A Theory of Type Polymorphism in Programming(2)
〜〜 概要 〜〜

児玉靖司
東京理科大学理工学部情報科学科

3 簡単な言語とその型

3.1 言語 Exp

- 変数 x は，識別子を (range over) し $x \in Id$ とする．
- 言語 Exp は，以下の文法からなる．

$$e \quad ::= \quad x \mid (e\, e') \mid if\ e\ then\ e'\ else\ e'' \mid \\ \lambda x \cdot e \mid fix\ x \cdot e \mid let\ x = e\ in\ e'.$$

- $(e\, e')$ は適用を意味し，$fix\ x \cdot e$ は $\lambda x \cdot e$ の最小不動点を意味し，最後は $let - in - end$ を意味する．
- d, e, f または，prime が付いた識別子は Exp を range over する．
- 定数は省略．
- Exp に対する通常の表示意味を与えることができる．そこでは"wrong" (実行時の失敗を意味する) を含んでいる．
- この小さな言語では，失敗とは，条件式 (conditional) での条件部に論理値でない値が出現した時と，適用の演算として関数値でない値が出現した場合のみである．
- 各意味領域は，cpos(complete partial orders) である．ある cpo D[9] は，半順序集合で，(a) 最小値 $\perp_D$ を持つ．(b) D の各部分集合 (有向な) は，D 中に最小上界を持つ．
- 基礎領域の集合 $\{B_i\}$ に対して，$B_0 = T$ とすると，以下のようになる．
- 以下のように再帰的に定義する．

$$\begin{aligned} V &= B_0 + B_1 + \cdots + F + W \\ F &= V \to V \quad (V \text{ から } V \text{ への連続関数}) \\ W &= \{\cdot\} \quad (\text{エラー}) \end{aligned}$$

この解は，Scott[15] により確かめられた (彼は，完備束を用いた，cpo を使って解いたのは Plotkin[11] である)．

- 意味関数 $\mathcal{E} \in Exp \to Env \to V$(ここで，$Env = Id \to V$，環境の領域とする)．
- η は環境を range over するとする．
- $\mathcal{E}$ の定義に関しては，Scott と Strachey[16] が用いた使い方をする．
  1. もし $d \in D$ の時，V 中に d があるということは，D の V への injection に対して，d が image であることである．
  2. もし $v \in V$ の時，

$$\begin{aligned} v \in D &= true \quad (\text{ある } d \in D \text{ に対して } V \text{ で} \\ &\qquad v = d \text{ の時}) \\ &= \perp_T \quad (v = \perp_V \text{の時}) \end{aligned}$$

  3. もし $v \in V$ の時，

$$\begin{aligned} v|D &= d \quad (\text{ある } d \in D \text{ に対して } V \text{ の中で} \\ &\qquad v = d \text{ の時}) \\ &= \perp_D \quad (\text{その他}) \end{aligned}$$

- 環境 $\eta' = \eta\{v/x\}$ は，$\eta'(x) = v$ 以外は η と同一．
- V の中の値 $\cdot$ は，$(\cdot \in W)$ で "wrong" を示す．
- 条件関数 $COND \in T \to V \to V \to V$，$COND\ t \to v, v'$ を定義する．

$$\begin{array}{ll} v & \text{もし } t = true \\ v' & \text{もし } t = false \\ \perp_V & t = \perp_T \end{array}$$

3.2 Exp の意味同値

- 以下の等式で，

$$\begin{aligned} \mathcal{E}[\![x]\!]_\eta &= \eta[\![x]\!] \\ \mathcal{E}[\![(e_1\, e_2)]\!]_\eta &= v_1 \in F \to (v_2 \in W \to wrong, \\ &\qquad (v_1|F)v_2), wrong \end{aligned}$$

