7　型推論

第 5 章で見たように、プログラムの型に関して、微妙に異なるいくつかの判定問題がある。その主要なものを再掲すると以下の通りである。

- Γ, M, A が与えられたとき，$\Gamma \vdash M : A$ が導けるか？ (type checking, 型検査問題)
- M が与えられたとき，$\Gamma \vdash M : A$ が導ける Γ, A があるか？ (type inference, 型推論問題その 1)
- Γ, M が与えられたとき，$\Gamma \vdash M : A$ が導ける A があるか？ (type inference, 型推論問題その 2)

本講義で扱う体系 (単純型付きラムダ計算、関数プログラミングの体系、および、後者を多相型に拡張したもの) では、上記の判定問題が決定可能となり、それぞれに対して，判定を行なう (有限時間で必ず停止する) アルゴリズムが存在する。

これらすべてを紹介すると膨大になるので、ここでは、関数プログラミングの体系に対する型推論のアルゴリズムについて学習する。

7.1　型変数の導入

計算体系 CoreML の型で特徴的なことは、1 つの項が複数の型を持つことである。

たとえば、$\lambda f.\lambda x.\ f@(f@x)$ という項は、$(\mathtt{int} \to \mathtt{int}) \to (\mathtt{int} \to \mathtt{int})$ という型を持つが、$(\mathtt{bool} \to \mathtt{bool}) \to (\mathtt{bool} \to \mathtt{bool})$ という型も持つ。一般に、$(\square \to \square) \to (\square \to \square)$ という形の任意の型を持つ。

また，$\lambda x.\ (\mathtt{left}(x), \mathtt{right}(x))$ という項は、$(\mathtt{int} \to \mathtt{bool}) \to (\mathtt{bool} \to \mathtt{int})$ という型を持つが、$(\mathtt{int} \to \mathtt{int}) \to (\mathtt{int} \to \mathtt{int})$ という型も持つ。もっと一般に、$(\square_1 \times \square_2) \to (\square_2 \times \square_1)$ という形の任意の型を持つ。

どちらのケースでも，「一般に」と書いた型は，型のパターンであり，それらの $\square$ 等の中に任意の型をいれることにより，具体的な型がでてくる．

型推論アルゴリズムにおいては，$\lambda f.\lambda x.\ f(f(x))$ という項に対して，$(\mathtt{int} \to \mathtt{int}) \to (\mathtt{int} \to \mathtt{int})$ のような特定の型ではなく，$(\square \to \square) \to (\square \to \square)$ という型のパターンを求めたい．このような，型のパターンにおける穴 $(\square)$ をあらわすため，型変数 $\alpha, \beta, \gamma, \cdots$ を用いる．

よって，本節においては、型は、 CoreML の定義に型変数を加えたものとする。上記の 2 つの型パターンは、型変数を使った型として、$(\alpha \to \alpha) \to (\alpha \to \alpha)$ および $(\alpha \to \beta) \to (\beta \to \alpha)$ と表すことができる。なお、型変数をどう選んでも，型のパターンとしては同じなので，1 つ目の $(\beta \to \beta) \to (\beta \to \beta)$ などと表してもよい．

このように、「型変数を含んでもよい型」のことを、型スキーム (type scheme) と呼ぶこともあるが、本章では、単に型と呼ぶ。

本章における型の定義:

$$A, B ::= \alpha \mid \mathtt{int} \mid \mathtt{bool} \mid A \times B \mid A \to B$$

ただし，α は型変数をあらわし，型変数は無限個あるとする．

7.2 型代入

定義 14 [型変数に対する代入] 型変数に対する代入 (単に「型代入」とも言う) とは，いくつかの型変数 $\alpha_1, \alpha_2, \cdots, \alpha_n$ (ただし、α_i は互いに相異なる)、および、それと同じ個数の (型変数を含むかもしれない) 型 $A_1, A_2, \cdots, A_n$ に対して，$[\alpha_1 := A_1, \alpha_2 := A_2, \cdots \alpha_n := A_n]$ の形をした表現である．

定義 15 [型代入の適用] 型代入 Θ が型変数 α を A に対応付けるとき (つまり，$\Theta = [\cdots, \alpha := A, \cdots]$ の形であるとき，$\Theta(\alpha) = A$ と定義し，α が Θ で対応付けられていないとき，$\Theta(\alpha) = \alpha$ と定義する．

