Types and Programming Languages
Chapter 11-14

Rei Tomori

July 26, 2025

概要

今回は以下の内容を扱う:

1. §11. 単純な拡張 (§11.4-)
2. §12. 正規化
3. §13. 参照
4. §14. 例外

§11. 単純な拡張

基本型

1. プログラミング言語には基本型[1].i.e. 構造を持たない単純な値の集合とその上のプリミティブな演算がある.
   - 今後は $\mathtt{Nat}, \mathtt{Bool}$ に加え，$\mathtt{String}, \mathtt{Float}$ 型を用いる.
2. (その上の演算を捨象した) 一般的な基本型を扱いたいことがある．そのためには，言語が”非解釈の”基本型の集合 $\mathscr{A}$[2]を備えているとする.
   - これを表わすためには，型の構文規則を変更してメタ変数 A($\mathscr{A}$ の要素を表わす) を加える.
   - 以降，基本型の名前として $\mathsf{A}, \mathsf{B}, \mathsf{C}$ を用いる.
3. 非解釈な型を導入することで，基本型の要素上を特定することなく，その上で走る変数を束縛できる.
   - $\lambda \mathsf{x} : \mathsf{A}.\mathsf{x}$ は $\mathsf{x} : \mathsf{A}$ が何であれ，x を x 自身に写す恒等関数である.

[1]基底型

[2]原子型，つまり型システムにおいては内部システムを持たない型の略

§11.2. Unit 型

要素を 1 つしか持たない型である Unit 型を導入する．この型は次のように解釈される:

- 唯一の要素は項定数 unit(u，しばしば () で表わされる) で，任意の Unit 型の項は一意的に unit に評価される．

Unit 型は主に副作用を持つ言語で応用される[3]．たとえば，可変参照を変更する関数では，返り値ではなく副作用に興味があるため，Unit 型が返り値の型とされる．類似物として C 系言語の void 型がある．

[3]純粋函数型言語では，たとえば各 $n \in \mathbb{N}$ に対して

Unit 型

Unit 型は[4]，以下の構文，型付け規則および派生形式 (i.e. 糖衣構文)，すなわち STLC への埋め込み方で定義される．

Def.Unit 型の定義

新しい構文形式

(項) $\mathtt{t} ::= \ldots$

$\quad \mathtt{Unit}$

(項) $\mathtt{v} ::= \ldots$

$\quad \mathtt{unit}$

(型) $\mathtt{T} ::= \ldots$

$\quad \mathtt{Unit}$

新しい型付け規則

$$\Gamma \vdash \mathtt{unit} : \mathtt{Unit}$$

新しい派生形式

$$\mathtt{t}_1; \mathtt{t}_2 \stackrel{\text{def}}{=} (\lambda \mathtt{x} : \mathtt{Unit}.\mathtt{t}_2)\mathtt{t}_1$$

ただし$x \notin FV(\mathtt{t}_2)$

[4]Haskell でいうところの 0-tuple

§11.3. 派生形式: 逐次実行とワイルドカード

副作用のある言語における文の逐次実行を形式化する．逐次実行形式は項 $\mathtt{t}_1, \mathtt{t}_2$ に対し，$\mathtt{t}_1$ を正規形まで評価し，結果を捨てた後に $\mathtt{t}_2$ を評価する．

$(;)$ の意味を直接表わす方法と，$(;)$ を内部言語のある項の略記とする 2 通りの形式化が考えられる．

1. $\mathtt{t}_1; \mathtt{t}_2$ を新たな構文要素とする方法．評価規則

$$\frac{\mathtt{t}_1 \to \mathtt{t}_1'}{\mathtt{t}_1; \mathtt{t}_2 \to \mathtt{t}_1'; \mathtt{t}_2}\ (\text{E-SEQ})$$

$$\mathtt{v}_1; \mathtt{t}_2 \to \mathtt{t}_2\ (\text{E-SEQNEXT})$$

   および型付け規則

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{Unit} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \mathtt{t}_1; \mathtt{t}_2 : \mathtt{T}_2}\ (\text{T-SEQ})$$