$$
\begin{aligned}
& where\ v_i\ is\ \mathcal{E}[\![e_i]\!]_\eta \quad (i=1,2). \\
\mathcal{E}[\![if\ e_1\ then\ e_2 & \\
else\ e_3\ ]\!]_\eta &= v_1 \in B_0 \to (v_1|B_0 \to v_2, v_3) \\
& \quad , wrong \\
& \quad where\ v_i\ is\ \mathcal{E}[\![e_i]\!]_\eta \quad (i=1,2,3) \\
\mathcal{E}[\![\lambda x \cdot e]\!]_\eta &= (\lambda v \cdot \mathcal{E}[\![e]\!]_{\eta\{v/x\}})\ in\ V \\
\mathcal{E}[\![fix x \cdot e]\!]_\eta &= Y(\lambda v \cdot \mathcal{E}[\![e]\!]_{\eta\{v/x\}}) \\
\mathcal{E}[\![let\ x = e_1\ in\ e_2]\!]_\eta &= v_1 \in W \to wrong, \\
& \quad \mathcal{E}[\![e_2]\!]_{\eta\{v_1/x\}} \\
& \quad where\ v_1 = \mathcal{E}[\![e_1]\!]_\eta
\end{aligned}
$$

1. ここで Y は，最小不動点演算子である．多くの言語では，$fix\ f \cdot e$ の中の e は，抽象 $\lambda y \cdot e'$ に制限されている．
2. そのため，

$$let\ \ f = fix\ f \cdot (\lambda y \cdot e')$$

   は，以下のように書くことができる．

$$let\ \ rec\ f(y) = e'$$

3. $\mathcal{E}$ のもとでは，$let\ x = e_1\ in\ e_2$ は，$(\lambda x \cdot e_2)e_1$ と同じ意味であることを確かめることができる．しかし，我々の目的の一つは，型規則である．<略>
4. $(e_1\ e_2)$ の意味論は，値渡しである．$v_2 \in W$ のテストは，e_2 が $\perp_V$ とすると，$(e_1\ e_2)$ が $\perp_V$ であることを意味する．このテストを省略すると，名前呼出しとなる．意味論的健全性ではこの場合は同じである．

3.3 型に関する議論

- 式の新しいクラスとして，型を定める．型を持つ値により何を意味するかを考える．
- 値は，多くの型を持つかもしれないし，全く型を持たないかもしれない．“wrong” は型を持たない．
- しかし関数値は型を持つ．引数は正しい kind(型) に適用する限り，正しい kind(型) を生成する．“wrong” にはならない．
- Exp の式 (適切な環境において) は，型を持つ値として評価することができ，wrong としてなることはない，ことを示す．
- 式は (環境に対して) “well-typing” であるということは，その式とサブ式に，ある一定の規則を用いて型を割り当てることができる，ことを示す．
- 2 つの重要な作業がある．一つは，正しい型割り当てを持つ式は “wrong” には行かない，ことである．
- もう一つは，正しい型割り当てを見つけることである．この作業は，しばしば，型チェックといわれる．もちろん，これは，与えられた型割り当てが，正しいことを verify することも意味する．
- 第 4 節では，与えられたプログラムに対して，正しい型のクラスを見る (我々は多相型を許しているので，そのクラスは一般には無限にある)．
- アルゴリズム (それが成功すれば，正しい型割り当てを生成する) を与える．
- 他の正しい型割り当ては，その substitution instance であることを示す (型変数へ型を割り当てる置換を示す)．

3.4 型とその意味

- 型の文法を以下のように定義する．
  1. ι_0，ι_i，$\cdots$ は，基礎型 (basic type) である．B_i の一つ．
  2. 型変数を α，β，γ，$\cdots$ で示す．
  3. ρ，σ を型とする時 $\rho \to \sigma$ も型である．
- 単相型 (monotype) は，型変数を含んでいない型を示す．μ，ν，π で示す．
- 多相型 (polytype) とは，型変数を含んでいるかもしれない，または，含んでいる型を示す時に使う．
- 最初に，単相型の意味を与える．値 $v \in V$ が，単相型 μ を持つ時 $v : \mu$ と書く．
  1. $v : \iota_i$ iff $v = \perp_V$ または $v \in B_i$
  2. $v : \mu \to \nu$ iff $v = \perp_V$ または，$v \in F$ かつ $(v|F)u : \nu$ ($u : \mu$ の時つねに)
- 型を持たない値が多く存在する．

$$wrong$$

$$(\lambda v \in V \cdot wrong)\ \mathrm{in}\ V$$

$$(\lambda v \in V \cdot v \in B_0 \to (v|B_0 \to x\,\mathrm{in}\,V, y\,\mathrm{in}\,V), wrong)\,\mathrm{in}\,V$$