この定義を，一般の型に対して以下のように拡張する．

$$\Theta(\texttt{int}) \stackrel{\text{def}}{=} \texttt{int}$$
$$\Theta(\texttt{bool}) \stackrel{\text{def}}{=} \texttt{bool}$$
$$\Theta(A \to B) \stackrel{\text{def}}{=} \Theta(A) \to \Theta(B)$$
$$\Theta(A \times B) \stackrel{\text{def}}{=} \Theta(A) \times \Theta(B)$$

型環境 $\Gamma = x_1 : A_1, \cdots, x_n : A_n$ に対しては、

$$\Theta(\Gamma) \stackrel{\text{def}}{=} x_1 : (\Theta(A_1)), \cdots, x_n : (\Theta(A_n))$$

と定義する．

たとえば，$\Gamma = x : \alpha \to \beta,\ y : \beta \times \gamma$ および $\Theta = [\alpha := \texttt{int} \to \delta,\ \beta := \texttt{bool}]$ のとき，$\Theta(\Gamma) = x : (\texttt{int} \to \delta) \to \texttt{bool},\ y : \texttt{bool} \times \gamma$ である．

7.3 代表的な型

先に述べたように、項 $\lambda x.\ (\texttt{left}(x), \texttt{right}(x))$ は $(\texttt{int} \to \texttt{bool}) \to (\texttt{bool} \to \texttt{int})$ や $(\texttt{int} \to \texttt{int}) \to (\texttt{int} \to \texttt{int})$ という型を持つ。また、型変数を含む型として、$(\alpha \to \beta) \to (\beta \to \alpha)$ や $(\beta \to \beta) \to (\beta \to \beta)$ も上記の項の型となっている。

これらの型のうち、$(\alpha \to \beta) \to (\beta \to \alpha)$ が代表的な型、あるいは、最も「良い」型であると言える。なぜなら、以下のように、この型の α, β に適当な型を代入すると、他の全ての型が得られるからである。

$$\begin{aligned}
[\alpha := \texttt{int},\ \beta := \texttt{bool}]((\alpha \to \beta) \to (\beta \to \alpha)) &= (\texttt{int} \to \texttt{bool}) \to (\texttt{bool} \to \texttt{int}) \\
[\alpha := \texttt{int},\ \beta := \texttt{int}]((\alpha \to \beta) \to (\beta \to \alpha)) &= (\texttt{int} \to \texttt{int}) \to (\texttt{int} \to \texttt{int}) \\
[\alpha := \beta]((\alpha \to \beta) \to (\beta \to \alpha)) &= (\beta \to \beta) \to (\beta \to \beta)
\end{aligned}$$

型推論アルゴリズムにおいて我々が求めたいのは、この「代表的な型」、つまり、$(\alpha \to \beta) \to (\beta \to \alpha)$ のような型であり、それに型代入を適用すると、他のすべての型が得られるものである。

7.4 型推論問題の定式化

ここまでの準備をもとに、型推論問題を定式化することにする。

型推論問題の定式化:

- 型文脈 Γ と (型がつかないかもしれない) 項 M に対して、$\Theta(\Gamma) \vdash M : A$ が導出できる Θ と A が存在するかどうかを判定する。(それが導出できる Θ と A のことを、Γ, M に対する型付けと呼ぶことにする[a])。
- そのような型付けが存在する場合、そのような Θ と A の中で代表的 (Principal) である Θ_0 と A_0 を求める。

[a] ここでは、Θ と A をセットにして「型付け」と呼んだが、これは便宜上のものである。通常、「型付け」という言葉は、与えられた M に対して $\Gamma \vdash M : A$ が成立する Γ と A のセットのことを意味する。

ここで、Γ と M に対して、Θ_0 と A_0 が代表的な型付けであるとは、以下の条件が両方とも成立することである。

- $\Theta_0(\Gamma) \vdash M : A_0$ が導出できる.
- $\Theta(\Gamma) \vdash M : A$ が導出できるような任意の Θ, A に対して、ある型代入 Φ があって、$\Phi(\Theta_0(\Gamma)) = \Theta(\Gamma)$ および $\Phi(A_0) = A$ が成立する。

