamma} \vdash \boldsymbol{M}\ \boldsymbol{N} : \boldsymbol{T}} \tag{T-App}$$

$$\frac{\boldsymbol{\Gamma},\ \boldsymbol{X} : \texttt{Type} \vdash \boldsymbol{M} : \boldsymbol{T} \qquad (\boldsymbol{X} \notin dom(\boldsymbol{\Gamma}))}{\boldsymbol{\Gamma} \vdash \texttt{fun}\ \boldsymbol{X} : \texttt{Type}\ \texttt{=>}\ \boldsymbol{M} : \forall \boldsymbol{X} : \texttt{Type}, \boldsymbol{T}} \tag{T-TFun}$$

$$\frac{\boldsymbol{\Gamma} \vdash \boldsymbol{M} : \forall \boldsymbol{X} : \texttt{Type}, \boldsymbol{S}[\boldsymbol{X}]}{\boldsymbol{\Gamma} \vdash \boldsymbol{M}\ \boldsymbol{T} : \boldsymbol{S}[\boldsymbol{T}]} \tag{T-TApp}$$

代入と変数キャプチャについて再び　単純型付ラムダ計算において，代入時には変数のキャプチャを避けなければならない，ということを延べたが，System F でも同じ問題が発生する．System F では，さらに，項への型の代入，型への型の代入があり，fun $\boldsymbol{X}$: Type => $\boldsymbol{M}$ や $\forall \boldsymbol{X}$: Type,$\boldsymbol{T}$ という，型パラメータを宣言する項・型があるため，**型変数のキャプチャ**を避ける必要がある．方法は単純型付ラムダ計算と同じで，**型変数の名前替え**を行うことになる．例えば，

$$\texttt{Y -> }(\forall\texttt{X : Type,X -> Y})$$

の Y に X -> nat を代入した場合，得られる結果は，

$$(\texttt{X -> nat})\texttt{ -> }(\forall\texttt{X : Type,X -> X -> nat})$$

ではなく，$\forall$X : Type, $\cdots$ の部分の X の名前替えをして

$$(\texttt{X -> nat})\texttt{ -> }(\forall\texttt{Z : Type,Z -> X -> nat})$$

となる．

引数の名前替えをした項同士 (例えば fun x : nat => x と fun y : nat => y) を同一視したのと同様，多相関数の型変数，さらには，多相関数型の型変数の名前替えも自由に行うのが通例である．例えば，

$$\texttt{fun X : Type => fun x : X => x} \quad \text{と} \quad \texttt{fun Y : Type => fun x : Y => x}$$

や

$$\forall\texttt{X : Type,X -> X} \quad \text{と} \quad \forall\texttt{Y : Type,Y -> Y}$$

は同一視する．

1.4 (純粋な) System F の重要な性質

定理 1 (合流性) 任意の項 $\boldsymbol{M_1}, \boldsymbol{M_2}, \boldsymbol{M_3}$ に対して，$\boldsymbol{M_1} \longrightarrow^* \boldsymbol{M_2}$ かつ $\boldsymbol{M_1} \longrightarrow^* \boldsymbol{M_3}$ ならば，ある $\boldsymbol{M_4}$ が存在し，$\boldsymbol{M_2} \longrightarrow^* \boldsymbol{M_4}$ かつ $\boldsymbol{M_3} \longrightarrow^* \boldsymbol{M_4}$ が成り立つ.

定理 2 (型保存定理) $\boldsymbol{\Gamma} \vdash \boldsymbol{M} : \boldsymbol{T}$ かつ，$\boldsymbol{M} \longrightarrow \boldsymbol{M'}$ ならば，$\boldsymbol{\Gamma} \vdash \boldsymbol{M'} : \boldsymbol{T}$ である.

定理 3 (強正規化性) $\boldsymbol{\Gamma} \vdash \boldsymbol{M} : \boldsymbol{T}$ とする．$\boldsymbol{M} \longrightarrow \boldsymbol{M_1} \longrightarrow \boldsymbol{M_2} \longrightarrow \cdots$ なる項の無限列 $\boldsymbol{M_1}, \boldsymbol{M_2}, \ldots$ は存在しない.