ここで，例えば $x \in B_1, y \in B_2$

- 最後の例では $y \in B_1$ ならば，関数は型 $\iota_0 \to \iota_1$ を持つ．
- 多くの型を持つ値もある．$(\lambda v \in V \cdot v)\ \mathrm{in}\ V$．各 μ に対して $\mu \to \mu$．
- $\perp_V$ は，全ての型を持つ (全ての型を持つ唯一の値)．
- 型のこの表現は，Scott[17] より引用した．実際，Scott (Curry の後) は，functionality と呼んだものは，retract とは区別した．
- 一時的に，「型を値の集合である」(値は型を持つ) とすると，downward closed で，directed complete であることを示すことができる．
  1. $\forall v, v' \in V \cdot (v : \mu \text{かつ } v' \sqsubseteq v) \Rightarrow v' : \mu$,
  2. V の各有向部分集合 X に対して，$(\forall v \in X \cdot v : \mu) \Rightarrow \sqcup X : \mu$.

- retract は，第二の性質を持つが，最初の性質は保持しない．最近では Shamir と Wadge[18] により両方の性質を保持したあらゆる集合に対する型を定義した．そして彼らは，値 v を型 $\{v' : v' \sqsubseteq v\}$ を持つ，と定義した．
- 多相型に対する意味は以下の通りである．$\rho \ll \sigma$ と書き，ρ は σ から得ることができることを示す (型変数に型を置換することにより)．
- $\ll$ は反射的，推移的である．例えば，

$$\mu \to \mu \ll \alpha \to \alpha \ll \beta \to \beta \ll \alpha \to \beta$$

- しかし，

$$\alpha \to \beta \ll \beta \to \beta \quad (\alpha = \beta\text{でない})$$

- $$v : \rho \text{ iff } \forall \mu \ll \rho \cdot v : \mu.$$

  例えば，

$$(\lambda v \cdot v)\, in\, V : \alpha \to \alpha.$$

- 多相型は，V の部分集合に基いている．これらも directed complete である．
- 全称限量 (universary quantify) が outermost に存在する．型変数 α は，monotype を range over する．
- 以下のような型は考えない．

$$(\forall \alpha \cdot \alpha \to \alpha) \to (\forall \alpha \cdot \alpha \to \alpha)$$

  Reynolds[12] (richer notion) での難しさは避ける．
- 以下の簡単な性質を必要とする．定義からすぐに得ることができる．

  命題: もし $v : \rho$ かつ $\tau \ll \sigma$ の時，$v : \tau$

  命題: もし $v : \sigma \to \tau$ かつ $v' : \sigma$，ならば $(v|F)v' : \tau$

3.5 型割り当て

- well-typed な式は，“go wrong” にならないという定理のための基礎を用意する．式を型付けすることは何を意味するかを定義する．
- 式にある型変数に型を与える型環境の意味を必要とする．
- $prefix\ \ p$ は有限の列とする．そのメンバは，$let\ \ x$，$fix\ \ x$，λx (x は変数) とする．
- prefixed expression (pe) は，$p|e$ という形を持つ．e での自由変数は，p のメンバとして出現する．
- $\cdot$ により prefix のメンバを区別する．
- 各 pe は，以下に定めるようにサブ pe をもつ．
  1. $p|e$ は，それ自身以外にサブ pe を持たない．
  2. $p|(ee')$ は，$p|e$ と $p|e'$ のサブ pe を持つ．
  3. $p|(if\, e\, then\, e'\, else\, e'')$ は，$p|e$，$p|e'$ と $p|e''$ のサブ pe を持つ．
  4. $p|(\lambda x \cdot e)$ は，$p \cdot \lambda x|e$ のサブ pe を持つ．
  5. $p|(fix\, x \cdot e)$ は，$p \cdot fix\, x|e$ のサブ pe を持つ．
  6. $p|(let\, x = e\, in\, e')$ は，$p|e$ と $p \cdot let\, x|e'$ のサブ pe を持つ．
- 例えば，$\lambda y|(let\ f = \lambda x \cdot (xy)\ in\ (fy))$ は，(自分以外に) 以下のサブ pe を持つ．
  - $\lambda y|\lambda x \cdot (xy)$
  - $\lambda y \cdot \lambda x|(xy)$
  - $\lambda y \cdot \lambda x|x$
  - $\lambda y \cdot \lambda x|y$
  - $\lambda y \cdot let\ f|(fy)$
  - $\lambda y \cdot let\ f|f$
  - $\lambda y \cdot let\ f|y$
- サブ pe とは，閉じるようにした変数束縛の全てで prefix された サブ式である．
- $\lambda x \cdot (xy)$ は $letf$ では，閉じていない．
- p のメンバ $let\ x$，$fix\ x$ または λx が active とは，p の中でその右側で 変数 x が出現しないときである．
- pe $p|e$ の型付けとは，p の各要素，とサブ式，各 x に対する λx，$fix\, x$，$let\, x$ に対して，型を割り当てることである (サブ式 $(let\ x = e'\ in\ e'')$ において，$let\, x$ と e' に同じ型を割り当てる)．
- 説明した pe の一つの型付けとして以下のようになる．