後半の条件は若干わかりにくかもしれないが、「Θ_0, A_0 に適当な型代入を適用すると、他のすべての型付け Θ, A が得られる」ということであり、要するに、Θ_0, A_0 が最も一般的、つまり、代表的であるということである。

いくつか例を述べる。

例 18　$\Gamma = x : \alpha \to \beta,\ y : \gamma$ および $M = \texttt{if}\ y\ \texttt{then}\ x@10\ \texttt{else}\ 20$ に対して、$\Theta = [\alpha := \texttt{int},\ \beta := \texttt{int},\ \gamma := \texttt{bool}]$, $A = \texttt{int}$ とすると、(Θ, A) は、その型付けの 1 つである。実際に、$\Theta(\Gamma) = x : \texttt{int} \to \texttt{int},\ y : \texttt{bool}$ であり、$x : \texttt{int} \to \texttt{int},\ y : \texttt{bool} \vdash \texttt{if}\ y\ \texttt{then}\ x@10\ \texttt{else}\ 20 : \texttt{int}$ が導出可能である。

なお,(Θ, A) は、上記の Γ, M に対する代表的な型付けである。

例 19　$\Gamma = x : \alpha,\ y : \beta$ および $M = x@(x@y)$ に対して、$\Theta_1 = [\alpha := \texttt{int} \to \texttt{int},\ \beta := \texttt{int}]$, $A_1 = \texttt{int}$ とすると、$(\Theta_1,\ A_1)$ は、その型付けの 1 つである。[2014/11/13 誤植修正: $x@(x@y)$ とすべきところ，配布資料では $(x@y)@y$ となっていました．]

また、$\Theta_2 = [\alpha := \texttt{bool} \to \texttt{bool},\ \beta := \texttt{bool}]$, $A_2 = \texttt{bool}$ とすると、$(\Theta_2,\ A_2)$ は、その型付けの 1 つである。

さらに、$\Theta_3 = [\alpha := \beta \to \beta]$, $A_2 = \beta$ とすると、$(\Theta_3,\ A_3)$ は、その型付けの 1 つであるとともに、代表的な型付けである。

例 20　$\Gamma = x : \alpha$ および $M = (\texttt{right}(x), \texttt{left}(x))$ に対して、$\Theta_1 = [\alpha := \texttt{int} \times \texttt{bool}]$, $A_1 = \texttt{bool} \times \texttt{int}$ とすると、$(\Theta_1,\ A_1)$ は、その型付けの 1 つである。

また、$\Theta_2 = [\alpha := \beta \times \beta]$, $A_2 = \beta \times \beta$ とすると、$(\Theta_2,\ A_2)$ は、その型付けの 1 つである。

さらに、$\Theta_3 = [\alpha := \gamma \times \delta]$, $A_3 = \delta \times \gamma$ とすると、$(\Theta_3,\ A_3)$ は、その型付けの 1 つである。これは、代表的なな型付けである。

例 21　$\Gamma = x : \alpha,\ y : \beta$ および $M = (x@y) + (y@x)$ に対する型付けは存在しない。

この節の最後に、Γ と M の型付けについての定理を、証明なしで紹介する。

定理 12（型推論問題の決定可能性） 本資料の単純型付きラムダ計算の体系、計算体系 CoreML、および、後述する多相型の体系では、型文脈 Γ と項 M に対する型付けが存在するかどうかは決定可能である。(それを有限時間で判定するアルゴリズムが存在する。)

定理 13（Principal Type の存在） 本資料の単純型付きラムダ計算の体系、計算体系 CoreML、および、後述する多相型の体系では、型文脈 Γ と項 M に対する型付けが存在する場合、その代表的な型付けが存在する。

ちなみに、多相型の体系以外の 2 つについては、Principal Type より強い性質である Principal Typing という性質も成立する。これは「型推論問題その 1」に対応し、M だけが与えられた時、$\Gamma \vdash M : A$ となる Γ と A が存在すれば、そのうち代表的なものが存在する、という定理である。

7.5 型推論アルゴリズムの概要

計算体系 CoreML に対して、前節の型推論問題を具体的に解くアルゴリズムを与えよう。
その概要は以下の通りである。

型推論アルゴリズム: 型文脈 Γ と (型がつかないかもしれない) 項 M を入力とし、「型付けできる/型付けできない」という情報と、型付できる場合は、その代表的な型付け Θ と A を出力する。