   を付け加えることで $(;)$ の振る舞いを特徴付ける．

2. 内部言語の項の略記とする方法．$\mathtt{t}_1; \mathtt{t}_2 \stackrel{\mathtt{def}}{=} (\lambda \mathtt{x} : \mathtt{Unit}.\mathtt{t}_2)\mathtt{t}_1$，ただし $x\mathtt{x} \notin FV(\mathtt{t}_2)$ とする．

派生形式

はじめの形式化で定めた評価および型付け規則は，Unit のみを型として持つ STLC の評価関係および型付け規則より従う．このことを確認しよう．

Thm.11.3.1[逐次実行は派生形式である]
Unit 型，逐次実行およびそれらの評価・型付け規則を持つ STLC を λ^E と書く．[5]また，Unit 型のみを持つ STLC を λ^I と書き，$e \in \lambda^I \to \lambda^E$ を，λ^I の各項を対応する λ^E の項に写す詳細化関数[6]とする．つまり，e は $t_1; t_2$ の各出現を，$(\lambda x : \mathtt{Unit}.t_2)t_1 (x \notin FV(t_2))$ に置き換える．すると，λ^E の各項 t に対して，

1. $t \to_E t' \Rightarrow e(t) \to_I e(t')$．逆に $e(t) \to_I u \Rightarrow \exists t'.t'$ は λ^E の項 $\wedge u = e(t') \wedge t \to_E t'$.
2. $\Gamma \vdash^E t : T \Leftrightarrow \Gamma \vdash^I e(t) : T$

[5]外部言語の略.
[6]elaboration function

Thm.11.3.1. の証明 (一部)

Proof.
$\mathtt{t}$ の構造帰納法で示す．(2) の主張は白板で示すことにする．

1. (1) の証明．
   1. ($\Rightarrow$) 新たな構文要素，つまり $\mathtt{t} = \mathtt{t}_1; \mathtt{t}_2$ または $\mathtt{t} = \mathtt{v}_1; \mathtt{t}_2$ の場合を考え，$\mathtt{t} \to_{\mathsf{E}} \mathtt{t}'$ なる $\mathtt{t}'$ の存在を仮定．
      - $\mathtt{t} = \mathtt{v}_1; \mathtt{t}_2$．仮定より $\mathtt{t}' = \mathtt{t}_2$．いま $\mathtt{x} \notin \mathit{FV}(\mathtt{t}_2)$ を任意に取ると，定義より $e(\mathtt{t}) = (\lambda \mathtt{x} : \mathtt{Unit}.e(\mathtt{t}_2))\ \mathtt{v}_1$．(E-APPABS) から $\mathtt{t} \to_I [\mathtt{x} \mapsto \mathtt{v}_1]\, e(\mathtt{t}_2) = \mathtt{t}'$ がいえ，$\mathtt{x} \notin \mathit{FV}(e(\mathtt{t}_2))$ からこれは $\mathtt{v}_1$ に等しい．
      - $\mathtt{t} = \mathtt{t}_1; \mathtt{t}_2$．仮定より $\mathtt{t}' = \mathtt{t}_1'; \mathtt{t}_2 \wedge \mathtt{t}_1 \to_{\mathsf{E}} \mathtt{t}_1'$ なる $\mathtt{t}', \mathtt{t}_1'$ が存在．IH より $e(\mathtt{t}_1) \to_I e(\mathtt{t}_1')$．(E-APP1) から $e(\mathtt{t}) = e(\mathtt{t}_1)e(\mathtt{t}_2) \to_I e(\mathtt{t}_1')e(\mathtt{t}_2 = e(\mathtt{t}')$.
   2. ($\Leftarrow$)．$e(\mathtt{t}) \to_I \mathtt{u}$ を仮定する．(1) と同様に場合を分ける．
      - $\mathtt{t} = \mathtt{v}_1; \mathtt{t}_2$ のとき．$e(\mathtt{t}) = (\lambda \mathtt{x} : \mathtt{Unit}.e(\mathtt{t}_2))\mathtt{v}_1$($\mathtt{x}$ は fresh な変数)，$\mathtt{u} = e(\mathtt{t}_2)$. このとき．$\mathtt{t}' = \mathtt{t}_2$ とすれば従う．
      - $\mathtt{t} = \mathtt{t}_1; \mathtt{t}_2$ のとき．$\mathtt{t} = (\lambda \mathtt{x} : \mathtt{Unit}.e(\mathtt{t}_2))\ e(\mathtt{t}_1)$ であり，仮定より $e(\mathtt{t}_1) \to \mathtt{u}_1'$ なる $\mathtt{u}_1'$ を取ると $\mathtt{u} = (\lambda \mathtt{x} : \mathtt{Unit}.e(\mathtt{t}_1))\ \mathtt{u}_1'$．帰納法の仮定より，$\mathtt{u}_1' = e(\mathtt{t}_1') \wedge \mathtt{t}_1 \to_E \mathtt{t}_1'$ なる λ^E の項 $\mathtt{t}_1'$ が取れる．ゆえに $\mathtt{u} = (\lambda \mathtt{x} : \mathtt{Unit}.e(\mathtt{t}_2))e(\mathtt{t}_1') = e(\mathtt{t}_1'; \mathtt{t}_2)$ となり従う．