$$\lambda y_\alpha|(let\ f_{(\alpha\to\beta)\to\beta} = (\lambda x_{\alpha\to\beta}(x_{\alpha\to\beta}y_\alpha)_\beta)_{(\alpha\to\beta)\to\beta}$$

$$in\ (f_{(\alpha\to\gamma)\to\gamma}y_\alpha)_\gamma)_\gamma.$$

- $p|e$ の型付けを $\bar{p}|\bar{e}$，または $\bar{p}|\bar{e}_\sigma$ と書く (式 e に型 σ を割り当てたい時)．
- $\bar{p}|\bar{e}$ かつ，$\bar{p}$ または，$\bar{e}$ における束縛 $let\, x_\sigma$ において，σ 中の型変数が，閉じた λy_τ または $fix\, y_\tau$ に出現しない時，generic type ($let\, x_\sigma$ に対して) という．
- 例えば，束縛 $let\, f_{(\alpha\to\beta)\to\beta}$ に対して，β は generic であるが，α はそうでない．
- 直感的には，束縛 $let\, x_\sigma$ に対して generic な型変数は，x が使われる型の自由さの度合いを表現している．
- それらは，x の局所的な多相性を表現している．
- $let\, x_\sigma$ により閉じられた時，λ や fix の束縛がない場合は，σ にある型変数はすべて generic である．
- σ の generic instance とは，generic type が instantiate した σ の instance である．

- $\bar{p}|\bar{d}$ が standard であるということは，すべての型付けされた サブ $\bar{p'}|\bar{d'}$ に対して，$\bar{p'}$ の各メンバ $let\, x_\sigma$ の generic な型変数が $\bar{p'}|\bar{d'}$ に出現しないときである．

- もし $let\, x_\rho = \bar{e_\rho}\, in\, \bar{e'_\sigma}$ が $\bar{d}$ のサブ式であるとすると，ρ にある generic な型変数は，$\bar{e'_\sigma}$ には出現しない (もちろん $\bar{e_\rho}$ には出現する)．

- 以上より，以下のように well-typeed(wt) の概念を定義する．

  1. $\bar{p}|x_\tau$ は wt である． iff standard であり，以下のどちらかに該当する時，
     (a) $\bar{p}$ の中で λx または $fix\, x_\tau$ が active な時，
     (b) $\bar{p}$ の中で $letx_\rho$ が active で，τ が σ の generic instance である時，
  2. $\bar{p}|(\bar{e_\rho}\bar{e'_\rho})$ が wt iff $\bar{p}|\bar{e}$ かつ $\bar{p}|\bar{e'}$ が両方とも wt でかつ $\rho = \sigma \to \tau$.
  3. $\bar{p}|(if\, \bar{e_\rho}\, then\, \bar{e'_\rho}\, else\, \bar{e''_\rho})_{\tau'}$ は wt iff $\bar{p}|\bar{e}$, $\bar{p}|\bar{e'}$ かつ $\bar{p}|\bar{e''}$ は全て wt, $\rho = \iota_0$ かつ $\sigma = \tau = \tau'$.
  4. $\bar{p}|(\lambda x_\rho \cdot \bar{e_\rho})_\tau$ は wt. iff $\bar{p} \cdot \lambda x_\rho|\bar{e}$ は wt かつ $\tau = \rho \to \sigma$.
  5. $\bar{p}|(fix\, x_\rho \cdot \bar{e_\sigma})_\tau$ は wt iff $\bar{p} \cdot fix\, x_\rho|\bar{e}$ は wt かつ $\rho = \sigma = \tau$.
  6. $\bar{p}|(let\, x_\rho = \bar{e_\rho}\, in\, \bar{e'_{sigma}})_\tau$ は wt iff $\bar{p}|\bar{e}$ かつ $\bar{p} \cdot let\, x_\rho|\bar{e'}$ は共に wt，かつ $\sigma = \tau$.