- ステップ 1. Γ と M に対して、型推論図を下から構成し、型に関する制約を生成する。
- ステップ 2. この制約を単一化アルゴリズムにかけて、制約を解消し、答えを得る。

ここで、型に関する制約 (constraint) とは、型に対する等式の集合であり、具体例は後述する。

7.6 型推論のステップ 1: 制約生成

制約生成アルゴリズムは、型付け図を下から構成する手順に従うだけである。一般的に書くほどの複雑さはないので、ここでは、具体的な例をあげる。

例 22 型環境 $\Gamma = y : \alpha$ と、項 $M = \lambda f.\lambda x.\ f@(x + y)$ が与えられたとする。

- フレッシュな型変数 (他で使われていない型変数) を 1 つ選ぶ。ここでは β とする。
- $\Gamma \vdash M : \beta$ の型導出の最後 (一番下) で使われた規則は lambda ルールなので、以下の形である。

$$\frac{y : \alpha,\ f : \square_1 \vdash \lambda x.\ f@(x + y) : \square_2}{y : \alpha \vdash M : \beta}\ \text{lambda}$$

ここで $\square_1, \square_2$ には何らかの型がはいるが、まだわからないので、フレッシュな型変数を 2 つ選びそこにいれる。ここでは、γ と δ とする。これにより、

$$\frac{y : \alpha,\ f : \gamma \vdash \lambda x.\ f@(x + y)\ :\ \delta}{y : \alpha \vdash M : \beta}\ \text{lambda}$$

となる。また、これが lambda ルールの正しい適用となるためには、$\beta = \gamma \to \delta$ でなければならない。そこで $\beta = \gamma \to \delta$ を、制約として残す。

- 次に、$y:\alpha,\ f:\gamma \vdash \lambda x.\ f@(x+y)\ :\ \delta$ を導いた最後の規則は lambda ルールなので、以下の形である。この場合も、わからない型が 2 つ出現するので、型変数を 2 つ選び、ϵ, ϕ とする。

$$\frac{y:\alpha,\ f:\gamma,\ x:\epsilon \vdash f@(x+y)\ :\ \phi}{y:\alpha,\ f:\gamma \vdash \lambda x.\ f@(x+y)\ :\ \delta}\ \text{lambda}$$

  $\delta = \epsilon \to \phi$ を制約に加える。
- 次に、最後に使った規則は apply ルールなので、以下の形である。この場合は、わからない型が 1 つなので、型変数を 1 つ選び ρ とする。

$$\frac{y:\alpha,\ f:\gamma,\ x:\epsilon \vdash f\ :\ \rho \to \phi \quad y:\alpha,\ f:\gamma,\ x:\epsilon \vdash x+y\ :\ \rho}{y:\alpha,\ f:\gamma,\ x:\epsilon \vdash f@(x+y)\ :\ \phi}\ \text{apply}$$

  ここでは、制約に追加するものはない。
- 次に、$\cdots \vdash f:\rho \to \phi$ の導出の最後に使った規則は var ルールである。この場合、新しい型変数の生成はない。$(f:\rho \to \phi) \in (y:\alpha,\ f:\gamma,\ x:\epsilon)$ となるためには、$\rho \to \phi = \gamma$ が必要であるので、これを制約に加える。
- 次に、$\cdots \vdash x+y:\rho$ の導出の最後に使った規則は plus ルールである。

$$\frac{y:\alpha,\ f:\gamma,\ x:\epsilon \vdash x\ :\ \texttt{int} \quad y:\alpha,\ f:\gamma,\ x:\epsilon \vdash y\ :\ \texttt{int}}{y:\alpha,\ f:\gamma,\ x:\epsilon \vdash x+y\ :\ \rho}\ \text{plus}$$

  この部分が正しい型付けとなるためには、$\rho = \texttt{int}$ が必要であるので、これを制約に加える。
- 残りは、x と y の部分であり、f の時と同様に考えると、$\epsilon = \texttt{int}$ および $\alpha = \texttt{int}$ が必要となり、これらを制約に加える。
- 以上で、M の型付け図は完成である。ここまでで生成された制約の集合は、以下の通りである。

$$\begin{aligned} \beta &= \gamma \to \delta \\ \delta &= \epsilon \to \phi \\ \rho \to \phi &= \gamma \\ \rho &= \texttt{int} \\ \epsilon &= \texttt{int} \\ \alpha &= \texttt{int} \end{aligned}$$