2 System F + 自然数，真偽値，リスト，再帰関数

$$
\begin{array}{llcl}
\text{(types)} & \boldsymbol{S}, \boldsymbol{T} & ::= & \cdots \\
 & & | & \texttt{nat} \\
 & & | & \texttt{bool} \\
 & & | & \texttt{list}\ \boldsymbol{T} \\
\\
\text{(terms)} & \boldsymbol{M}, \boldsymbol{N} & ::= & \cdots \\
 & & | & \texttt{O} \\
 & & | & \texttt{S} \\
 & & | & \texttt{match}\ \boldsymbol{M}\ \texttt{with O =>}\ \boldsymbol{N_1}\ \texttt{| S}\ \boldsymbol{x}\ \texttt{=>}\ \boldsymbol{N_2}\ \texttt{end} \\
 & & | & \texttt{true} \\
 & & | & \texttt{false} \\
 & & | & \texttt{if}\ \boldsymbol{M}\ \texttt{then}\ \boldsymbol{N_1}\ \texttt{else}\ \boldsymbol{N_2} \\
 & & | & \texttt{nil} \\
 & & | & \texttt{cons} \\
 & & | & \texttt{match}\ \boldsymbol{M}\ \texttt{with nil =>}\ \boldsymbol{N_1}\ \texttt{| cons}\ \boldsymbol{x}\ \boldsymbol{y}\ \texttt{=>}\ \boldsymbol{N_2} \\
 & & | & \texttt{fix}\ \boldsymbol{x}\texttt{(}\boldsymbol{y}\ \texttt{:}\ \boldsymbol{S}\texttt{)}\ \texttt{:}\ \boldsymbol{T}\ \texttt{:=}\ \boldsymbol{M} \\
 & \boldsymbol{\Gamma} & ::= & \cdots
\end{array}
$$

2.1 簡約

$$\texttt{match O with O =>}\ \boldsymbol{N_1}\ \texttt{| S}\ \boldsymbol{x}\ \texttt{=>}\ \boldsymbol{N_2}\ \texttt{end} \longrightarrow \boldsymbol{N_1} \tag{R-MatchZ}$$

$$\texttt{match S}\ \boldsymbol{M}\ \texttt{with O =>}\ \boldsymbol{N_1}\ \texttt{| S}\ \boldsymbol{x}\ \texttt{=>}\ \boldsymbol{N_2}[\boldsymbol{x}]\ \texttt{end} \longrightarrow \boldsymbol{N_2}[\boldsymbol{M}] \tag{R-MatchS}$$

$$\texttt{if true then}\ \boldsymbol{N_1}\ \texttt{else}\ \boldsymbol{N_2} \longrightarrow \boldsymbol{N_1} \tag{R-IfT}$$

$$\texttt{if false then}\ \boldsymbol{N_1}\ \texttt{else}\ \boldsymbol{N_2} \longrightarrow \boldsymbol{N_2} \tag{R-IfF}$$

$$\texttt{match nil}\ \boldsymbol{T}\ \texttt{with nil =>}\ \boldsymbol{N_1}\ \texttt{| cons}\ \boldsymbol{x}\ \boldsymbol{y}\ \texttt{=>}\ \boldsymbol{N_2}[\boldsymbol{x}, \boldsymbol{y}] \longrightarrow \boldsymbol{N_1} \tag{R-MatchN}$$

$$\texttt{match cons}\ \boldsymbol{T}\ \boldsymbol{M_1}\ \boldsymbol{M_2}\ \texttt{with nil =>}\ \boldsymbol{N_1}\ \texttt{| cons}\ \boldsymbol{x}\ \boldsymbol{y}\ \texttt{=>}\ \boldsymbol{N_2}[\boldsymbol{x}, \boldsymbol{y}] \longrightarrow \boldsymbol{N_2}[\boldsymbol{M_1}, \boldsymbol{M_2}] \tag{R-MatchC}$$

$$\texttt{(fix}\ \boldsymbol{x}\texttt{(}\boldsymbol{y}\ \texttt{:}\ \boldsymbol{S}\texttt{)}\ \texttt{:}\ \boldsymbol{T}\ \texttt{:=}\ \boldsymbol{M}[\boldsymbol{x}, \boldsymbol{y}]\texttt{)}\ \boldsymbol{N} \longrightarrow \boldsymbol{M}[\texttt{fix}\ \boldsymbol{x}\texttt{(}\boldsymbol{y}\ \texttt{:}\ \boldsymbol{S}\texttt{)}\ \texttt{:}\ \boldsymbol{T}\ \texttt{:=}\ \boldsymbol{M}[\boldsymbol{x}, \boldsymbol{y}], \boldsymbol{N}] \tag{R-Fix}$$