- これらの再帰定義は，証明に役立つ．wt の他の特徴も時々役立つ．次の命題の証明は，自明である．

命題 3:$\bar{p}|\bar{d}$ は wt iff 以下が成り立つ場合である．

1. standard である．
2. 各束縛された x_σ の出現に対して，対応する束縛の出現は，λx_σ または $fix\, x_\sigma$ または $let\ x_\tau$ (σ は τ の generic instance である)．

3.6 置換

- 置換 S は，型変数から型へのマップを示す．S は，自然に型から型，型のついた pe から pe へのマップも示す．

- S は，以下の条件を満たす時，型変数を含むという．
  - $S\alpha \neq \alpha$
  - ある $\beta \neq \alpha$，$\alpha \in S\beta(\alpha \in \tau$ は，τ の中に α が出現することを示す)．

命題 4:もし，S が wt の $\bar{p}|\bar{d}$ の generic variables を含まないならば，$S(\bar{p}|\bar{d})$ もまた wt である．

証明

命題 3 を使う．第一に，S に関する仮定より，$S(\bar{p}|\bar{d})$ における各束縛の generic variables は，$\bar{p}|\bar{d}$ における対応する束縛のそれに相当する．β が generic でないとき，$S\beta$ が generic variable を含んでいないので，$S(\bar{p}|\bar{d})$ は standard である．

第二に，もし x_σ が $\bar{p}|\bar{d}$ の中で，λx_σ または，$fix\, x_\sigma$ に束縛される場合，$S(\bar{p}|\bar{d})$ の中の $x_{S\sigma}$ は，$\lambda x_{S\sigma}$ または，$fix\, x_{S\sigma}$ に束縛される．もし，x_σ が $let\, x_\tau$ より束縛され，$\sigma = [\rho_1/\alpha_1, \cdot, \rho_n/\alpha_n]_\tau$ (α_i は τ の generic variable とする) とすると，$S(\bar{p}|\bar{d})$ の中では，$x_{S\sigma}$ は $let\, x_{S\tau}$ により束縛され，$S\sigma = [S\rho_1/\alpha_1, \cdots, S\rho_n/\alpha_n](S\tau)$ は，$S\tau$ の generic instance である．

第三に，命題 3 の条件 (C) により，$S(\bar{p}|\bar{d})$ は簡単に確かめることができる ($S(\sigma \to \tau) = S\sigma \to S\tau$ のように)．

3.7 Well-Typed な式は，Wrong にならない

- 最初に，意味論環境 η と，型環境 (型付けされた $\bar{p}$) との簡単な関係が必要である．

  η respects $\bar{p}$ iff $let\, x_\rho$ または，λx_ρ または，$fix\, x_\rho$ が $\bar{p}$ において active であり，$\eta[\![x]\!] : \rho$ であるとき．

定理 1 (意味論的健全性)

もし η respects $\bar{p}$ かつ $\bar{p}|\bar{d}$ が well typed の時，$\mathcal{E}[\![d]\!]_\eta : \tau$.

