

1 型システム

別紙参照

2 型推論アルゴリズム

概要: 以下の 2 ステップから構成

- 制約生成: 与えられた項にたいして, (型および classifier に関する) 制約を返す.
- 制約を解く.

2.1 制約生成

これは, もとの型システム (T_1 とする) を「トップダウンでの制約生成向け型システム (T_2 とする)」に変形することであたえる.
 T_2 の設計指針:

- T_1 と T_2 は「型付けできる」という関係として等価である.
- T_2 は, term-oriented である. (結論側の式のトップレベルの形だけで, どの型付けルールを適用可能か, 一意的にわかる.)
- T_2 は, 制約生成をする. (結論側の式の要素は変数として, 「それがこういう形でなければいけない」という条件は, 制約の形で「生成」する.)

以上をどう満たすか? ポイントは, subsumption rule の適用タイミング (なるべく subsumption rule を適用するのを避けたい) である.

2.2 型システム T_2 の導入

subsumption rule が出現する場所を限定することができる. 特に, 型付け規則と, その直後に subsumption がつかわれる場合を考えてみよう. 以下で, 「var1」等といった表記は, 「もともとある var1 規則を subsumption 規則と組み合わせた形に改訂したもの」である. また, 横棒の右に書いてある Constr;... は (規則を下から上にむけて使うとき), Constr 以下の制約が生成される, という意味である.

また, 型 t_1, t_2 に対する \geq の記号は以下の意味であるが, とりあえず, (以下の意味にしたがって分解はせずに) $t_1 \geq t_2$ の形のまま, 制約として生成する.

- $\langle t_1 \rangle^{\gamma_1} \geq \langle t_2 \rangle^{\gamma_2}$ は, 「 $t_1 = t_2$ かつ $\gamma_1 \geq \gamma_2$ 」
- $\langle t \rangle^\gamma$ の形でない t_1, t_2 に対しては, $t_1 = t_2$.

(型推論のプロセスの最中では, t_1, t_2 はメタ型変数である. どちらの形かは決定できないので, $t_1 \geq t_2$ を上記の意味にしたがって, 「ほどく」ことはできない. なので, $t_1 \geq t_2$ という形のまま制約として生成する.)

(var1)

$$\frac{(x : t') \in \Gamma}{\Gamma \vdash x : t; \sigma} \text{Constr}; \Gamma \models t \geq t'$$

(var2)

$$\frac{(u : t)^{\gamma'} \in \Gamma}{\Gamma \vdash^{\gamma} u : t; \sigma} \text{Constr}; \Gamma \models \gamma \geq \gamma'$$

(const)

$$\frac{}{\Gamma \vdash^L c : t; \sigma} \text{Constr}; \Gamma \models t \geq t^c$$

t^c は定数 c が持つ型を意味し, これは環境によらずに 常に一意である. ex. *true, false, 1, 2, 3, ...*

(app)

$$\frac{\Gamma \vdash^L e_1 : t_2 \rightarrow t_1; \sigma \quad \Gamma \vdash^L e_2 : t_2; \sigma}{\Gamma \vdash^L e_1 e_2 : t; \sigma} \text{Constr}; \Gamma \models t \geq t_1$$

(lambda0)

$$\frac{\Gamma, x : t_1 \vdash e : t_2; \sigma'}{\Gamma \vdash \lambda x. e : t; \sigma} \text{Constr}; t = t_1 \xrightarrow{\sigma'} t_2$$

(lambda1)

$$\frac{\Gamma, (u : t_1)^{\gamma} \vdash^{\gamma} e : t_2; \sigma'}{\Gamma \vdash^{\gamma} \lambda u. e : t; \sigma} \text{Constr}; t = t_1 \rightarrow t_2$$

oishi said: σ' ってどうすればいいんだ...

(if)

$$\frac{\Gamma \vdash^L e_1 : \text{Bool}; \sigma \quad \Gamma \vdash^L e_2 : t; \sigma \quad \Gamma \vdash^L e_3 : t; \sigma}{\Gamma \vdash^L \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : t; \sigma} \text{Constr}; (\text{none})$$

(code-lambda)

$$\frac{\Gamma, \gamma' \geq \gamma, x : \langle t_1 \rangle^{\gamma'} \vdash^L e : \langle t_2 \rangle^{\gamma'}; \sigma}{\Gamma \vdash^L \underline{\lambda} x. e : t; \sigma} \text{Constr}; \Gamma \models t \geq \langle t_1 \rightarrow t_2 \rangle^{\gamma}$$