  作り方から明らかなように、これらの等式すべてが同時に満足される解が存在することと、$\Gamma \vdash M:\beta$ が導出できることは同値である。

7.7 型推論のステップ 2: 制約解消 (単一化)

ステップ 1 で生成された型制約を解くアルゴリズムについて述べる。型制約は、型変数を未知変数とする連立方程式であり、その解は、型変数がどういう型になっているかを示すもの、つまり、型代入として表される。

前節の例では以下の等式の集合 E が生成された。ただし、ここでは「まだ解かれていない方程式」という側面を強調するため、等式の「$=$」という記号を「$\stackrel{?}{=}$」に置きかえている。

$$E = \{\beta \stackrel{?}{=} \gamma \to \delta,\ \delta \stackrel{?}{=} \epsilon \to \phi,\ \rho \to \phi \stackrel{?}{=} \gamma,\ \rho \stackrel{?}{=} \texttt{int},\ \epsilon \stackrel{?}{=} \texttt{int},\ \alpha \stackrel{?}{=} \texttt{int}\}$$

この等式集合の解の 1 つは、次の型代入 Θ である。

$$\begin{aligned}\Theta =&[\alpha := \mathtt{int},\ \beta := (\mathtt{int} \to \phi) \to (\mathtt{int} \to \phi),\\ &\gamma := \mathtt{int} \to \phi,\ \delta := \mathtt{int} \to \phi,\ \epsilon := \mathtt{int},\ \rho := \mathtt{int}]\end{aligned}$$

この Θ が、確かに上記の型制約の解になっていることを確かめるには、以下の等式が実際に成立していること (左右両辺の型が完全に同一の表現であること) を確かめればよい。

$$\begin{aligned}\Theta(\beta) &= \Theta(\gamma \to \delta)\\ \Theta(\delta) &= \Theta(\epsilon \to \phi)\\ \Theta(\rho \to \phi) &= \Theta(\gamma)\\ \Theta(\rho) &= \Theta(\mathtt{int})\\ \Theta(\epsilon) &= \Theta(\mathtt{int})\\ \Theta(\alpha) &= \Theta(\mathtt{int})\end{aligned}$$

一般に、型制約 (型に関する等式の集合) $\{A_1 \stackrel{?}{=} B_1, \cdots, A_n \stackrel{?}{=} B_n\}$ と型代入 Θ が、$\Theta(A_1) = \Theta(B_1), \cdots, \Theta(A_n) = \Theta(B_n)$ を満たす時 (これらの等式の左右両辺がそれぞれ同一の型となる時)、Θ はこの型制約の解 (の 1 つ) と言う。型制約に対して、その解が存在するかどうかを問う問題を「単一化問題 (unification problem)」と言う。

Robinson が考案した単一化アルゴリズム (unification algorithm) は、制約を満たす解 (より正確には、「制約を満たす解のうち代表的なもの」) を具体的に求めるアルゴリズムであり、その概要は以下の通りである。

`unify`(E, Θ): E は型制約 (型に関する等式の有限集合)、Θ は型代入で、型代入を返す。

(1) $E = \{\}$ のとき、「Θ という解あり」を答として返す。
(2) $E \neq \{\}$ のとき、E から任意の等式 1 つを選び $A \stackrel{?}{=} B$ とする。$E_1 = E - \{A \stackrel{?}{=} B\}$ とおく。
  (2-1) $A = B$ の時、つまり、もともと A と B が同一の型の時、`unify`(E_1, Θ) を呼び出して、その答を返す。
  (2-2) $A \neq B$ で、A が型変数のとき、
    (2-2-1) 型 B に A が現れるなら、「解なし」を答として返す。
    (2-2-2) 型 B に A が現れないなら、$\Theta_1 = [A := B]$ と置いた上で、`unify`$(\Theta_1(E_1),\ \Theta_1(\Theta))$ を計算し、その答を返す。
  (2-3) $A \neq B$ で、B が型変数のとき、A と B を逆にして 1 つ前のケースを適用。
  (2-4) $A \neq B$ で、$A = A_1 \times A_2$, $B = B_1 \times B_2$ のとき、$E_2 = E_1 \cup \{A_1 = B_1,\ A_2 = B_2\}$ と置き、`unify`(E_2, Θ) を呼び出し、その答を返す。
  (2-5) $A \neq B$ で、$A = A_1 \to A_2$, $B = B_1 \to B_2$ のとき、1 つ前のケースと同様。
  (2-6) 上記のいずれのケースでもないとき、「解なし」を答として返す。