証明

単に，簡単な構造的帰納法を使う．$\bar{d_\tau}$ についての 6 種類を考える．

1. x_τ の場合．$\bar{p}$ の中で λx_τ，$fix\, x_\tau$ が active の時，$\eta[\![x]\!] : \tau$，そのため $\mathcal{E}[\![x]\!]_\eta : \tau$．$\bar{p}$ の中で $let\, x_\sigma$ が active の時，$\eta[\![x]\!] : \sigma$．しかし，$\tau \ll \sigma$ かつ 命題 1 より，$\mathcal{E}[\![x]\!]_\eta = \eta[\![x]\!] : \tau$.
2. $(\bar{e}_{\sigma\to\tau}\ \bar{e}_\sigma)$ の場合．$\bar{p}|\bar{e}_{\sigma\to\tau}$ は wt．より $\mathcal{E}[\![e]\!]_\eta : \sigma \to \tau$．同様に，$\mathcal{E}[\![e']\!]_\eta : \sigma$．意味的等式より (wrong は型を持たない)，かつ，命題 2 より $\mathcal{E}[\![d]\!]_\eta : \tau$.
3. $(if\, \bar{e}_{\iota_0}\, then\, \bar{e'}_\sigma\, else\, \bar{e''}_\sigma)$ の場合．明らか．$\perp_V$ は，全ての型を持つ．
4. $(\lambda x_\rho \cdot \bar{e}_\sigma)_{\rho\to\sigma}$ の場合．$\bar{p} \cdot \lambda x_\rho|\bar{e}_\sigma$ は wt．ここで，$(\lambda v \cdot \mathcal{E}[\![e]\!]_{\eta\{v/x\}})$ in $V : \rho \to \sigma$ が必要である．f in $V : \rho \to \sigma$ を示すためには，$v : \rho$ の時，$fv : \sigma$ であることでは不十分である．すべての $\mu \to \nu \ll \rho \to \sigma$ に対して $f\ in\ V : \mu \to \nu$ のためには，何か必要であるか．

   $\mu \to \nu \ll \rho \to \sigma$ を仮定する．ρ と σ にある型変数のみを含む 置換 S，$\mu \to \nu = S(\rho \to \sigma)$ を考える．これらの型変数は $\bar{p} \cdot \lambda x_\rho|\bar{e}_\sigma$ では generic ではないので，命題 4 より $S(\bar{p}) \cdot \lambda x_\mu|S(\bar{e})_\nu$ が wt になる．η respect $S(\bar{p})$ (命題 1 より，$\eta[\![x]\!] : \sigma'$

かつ $\tau' \ll \sigma'$，$\eta[\![x]\!] : \tau')$ より，すべての $v : \mu$ に対して $\eta\{v/x\}$ respects $S(\bar{p}) \cdot \lambda x_\mu$ が成り立つ．
帰納法の仮定より $\mathcal{E}[\![e]\!]_{\eta\{v/x\}} : \nu$．よって $v : \mu$ は $fv : \nu$ かつ，$f\ V : \mu \to \nu$ となる．

5. $(fix\, x_\rho \cdot \bar{e}_\rho)_\rho$ の場合．$\bar{p} \cdot fix\, x_\rho | \bar{e}_\rho$ は wt.

$$v = Y(\lambda v' \cdot E[\![e]\!]_{\eta\{v'/x\}}).$$

の時 $v : \rho$ を必要とする．$v_0 = \perp_V$，$v_{i+1} = \mathcal{E}[\![e]\!]_{\eta\{v_i/x\}}$ で $v = \sqcup_i v_i$ とする．型の directed completeness より 各 i に対して $v_i : \rho$ であることだけを示せばよい．
明らかに $v_0 : \rho$．$v_i : \rho$ を仮定すると，η respect $\bar{p}$ より $\eta\{v_i/x\}$ respect $\bar{p} \cdot fix\, x_\rho$，そして，帰納法の仮定より 各 i に対して $v_{i+1} : \rho$ となる．

6. $(let\ x = \bar{e}_\rho\ in\ \bar{e'}_\sigma)_\sigma$ の場合．$\bar{p} | \bar{e}_\rho$ は wt．そのため $v = \mathcal{E}[\![e]\!]_\eta$ とすると $v : \rho$ となる．ここで，$\mathcal{E}[\![e']\!]_{\eta\{v/x\}} : \sigma$ を必要とする．
$\bar{p} \cdot let\, x_\rho | \bar{e'}_\sigma$ もまた wt．$v : \rho$ のため，$eta\{\sigma/x\}$ respects $\bar{p} \cdot let\, x_\rho$ となり，帰納法の仮定より上がいえる．

- この結果 (系として)，以下を証明した．

$$\mathcal{E}[\![d]\!]_\eta \neq wrong$$

(wrong は型を持たない)．