□

項の型を明示的に指定する機能を導入しよう.

Def. 型指定
型 T を指定した項 t を，t as T と書く．評価規則を次のように定める:

$$v_1 \text{ as } T \longrightarrow v_1 \qquad \text{(E-Ascribe)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1 \text{ as } T \longrightarrow t_1' \text{ as } T} \qquad \text{(E-Ascribe1)}$$

型付け規則を次のように定める:

$$\frac{\Gamma \vdash t_1 : T}{\Gamma \vdash t_1 \text{ as } T : T} \qquad \text{(T-Ascribe1)}$$

演習 11.4.1.

演習 11.4.1.[推奨]

1. 型指定を派生形式として形式化せよ．「公式」の型付け規則および評価規則が，その形式化に何らかの適切な意味で対応することを証明せよ．
2. Def. で定めた評価規則の代わりに，以下の先行的な規則が与えられたとする:

$$t_1 \text{ as } T \longrightarrow t_1 \qquad \text{(E-AscribeEager)}$$

．この場合にも，型指定を派生形式として扱うことはできるか．

Ex11.4.1. の解答

1. t as T の派生形式は $(\lambda \mathsf{x} : \mathsf{T}.\mathsf{x})\ \mathsf{t}$ とすればよい．また，拡張された構文の項を拡張前の項に写す関数 e を次のように定める:

$$e(t) = \begin{cases} (\lambda \mathsf{x} : \mathsf{T}.\mathsf{x})\ \mathsf{t}_1 & \mathsf{t} \text{ が } \mathsf{t}_1 : \mathsf{T} \text{ の形のとき} \\ \mathsf{t} & \text{otherwise} \end{cases}$$

．e は明らかに全単射である．「公式」の型付けおよび評価規則が派生形式と対応することを示すためには，次の補題を示せばよい．

Cor.

1. $\mathsf{t} \longrightarrow \mathsf{t}'$ のとき $\mathsf{e}(\mathsf{t}) \longrightarrow \mathsf{e}(\mathsf{t}')$．また，$\mathsf{e}(\mathsf{t}) \longrightarrow \mathsf{u}$ ならば型指定された項 t' が存在して $\mathsf{u} = \mathsf{e}(\mathsf{t}') \wedge \mathsf{t} \longrightarrow \mathsf{t}'$
2. (型付け)$\Gamma \vdash \mathsf{t} : \mathsf{T}$，かつそのときに限り $\Gamma \vdash \mathsf{e}(\mathsf{t}) : \mathsf{T}$

Ex11.4.1: 補題の証明

1. 前半部分は，t を $t = t_1 : T$ の形に限定し，t_1 が値か否かで分ければ示せる．後半部分についても t が型指定された項 t_1 as T の場合のみに限定し，$e(t) = (\lambda x : T.x)t_1$，u は t_1 が値のとき t_1，そうでないとき $(\lambda x : T.x)t_1'$(ただし $t_1 \longrightarrow t_1'$) に評価されることに注意すれば，評価された項に e の逆関数を適用して存在が示せる．
2. $\Gamma \vdash t$ as T の型導出は次のようになるので，$\Gamma \vdash t : T$ が成立:

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash t : T}}{\Gamma \vdash t \text{ as } T : T}$$

$\Gamma \vdash e(t \text{ as } T) : T$ に至る導出木は，$\Gamma \vdash t : T$ に至る部分木から下の導出木を作ることで示せる:

$$\dfrac{\dfrac{\Gamma, x : T \vdash x : T}{\Gamma \vdash \lambda x : T.x : T \to T} \quad \dfrac{\vdots}{\Gamma \vdash t : T}}{\Gamma \vdash e(t \text{ as } T) = (\lambda x : T.x)\ t : T}$$

逆向きの主張は，上の導出木の部分木で $\Gamma \vdash t : T$ が成立していることから，$T - \mathrm{ASCRIBE}$ を用いて示せる．

Ex11.4.1(2)

派生形式として表わすことができる．型指定された項の評価を進めることなく消去するので，次のように定義すればよい:

$$\mathtt{t\ as\ T} \stackrel{\mathtt{def}}{=} (\lambda \mathtt{x} : \mathtt{Unit} \to \mathtt{T.x\ Unit})(\lambda \mathtt{y} : \mathtt{Unit}.\mathtt{t}_1) \tag{1}$$

部分式に対して名付けるための機能である let 束縛子を導入する.

Def.let 束縛
式 t_1 を評価した値を変数 x に束縛した状態で項 t_2 を評価する．これを，式 $\texttt{let}\ x = t_1\ \texttt{in}\ t_2$ で表わす．評価規則を次のように定める:

$$\texttt{let}\ x = v_1\ \texttt{in}\ t_2 \longrightarrow [x \mapsto v_1]\, t_2 \qquad \text{(E-LetV)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{let}\ x = t_1\ \texttt{in}\ t_2 \longrightarrow \texttt{let}\ x = t_1'\ \texttt{in}\ t_2} \qquad \text{(E-Let)}$$

つまり，この定義における let 束縛子は値呼びである．型付け規則を次のように定める:

$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \texttt{let}\ x = t_1\ \texttt{in}\ t_2} \qquad \text{(T-Let)}$$

つまり，束縛項の型を計算し，その型を持つ束縛で水増しされた文脈で本体の型を計算することで，let 式全体の型を計算している.

let 束縛に関する補足

1. let 式は派生形式 $\texttt{let}\ x = t_1\ \texttt{in}\ t_2 \stackrel{\text{def}}{=} (\lambda x : T_1.t_2)t_1$ として書けるが，束縛項の型は型検査器からしか分からない．
   - 型指定などのように項を脱糖するのではない (構文的には束縛項の型が分からない!)
   - 派生形式は型付け導出

$$\frac{\vdots \qquad\qquad \vdots}{\Gamma \vdash t_1 : T_1 \qquad \Gamma, x : T_1 \vdash t_2 : T_2}$$
$$\Gamma \vdash \texttt{let}\ x = t_1\ \texttt{in}\ t_2$$

   を

$$\frac{\dfrac{\vdots}{\Gamma, x : T_1 \vdash t_2 : T_2}}{\Gamma \vdash \lambda x : T_1.t_2 : T_1 \to T_2} \qquad \frac{\vdots}{\Gamma \vdash t_1 : T_1}$$
$$\Gamma \vdash (\lambda x : T_1.t_2)t_1 : T_2$$

   に写す変換とみなす必要がある．
   - また，今後 let 多相を導入するさいに問題が生じる (後述).