このアルゴリズムで、やや難しいポイントを解説する。

まず、`unify` の第二引数 Θ は、「作りかけの答 (型代入)」を保持するためのものである。`unify` を最初に呼び出す時は、第二引数は空の型代入 [ ] とするが、再帰呼び出しにおいて、次第に大きな型代入となり、単一化が終了する時には答として返される。

上記のアルゴリズムにおける「$A \neq B$ で、A が型変数で、B に A が現れる」ケースは、たとえば、$A = \alpha$,

$B = \beta \times \alpha \to \texttt{int}$ という場合である。この場合、α と β に何を代入しても、A と B が一致することはないので、「解なし」という答を返す[*17]。

「$A \neq B$ で、A が型変数で、B に A が現れない」ケースは、A は型変数なのでこれを α とおくと、型代入 $[\alpha := B]$ をすることにより、A と B が一致する。このとき、残りの等式 E_1 に含まれる α には、B を代入 (上記のアルゴリズムで、$\Theta_1(E_1)$ と表されている) すると共に、「作りかけの解」Θ に、$[\alpha := B]$ を追加 ($\Theta_1(\Theta)$ と表されている) する必要がある。

ここで、等式集合 $E = \{A_1 \stackrel{?}{=} B_1, \cdots, A_n \stackrel{?}{=} B_n\}$ に型代入 Θ_1 を適用した結果を、

$$\Theta_1(E_1) \stackrel{\text{def}}{=} \{\Theta_1(A_1) \stackrel{?}{=} \Theta_1(B_1),\ \cdots,\ \Theta_1(A_n) \stackrel{?}{=} \Theta_1(B_n)\}$$

と定義する。また、型代入 $\Theta_0 = [\alpha_1 := A_1, \cdots, \alpha_n := A_n]$ に、型代入 $\Theta_1 = [\beta_1 := B_1, \cdots, \beta_m := B_m]$ を適用した結果を、

$$\Theta_1(\Theta_0) \stackrel{\text{def}}{=} [\alpha_1 := \Theta_1(A_1), \cdots, \alpha_n := \Theta_1(A_n), \beta_1 := B_1, \cdots, \beta_m := B_m]$$

と定義する。たとえば、$[\gamma := \texttt{int} \times \delta]$ を $[\alpha := \texttt{bool} \to \gamma, \beta := \delta]$ に適用すると、$[\alpha := \texttt{bool} \to (\texttt{int} \times \delta), \beta := \delta, \gamma := \texttt{int} \times \delta]$ を得る[*18]。

上記の単一化アルゴリズムは非決定的であり、E が 2 つ以上の等式を含む場合、どの等式から解消していくかは一意的ではない。しかし、実際には、どのような順番で等式を解消していっても、必ず停止し、かつ、その解は実質的に同じであることが証明されている。

例 23（単一化アルゴリズムの実行例 1） $E = \{\alpha \to \beta \stackrel{?}{=} \gamma \to \delta,\ \beta \stackrel{?}{=} \epsilon \times \texttt{int},\ \delta \stackrel{?}{=} \epsilon \times \alpha\}$ に対して、$\texttt{unify}(E, [\,])$ を実行する。(最初は、$\Theta = [\,]$ である。)