(reset0)

$$\frac{\Gamma \vdash e : \langle t' \rangle^{\gamma}; \langle t' \rangle^{\gamma}, \sigma}{\Gamma \vdash \text{reset0 } e : t; \sigma} \text{Constr}; \Gamma \models t \geq \langle t' \rangle^{\gamma}$$

(shift0)

$$\frac{\Gamma, k : \langle t_1 \rangle^{\gamma_1} \xrightarrow{\sigma} \langle t_0 \rangle^{\gamma_0} \vdash e : \langle t_0 \rangle^{\gamma_0}; \sigma \quad \Gamma \models \gamma_1 \geq \gamma_0}{\Gamma \vdash \text{shift0 } k \rightarrow e : t; t_2, \sigma} \text{Constr}; \Gamma \models t \geq \langle t_1 \rangle^{\gamma_1}, t_2 = \langle t_0 \rangle^{\gamma_0}$$

oishi said: $\Gamma \models \gamma_1 \geq \gamma_0$ は制約に加えるのではなく、型付け図に書くのはなぜ？

(throw0)

$$\frac{\Gamma \vdash v : \langle t_1 \rangle^{\gamma_1 \cup \gamma_2}; \sigma \quad \Gamma \models \gamma_2 \geq \gamma_0}{\Gamma, k : \langle t_1 \rangle^{\gamma_1} \xrightarrow{\sigma} \langle t_0 \rangle^{\gamma_0} \vdash \mathbf{throw} \ k \ v : t; \sigma} \text{Constr}; t \geq \langle t_0 \rangle^{\gamma_2}$$

(code)

$$\frac{\Gamma \vdash^\gamma e : t_1; \sigma}{\Gamma \vdash \langle e \rangle : t; \sigma} \text{Constr}; t \geq \langle t_1 \rangle^\gamma$$

この新しい型システム T_2 は T_1 と同じ型付けをあたえる.

型推論では、与えられた項 e に対して、 T_2 を「下から上」の向きに適用して、Constraint を生成する。(もちろん、途中で、つまってしまったら、型推論は失敗する.)

下から上に行くときに、「横棒の下には存在しないで、横棒の上、あるいは、Constraint の中にのみ存在する」型や classifier があるときは、これらを型変数や classifier 変数として生成する.

なお、code-lambda 規則での新しい classifier は、classifier 変数ではなく、code-lambda 規則の新しい classifier は、固有変数条件を満たす必要があるので、classifier 定数とする.

(この段階では、変数も定数も差はないが、制約を解消するとき、classifier 定数に対する代入はしない、という違いがある.)

3 制約の解消

制約生成が成功したとき (つまり、途中で「つかかかったり」しないとき)、以下の性質が成立するはずである.

入力を Γ, L, e, t, σ として、前章の制約生成アルゴリズムを走らせ、それが成功して C という制約を生成したとき、

- $\Gamma \vdash^L e : t; \sigma$ が導出可能ならば、 C を満たす解が存在し、
- C を満たす解が存在すれば、ある代入 θ に対して、 $\Gamma \theta \vdash^L e \theta : t \theta; \sigma \theta$ が導出可能である.

という性質が成立することができる。(これが成立すれば、もともとの型推論問題を、型制約の解消問題に帰着できたことになる.)

というわけで、制約の解消をはじめよう. 制約は以下の文法で与えられたものの有限集合である.

- $\Gamma \models t^0 \geq t^0$
- $\Gamma \models c \geq c$
- $t^0 = t^0$
- $t^1 = t^1$

ただし、ここで t^0, t^1, c は以下の文法で定義される.

$$\begin{aligned} t^0 &::= \alpha^0 \mid \mathbf{Int} \mid \mathbf{Bool} \mid t^0 \xrightarrow{c} t^0 \mid \langle t^1 \rangle^c \\ t^1 &::= \alpha^1 \mid \mathbf{Int} \mid \mathbf{Bool} \mid t^1 \rightarrow t^1 \\ c &::= \gamma \mid d \mid c \cup c \end{aligned}$$

t^0, t^1, c はそれぞれ、レベル 0 型、レベル 1 型、classifier をあらわす表現 (メタ変数) である. また、 α^i はそれぞれのレベルの型変数、 γ は classifier 変数である. また、 d は、固有変数条件をもつ classifier 変数のことであり、型推論のあいだは、これは (実質的に) 定数として扱われる (つまり、 d に対しては代入しない、classifier 変数 γ に対しては代入する.)