型 $T_1, \ldots, T_n$ に対して直積型 $\{\mathtt{T_i}^{i\in 1\ldots n}\}$ と，直積型の項 $t : \{\mathtt{T_i}^{1\in 1\ldots n}\}$ と各 $j = 1, \ldots, n$ に対する射影演算子 $t.j : \{\mathtt{T_i}^{i\in 1\ldots n}\} \to T_j$ を導入する．

Def.n-tuple
n 項の直積の導入に伴う項，値および型の構文要素を次のように定める:

1. (項)$\mathtt{t} ::= \ldots | \{\mathtt{t_i}^{\mathtt{i}\in 1\ldots \mathtt{n}}\} | \mathtt{t.i}$
2. (値)$\mathtt{v} ::= \ldots | \{\mathtt{v_i}^{\mathtt{i}\in 1\ldots \mathtt{n}}\}$
3. (型)$\mathtt{T} ::= \ldots | \{\mathtt{T_i}^{\mathtt{i}\in 1\ldots \mathtt{n}}\}$

評価規則を次のように定める:

$$\{\mathtt{v_i}^{\mathtt{i}\in 1\ldots \mathtt{n}}\}.\mathtt{j} \longrightarrow \mathtt{v_j}\ (\textsc{E-ProjTuple})$$

$$\frac{\mathtt{t_1} \longrightarrow \mathtt{t_1'}}{\mathtt{t_1.i} \longrightarrow \mathtt{t_1'.i}}\ (\textsc{E-Proj})$$

$$\frac{\mathtt{t_j} \longrightarrow \mathtt{t_j'}}{\{\mathtt{v_i}^{\mathtt{i}\in 1\ldots \mathtt{j}-1}, t_j, \mathtt{t_k}^{\mathtt{k}\in \mathtt{j}+1\ldots \mathtt{n}}\} \longrightarrow \{\mathtt{v_i}^{\mathtt{i}\in 1\ldots \mathtt{j}-1}, t_j', \mathtt{t_k}^{\mathtt{k}\in \mathtt{j}+1\ldots \mathtt{n}}\}}\ (\textsc{E-Tuple})$$

Def.n-tuple(つづき)
型付け規則は以下の 2 つ:

$$\frac{\forall i. \Gamma \vdash t_i : T_i}{\Gamma \vdash \{t_i^{i \in 1 \ldots n}\} : \{T_i^{i \in 1 \ldots n}\}} \quad \text{(T-Tuple)}$$

$$\frac{\Gamma \vdash t_1 : \{T_i^{i \in 1 \ldots n}\}}{\Gamma \vdash t_1.j : T_j} \quad \text{(T-Proj)}$$

規則 E-ProjTuple は，次のように書き直せる:

$$\frac{1 \leq j \leq n}{\{v_i^{i \in 1 \ldots j-1}, v_j, v_k^{k \in j+1 \ldots n}\}.j \longrightarrow v_j} \quad \text{(E-ProjTuple)}$$

型 T_1, T_2 に対する直積型 $T_1 \times T_2$ と，直積型の項 $t = \{t_1, t_2\} : T_1 \times T_2$ に対する射影演算子 $t.1 : T_1 \times T_2 \to T_1, t.2 : T_1 \times T_2 \to T_2$ を導入する．

Def.2-tuple
2 つ組の導入に伴う項，値および型の構文要素を次のように定める:

1. $t ::= \ldots \mid \{t, t\} \mid t.1 \mid t.2$
2. $v ::= \ldots \mid \{v, v\}$
3. $T ::= \ldots \mid T \times T$

評価規則を次のように定める．組は第一成分から評価を進める．

$$\{v_1, v_2\}.1 \longrightarrow v_1 \qquad \text{(E-PairBeta1)}$$

$$\{v_1, v_2\}.2 \longrightarrow v_2 \qquad \text{(E-PairBeta2)}$$

Def.2-tuple(続き)

$$\frac{t_1 \longrightarrow t_1'}{t_1.1 \longrightarrow t_1'.1} \text{ (E-Proj1)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1.2 \longrightarrow t_1'.2} \text{ (E-Proj2)}$$

$$\frac{t_1 \longrightarrow t_1'}{\{t_1, t_2\} \longrightarrow \{t_1', t_2\}} \text{ (E-Pair1)}$$

$$\frac{t_2 \longrightarrow t_2'}{\{v_1, t_2\} \longrightarrow \{v_1, t_2'\}} \text{ (E-Pair2)}$$