$$\frac{\boldsymbol{M} \longrightarrow \boldsymbol{M'}}{\begin{array}{l}\texttt{match}\ \boldsymbol{M}\ \texttt{with O =>}\ \boldsymbol{N_1}\ \texttt{| S}\ \boldsymbol{x}\ \texttt{=>}\ \boldsymbol{N_2}\ \texttt{end} \\ \longrightarrow \texttt{match}\ \boldsymbol{M'}\ \texttt{with O =>}\ \boldsymbol{N_1}\ \texttt{| S}\ \boldsymbol{x}\ \texttt{=>}\ \boldsymbol{N_2}\ \texttt{end}\end{array}} \tag{RC-Match1}$$

$$\frac{\boldsymbol{N_1} \longrightarrow \boldsymbol{N_1'}}{\begin{array}{l}\texttt{match}\ \boldsymbol{M}\ \texttt{with O =>}\ \boldsymbol{N_1}\ \texttt{| S}\ \boldsymbol{x}\ \texttt{=>}\ \boldsymbol{N_2}\ \texttt{end} \\ \longrightarrow \texttt{match}\ \boldsymbol{M}\ \texttt{with O =>}\ \boldsymbol{N_1'}\ \texttt{| S}\ \boldsymbol{x}\ \texttt{=>}\ \boldsymbol{N_2}\ \texttt{end}\end{array}} \tag{RC-Match2}$$

$$\frac{N_2 \longrightarrow N_2'}{\begin{array}{c}\texttt{match } M \texttt{ with 0 => } N_1 \texttt{ | S } x \texttt{ => } N_2 \texttt{ end} \\ \longrightarrow \texttt{match } M \texttt{ with 0 => } N_1 \texttt{ | S } x \texttt{ => } N_2' \texttt{ end}\end{array}} \quad \text{(RC-Match3)}$$

$$\frac{M \longrightarrow M'}{\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2 \longrightarrow \texttt{if } M' \texttt{ then } N_1 \texttt{ else } N_2} \quad \text{(RC-If1)}$$

$$\frac{N_1 \longrightarrow N_1'}{\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2 \longrightarrow \texttt{if } M \texttt{ then } N_1' \texttt{ else } N_2} \quad \text{(RC-If2)}$$

$$\frac{N_2 \longrightarrow N_2'}{\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2 \longrightarrow \texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2'} \quad \text{(RC-If3)}$$

$$\frac{M \longrightarrow M'}{\texttt{fun } x : T \texttt{ => } M \longrightarrow \texttt{fun } x : T \texttt{ => } M'} \quad \text{(RC-Fun)}$$

$$\frac{M \longrightarrow M'}{\texttt{fix } x(y : S) : T \texttt{ := } M \longrightarrow \texttt{fix } x(y : S) : T \texttt{ := } M'} \quad \text{(RC-Fix)}$$

2.2 型付け規則

$$\frac{}{\Gamma \vdash \texttt{0} : \texttt{nat}} \quad \text{(T-Zero)}$$

$$\frac{}{\Gamma \vdash \texttt{S} : \texttt{nat -> nat}} \quad \text{(T-Succ)}$$

$$\frac{\Gamma \vdash M : \texttt{nat} \qquad \Gamma \vdash N_1 : T \qquad \Gamma,\ x : \texttt{nat} \vdash N_2 : T}{\Gamma \vdash \texttt{match } M \texttt{ with 0 => } N_1 \texttt{ | S } x \texttt{ => } N_2 \texttt{ end} : T} \quad \text{(T-Match)}$$

$$\frac{}{\Gamma \vdash \texttt{true} : \texttt{bool}} \quad \text{(T-True)}$$

$$\frac{}{\Gamma \vdash \texttt{false} : \texttt{bool}} \quad \text{(T-False)}$$

$$\frac{\Gamma \vdash M : \texttt{bool} \qquad \Gamma \vdash N_1 : T \qquad \Gamma \vdash N_2 : T}{\Gamma \vdash \texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2 : T} \quad \text{(T-If)}$$

$$\frac{\Gamma,\ x : S \texttt{->} T,\ y : S \vdash M : T}{\Gamma \vdash \texttt{fix } x(y : S) : T \texttt{ := } M : S \texttt{->} T} \quad \text{(T-Fix)}$$