また、 Γ は、一般の型文脈であるが、不要な情報を落として以下の形にしてよい。

$$\Delta ::= \cdots \mid \Delta, d \geq c$$

(左辺は、固変数なので、classifier 定数である。右辺は一般の classifier 式がなんでも来る可能性がある。)

制約の解消とは、制約 (上記の形の有限集合) が与えられたとき、その「解」となる代入をとるとめることである。代入 S は、型変数 α^0, α^1 への型の代入と、classifier 変数 γ への classifier の代入とから構成される。この代入は「最も一般的」であるべきである。(定義の詳細はいまは省略)

3.1 制約の解消アルゴリズム (前半)

$t^0 = s^0$ と $t^1 = s^1$ の形の制約は、普通の型推論で解ける。それを解いた結果、 $\alpha^0, \alpha^1, \gamma$ に対する代入が生じる (か、あるいは、「解なし」という結果になる)。

$\Delta \models t^0 \geq s^0$ の形の制約は、両方ともが型変数の場合以外は、簡単に解ける。(その結果として、 $t^i = s^i$ の形の制約や、 $\Delta \models c \geq c$ の形の制約を生む可能性があるが、前者は前と同様に解けばよく、前者を解いている間にあらたに $\Delta \models t^0 \geq s^0$ の形の制約は生じない。)

ここまでの段階で (代入がいくつか出たほか)、残る制約は、以下のものだけである。

- $\Delta \models \alpha^0 \geq \beta^0$
- $\Delta \models c \geq c$

ここまでにでてきた代入はすべて、上記の制約たちに適用済みとする。(つまり、 $\alpha := \text{Int}$ という代入がでてきたら、上の式にある α はすべて Int にしておく。その結果、「代入における左辺にでてくる型変数や classifier 変数」は、上記の制約たちには、のこっていない。)

3.2 制約の解消アルゴリズム (後半)

$\Delta \models c \geq c$ の形の制約を解こう。($\Delta \models \alpha^0 \geq \beta^0$ については次の subsection で考えるので、ここでは無視する。)

この形の制約たちを、 $\Delta_i \models c_i \geq c'_i$ とすると、それぞれの Δ_i は compatible であるはずなので (ここはあとでチェックが必要) $\Delta = \Delta_1 \cup \cdots \cup \Delta_n$ という風に全部を合体させた上で、 $\Delta \models c_i \geq c'_i$ を解けばよい。

(ステップ 1: classifier 変数の除去) 使われている classifier 変数の 1 つに着目する。(どれでもよい。) それを γ とする。 $c_i \geq \gamma$ の形の制約 ($i = 1, 2, \dots, I$) とを全部あつめる。 $\gamma \geq c'_j$ の形の制約 ($j = 1, 2, \dots, J$) と、これらを消去して、かわりに、以下の制約を、すべての (i, j) に対して追加する。

$$c_i \geq c'_j$$

これにより、classifier 変数は 1 つ減る。(制約は一般には増えるかもしれない。) ステップ 1 を繰返すと、classifier 変数はなくなる。

(ステップ 2) $c_1 \cup c_2 \geq c_3$ を $c_1 \geq c_3$ と $c_2 \geq c_3$ に変換する。

これにより、不等号の左辺にある \cup の個数が 1 つ減る。ステップ 2 を繰返すと、不等号の左辺にある \cup はなくなる。

ステップ 2 の繰返しがおわると、制約は、 $\Delta \models d \geq c$ の形になる。

(ステップ 3) ここで「 d は atomic」という仮定をおく。これについてはあとで吟味する。これは、「 $d \geq c_1 \cup c_2$ ならば $d \geq c_1$ または、 $d \geq c_2$ 」という内容である。これらの逆向きは、いつでも成立するので、結局、ここの「ならば」は「同値」といってもよい。

これもちいて、 $d \geq c$ の右辺も分解できて、

$$\Delta \models d_1 \geq d'_1 \vee \cdots \vee d_n \geq d'_n$$

となり、さらに Δ も $d_1 \geq d_2$ の形を「かつ」と「または」でつないだ形になるはずである。
これは decidable なので、「解があるかどうか」も decidable である。

3.3 制約の解消アルゴリズム (後半のおまけ)

実は、まだ、 $\Delta \models \alpha^0 \geq \beta^0$ という制約がのこっていた。

これを最後まで残した (解かなかった) のは、以下の 2 つの可能性があるのである。これはどうするか？

- $\alpha^0 = \beta^0$
- $\alpha^0 = \langle t \rangle^{\gamma_1}, \beta^0 = \langle t \rangle^{\gamma_2}, \Delta \models \gamma_1 \geq \gamma_2$