型付け規則

$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : T_2}{\Gamma \vdash \{t_1, t_2\} : T_1 \times T_2} \text{ (T-Pair)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \times T_{12}}{\Gamma \vdash t_1.1 : T_{11}} \text{ (T-Proj1)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \times T_{12}}{\Gamma \vdash t_1.2 : T_{12}} \text{ (T-Proj2)}$$

評価規則により，二つ組は第一成分が値に評価されたのちに第二成分が評価される．たとえば，$\mathtt{t} = \{\mathtt{pred}\ 4, \mathtt{if}\ \mathtt{true}\ \mathtt{then}\ \mathtt{false}\ \mathtt{then}\ \mathtt{false}\}.1$ の簡約列は次のように定まる:

$$\begin{aligned} \mathtt{t} &\longrightarrow \{3, \mathtt{if}\ \mathtt{true}\ \mathtt{then}\ \mathtt{true}\ \mathtt{else}\ \mathtt{false}\}.1 \\ &\longrightarrow \{3, \mathtt{true}\}.1 \longrightarrow 3 \end{aligned}$$

値の構文を追加することで，関数の引数として渡された 2 つ組は本体の評価前に完全に評価される．

n-タプルに対し，各フィールド t_i に集合 $\mathscr{L}$ から選んだラベル l_j を注記することで，ラベル付きレコードに一般化する．次のように定義する:

Def. レコード
新しい構文形式:

1. $t ::= \ldots \{l_i = t_i^{i \in 1 \ldots n}\} | t.l$
2. $v ::= \ldots | \{l_i = v_i^{i \in 1 \ldots n}\}$
3. $T ::= \ldots | \{t_i : T_i^{i \in 1 \ldots n}\}$

Def. レコード (つづき)

評価規則は以下の通り.

$$\{l_i = v_i^{i\in 1\ldots n}\}.l_j \longrightarrow v_j \ (\text{E-ProjRcd}) \qquad \text{型付け規則は次の通り.}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1.l \longrightarrow t_1'.l} \ (\text{E-Proj}) \qquad \frac{\forall i, \Gamma \vdash t_i : T_i}{\Gamma \vdash \{l_i = t_i^{i\in 1\ldots n}\}\{T_i^{i\in 1\ldots n}\}} \ (\text{T-Rcd})$$

$$\frac{t_j \longrightarrow t_j'}{\{l_i = v_i^{i\in 1\ldots j-1}, l_j = t_j, l_k = t_k^{k\in j+1\ldots n}\} \longrightarrow \{l_i = v_i^{i\in 1\ldots j-1}, l_j = t_j', l_k = t_k^{k\in j+1\ldots n}\}} \ (\text{E-Rcd})$$

$$\frac{\Gamma \vdash t_1 : \{t_i : T_i^{i\in 1\ldots n}\}}{\Gamma \vdash t_1.l_j : T_j} \ (\text{T-Proj})$$

E-ProjRcd は次のように書き直せる:

$$\frac{1 \le j \le n \qquad l_j \in \mathscr{L}}{\{l_i = v_i^{i\in 1\ldots n}\}.l_j \longrightarrow v_j} \qquad (\text{E-ProjRcd})$$

§11.8. レコードパターン

型 T_1, T_2 に対する (二項の) 和型 $T_1 + T_2$ と，項 $t_1 : T_1 (t_2 : T_2)$ から $T_1 + T_2$ の項を構成するタグ $\texttt{inl}(\texttt{inr})$，そして和型上のパターンマッチのための構文を導入する．[7]

Def. 和型
新しい構文形式

1. (項)$t ::= \ldots | \texttt{inl}\ t | \texttt{inr}\ t | \texttt{case}\ t\ \texttt{of}\ \texttt{inl}\ x \Rightarrow t | \texttt{inr}\ x \Rightarrow t$
2. (値)$v ::= \ldots | \texttt{inl}\ v | \texttt{inr}\ v$
3. (型)$T ::= \ldots | T + T$