- E から $\alpha \to \beta \stackrel{?}{=} \gamma \to \delta$ を選ぶ。つまり、$A = \alpha \to \beta$, $B = \gamma \to \delta$ である。(2-5) に該当するので、次の unify の呼び出しでは、$E = \{\alpha \stackrel{?}{=} \gamma,\ \beta \stackrel{?}{=} \delta,\ \beta \stackrel{?}{=} \epsilon \times \texttt{int},\ \delta \stackrel{?}{=} \epsilon \times \alpha\}$ および $\Theta = [\,]$ となる。
- E から $\alpha \stackrel{?}{=} \gamma$ を選ぶと (2-2-2) に該当する。次の unify の呼び出しでは、$E = \{\beta \stackrel{?}{=} \delta,\ \beta \stackrel{?}{=} \epsilon \times \texttt{int},\ \delta \stackrel{?}{=} \epsilon \times \gamma\}$ および $\Theta = [\alpha := \gamma]$ となる。
- E から $\beta \stackrel{?}{=} \delta$ を選ぶと (2-2-2) に該当する。次の unify の呼び出しでは、$E = \{\delta \stackrel{?}{=} \epsilon \times \texttt{int},\ \delta \stackrel{?}{=} \epsilon \times \gamma\}$ および $\Theta = [\alpha := \gamma,\ \beta := \delta]$ となる。
- E から $\delta \stackrel{?}{=} \epsilon \times \texttt{int}$ を選ぶと (2-2-2) に該当する。次の unify の呼び出しでは、$E = \{\epsilon \times \texttt{int} \stackrel{?}{=} \epsilon \times \gamma\}$ および $\Theta = [\alpha := \gamma,\ \beta := \epsilon \times \texttt{int},\ \delta := \epsilon \times \texttt{int}]$ となる。
- E から $\epsilon \times \texttt{int} \stackrel{?}{=} \epsilon \times \gamma$ を選ぶと (2-4) に該当する。次の unify の呼び出しでは、$E = \{\epsilon \stackrel{?}{=} \epsilon,\ \texttt{int} \stackrel{?}{=} \gamma\}$ および $\Theta = [\alpha := \gamma,\ \beta := \epsilon \times \texttt{int},\ \delta := \epsilon \times \texttt{int}]$ となる。
- E から $\epsilon \stackrel{?}{=} \epsilon$ を選ぶと (2-1) に該当する。次の unify の呼び出しでは、$E = \{\texttt{int} \stackrel{?}{=} \gamma\}$ および $\Theta = [\alpha := \gamma,\ \beta := \epsilon \times \texttt{int},\ \delta := \epsilon \times \texttt{int}]$ となる。
- E から $\texttt{int} \stackrel{?}{=} \gamma$ を選ぶと (2-3) に該当する。次の unify の呼び出しでは、$E = \{\}$ および $\Theta = [\alpha := \texttt{int},\ \beta := \epsilon \times \texttt{int},\ \delta := \epsilon \times \texttt{int},\ \gamma := \texttt{int}]$ となる。
- $E = \{\ \}$ なので、$\Theta = [\alpha := \texttt{int},\ \beta := \epsilon \times \texttt{int},\ \delta := \epsilon \times \texttt{int},\ \gamma := \texttt{int}]$ を返す。

[*17] 変数が式に現れるかどうかのチェックを、occurs check と言う。

[*18] この定義は、Θ_0 と Θ_1 が、同じ型変数に代入しない場合のみ有効である。たとえば、$[\alpha := \texttt{int}]$ を $[\alpha := \texttt{bool}, \beta := \texttt{int}]$ に適用すると、上記の定義では、$[\alpha := \texttt{bool}, \beta := \texttt{int}, \alpha := \texttt{int}]$ となるが、これは、α が二度現れてしまい、型代入とならない。なお、unify のアルゴリズムでは、そのようなケースは出てこない。

例 24（単一化アルゴリズムの実行例 2） $E = \{\alpha \times \mathtt{bool} \stackrel{?}{=} \mathtt{int} \times \beta,\ \beta \stackrel{?}{=} \alpha \to \mathtt{int}\}$ に対して、$\mathtt{unify}(E, [\,])$ を実行する。最初は、$\Theta = [\,]$ である。