$$\frac{}{\Gamma \vdash \texttt{nil} : \forall X : \texttt{Type,list } X} \quad \text{(T-Nil)}$$

$$\frac{}{\mathbf{\Gamma} \vdash \texttt{cons} : \forall \boldsymbol{X} : \texttt{Type}, \boldsymbol{X} \texttt{ -> list } \boldsymbol{X} \texttt{ -> list } \boldsymbol{X}} \quad (\text{T-Cons})$$

$$\frac{\mathbf{\Gamma} \vdash \boldsymbol{M} : \texttt{list } \boldsymbol{S} \qquad \mathbf{\Gamma} \vdash \boldsymbol{N_1} : \boldsymbol{T} \qquad \mathbf{\Gamma}, \boldsymbol{x} : \boldsymbol{S}, \boldsymbol{y} : \texttt{list } \boldsymbol{S} \vdash \boldsymbol{N_2} : \boldsymbol{T}}{\mathbf{\Gamma} \vdash \texttt{match } \boldsymbol{M} \texttt{ with nil => } \boldsymbol{N_1} \texttt{ | cons } \boldsymbol{x}\ \boldsymbol{y} \texttt{ => } \boldsymbol{N_2} : \boldsymbol{T}} \quad (\text{T-MatchL})$$

項 $\boldsymbol{M}$ が**標準形** (*canonical form*) である，とは，$\boldsymbol{M}$ が `O`, `S` $\boldsymbol{N}$，`true`，`false`，`fun` $\boldsymbol{x} : \boldsymbol{T}$ `=>` $\boldsymbol{N}$，`fix` $\boldsymbol{x}(\boldsymbol{y} : \boldsymbol{S}) : \boldsymbol{T}$ `:=` $\boldsymbol{N}$, もしくは `fun` $\boldsymbol{X}$: `Type` `=>` $\boldsymbol{M}$ のいずれかの形をしていることをいう．

定理 4 (前進性) $\vdash \boldsymbol{M} : \boldsymbol{T}$ とする．任意の $\boldsymbol{M}$ の部分項 $\boldsymbol{M_0}$ について，以下が成立する．

- $\boldsymbol{M_0}$ が $\boldsymbol{M_1}\ \boldsymbol{M_2}$ の形 (ただし $\boldsymbol{M_1}$ は標準形) ならば，$\boldsymbol{M_1}$ は `fun` $\boldsymbol{x} : \boldsymbol{T'}$ `=>` $\boldsymbol{M'}$ もしくは `fix` $\boldsymbol{x}(\boldsymbol{y} : \boldsymbol{S'}) : \boldsymbol{T'}$ `:=` $\boldsymbol{M'}$ の形である．

- $\boldsymbol{M_0}$ が `match` $\boldsymbol{M_1}$ `with O =>` $\boldsymbol{M_2}$ `| S` $\boldsymbol{x}$ `=>` $\boldsymbol{M_3}$ `end` の形 (ただし $\boldsymbol{M_1}$ は標準形) ならば，$\boldsymbol{M_1}$ は `O` または，`S` $\boldsymbol{M'}$ の形である．

- $\boldsymbol{M_0}$ が `match` $\boldsymbol{M_1}$ `with nil =>` $\boldsymbol{M_2}$ `| cons` $\boldsymbol{x}\ \boldsymbol{y}$ `=>` $\boldsymbol{M_3}$ `end` の形 (ただし $\boldsymbol{M_1}$ は標準形) ならば，$\boldsymbol{M_1}$ は `nil` または，`cons` $\boldsymbol{M'}\ \boldsymbol{M''}$ の形である．

- $\boldsymbol{M_0}$ が `if` $\boldsymbol{M_1}$ `then` $\boldsymbol{M_2}$ `else` $\boldsymbol{M_3}$ の形 (ただし $\boldsymbol{M_1}$ は標準形) ならば，$\boldsymbol{M_1}$ は `true` もしくは `false` である．

- $\boldsymbol{M_0}$ が $\boldsymbol{M_1}\ \boldsymbol{S}$ の形 (ただし $\boldsymbol{M_1}$ は標準形) ならば，$\boldsymbol{M_1}$ は `fun` $\boldsymbol{X}$: `Type` `=>` $\boldsymbol{M'}$ の形である．