評価規則を次のように定める:

$$\begin{array}{c} \texttt{case}(\texttt{inl}\ v_0)\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 \\ \quad | \texttt{inr}\ x_2 \Rightarrow t_2 \\ \longrightarrow [x_1 \mapsto v_0] t_1 \end{array} \quad \text{(E-CaseInl)}$$

$$\begin{array}{c} \texttt{case}(\texttt{inr}\ v_0)\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 \\ \quad | \texttt{inr}\ x_2 \Rightarrow t_2 \\ \longrightarrow [x_2 \mapsto v_0] t_2 \end{array} \quad \text{(E-CaseInr)}$$

Def. 和型

$$\frac{t_0 \longrightarrow t_0'}{\texttt{case}\ t_0\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2 \longrightarrow \texttt{case}\ t_0'\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2}\ (\text{E-Case})$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inl}\ t_1 \longrightarrow \texttt{inl}\ t_1'}\ (\text{E-Inl})$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inr}\ t_1 \longrightarrow \texttt{inr}\ t_1'}\ (\text{E-Inr})$$

$$\frac{\Gamma \vdash t_1 : T_1}{\Gamma \vdash \texttt{inl}\ t_1 : T_1 + T_2}\ (\text{T-Inl})$$

$$\frac{\Gamma \vdash t_1 : T_2}{\Gamma \vdash \texttt{inr}\ t_1 : T_1 + T_2}\ (\text{T-Inr})$$

$$\frac{\Gamma \vdash t_0 : T_1 + T_2 \qquad \Gamma, x_1 : T_1 \vdash t_1 : T \qquad \Gamma, x_2 : T_2 \vdash t_2 : T}{\Gamma \vdash \texttt{case}\ t_0\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2 : T}\ (\text{T-Case})$$

§11.9. 和と型の一意性

型付け規則 T-Inl(T-Inr) によれば，型導出の仮定に含まれない型には任意性がある．したがって，型の一意性 (Thm.9.3.3) は成立しない．一意性を取り戻す方法としては，3 つ考えられる:

1. 型検査アルゴリズムに変更を加え，不定となっている型を推論させる (§22 で扱う．)
2. 型の言語を改良し，可能な全ての T_2 を統一的に表現できるようにする．部分型を用いる．
3. 型注釈を付ける．今回はこちらを用いる．

Def. 和 (一意型付け)

新しい構文形式

1. 項:$t ::= \ldots \mid \texttt{inl}\ t\ \texttt{as}\ T \mid \texttt{inr}\ t\ \texttt{as}\ T$
2. 値:$v ::= \ldots \mid \texttt{inl}\ v\ \texttt{as}\ T \mid \texttt{inr}\ v\ \texttt{as}\ T$

新しい評価規則

$$\texttt{case}\ (\texttt{inl}\ v_0\ \texttt{as}\ T_0)\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 \mid \texttt{inr}\ x_2 \Rightarrow t_2 \longrightarrow [x_1 \mapsto v_0]t_1 \quad \text{(E-CaseInl)}$$

$$\texttt{case}\ (\texttt{inr}\ v_0\ \texttt{as}\ T_0)\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 \mid \texttt{inr}\ x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_0]t_2 \quad \text{(E-CaseInr)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inl}\ t_1\ \texttt{as}\ T_2 \longrightarrow \texttt{inl}\ t_1'\ \texttt{as}\ T_2} \quad \text{(E-Inl)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inr}\ t_1\ \texttt{as}\ T_2 \longrightarrow \texttt{inr}\ t_2'\ \texttt{as}\ T_2} \quad \text{(E-Inr)}$$

型付け規則

$$\frac{\Gamma \vdash t_1 : T_1}{\Gamma \vdash \texttt{inl}\ t_1\ \texttt{as}\ T_1 + T_2 : T_1 + T_2} \quad \text{(T-Inl)}$$

$$\frac{\Gamma \vdash t_1 : T_2}{\Gamma \vdash \texttt{inr}\ t_1\ \texttt{as}\ T_1 + T_2 : T_1 + T_2} \quad \text{(T-Inr)}$$