- E から $\alpha \times \mathtt{bool} \stackrel{?}{=} \mathtt{int} \times \beta$ を選ぶと (2-4) に該当し、次の unify の呼び出しでは、$E = \{\alpha \stackrel{?}{=} \mathtt{int},\ \mathtt{bool} \stackrel{?}{=} \beta,\ \beta \stackrel{?}{=} \alpha \to \mathtt{int}\}$ および $\Theta = [\,]$ となる。
- $\alpha \stackrel{?}{=} \mathtt{int}$ を選ぶと (2-2-2) に該当し、次の unify の呼び出しでは、$E = \{\mathtt{bool} \stackrel{?}{=} \beta,\ \beta \stackrel{?}{=} \mathtt{int} \to \mathtt{int}\}$ および $\Theta = [\alpha := \mathtt{int}]$ となる。
- $\mathtt{bool} \stackrel{?}{=} \beta$ を選ぶと (2-3) に該当し、次の unify の呼び出しでは、$E = \{\mathtt{bool} \stackrel{?}{=} \mathtt{int} \to \mathtt{int}\}$ および $\Theta = [\alpha := \mathtt{int},\ \beta := \mathtt{bool}]$ となる。
- $\mathtt{bool} \stackrel{?}{=} \mathtt{int} \to \mathtt{int}$ を選ぶと (2-6) に該当し、「解なし」が答となる。

例 25（型推論アルゴリズムの実行例） 型環境 $\Gamma = y : \alpha$ と、項 $M = \lambda f.\lambda x.\ f@(x + y)$ に対する型推論を行なう。

- ステップ 1 を実行する。これは、例 5 の通りであり、M の型を β と置いた上で、型制約 $E = \{\beta \stackrel{?}{=} \gamma \to \delta, \delta \stackrel{?}{=} \epsilon \to \phi, \rho \to \phi \stackrel{?}{=} \gamma, \rho \stackrel{?}{=} \mathtt{int}, \epsilon \stackrel{?}{=} \mathtt{int}, \alpha \stackrel{?}{=} \mathtt{int}\}$ が得られる。
- ステップ 2 を実行する。$\mathtt{unify}(E, [\,])$ の形で計算を行なうことにより、型代入 $[\alpha := \mathtt{int}, \beta := (\mathtt{int} \to \phi) \to (\mathtt{int} \to \phi), \delta := \mathtt{int} \to \phi, \gamma := \mathtt{int} \to \phi, \epsilon := \mathtt{int}, \rho := \mathtt{int}]$ が得られる。この型代入を Θ と置く。
- 型代入 Θ に対して、$\Theta(\Gamma) = y : \mathtt{int}$ および $\Theta(\beta) = (\mathtt{int} \to \phi) \to (\mathtt{int} \to \phi)$ なので、

  $$y : \mathtt{int} \vdash M : (\mathtt{int} \to \phi) \to (\mathtt{int} \to \phi)$$

  が代表的な型付けであることがわかった。

問題 1　以下の項の型推論を行いなさい。(型推論に失敗する例も含まれていることに注意せよ。)

- $\lambda x.\lambda y.\ \mathtt{if}\ x = 0\ \mathtt{then}\ y\ \mathtt{else}\ y + 1$
- $\lambda x.\ x@x$
- $\lambda x.\lambda y.\lambda z.\ (x@z)@(y@z)$
- $\lambda f.\lambda x.\ f@(f@x)$
- $\lambda x.\lambda y.\ (x@y) + (y@x)$
- $\lambda x.\ \mathtt{fix}\ f.y.\mathtt{if}\ y = 0\ \mathtt{then}\ x\ \mathtt{else}\ f@(y - 1) + 1$
- $\lambda x.\ \mathtt{fix}\ f.y.\mathtt{if}\ y = 0\ \mathtt{then}\ 1\ \mathtt{else}\ x * (f@(y - 1))$

この章の最後に、単一化問題と単一化アルゴリズムに関する定理を証明なしで紹介する。

定理 14（単一化問題の決定可能性） 1. 本資料の単純型付きラムダ計算の体系、計算体系 CoreML、および、後述する多相型の体系に対して、型制約の単一化問題は決定可能である。(それを有限時間で判定するアルゴリズムが存在する。)

2. 型制約 E の単一化問題の解が存在すれば、E の代表的な解が存在する。ただし、代表的な解とは、適当な型代入を適用することによって、他のどんな解も得られるものである。

定理 15（単一化アルゴリズムの正しさ） 任意の型制約 E に対して、上記の unify は、必ず有限時間で停止し、もし、E の単一化問題の解が存在すれば、その代表的な解 (の 1 つ) を返し、解が存在しなければ「解な

し」という答を返す。

前に述べた型制約の生成とあわせると、以下の性質が得られる。

定理 16（型推論アルゴリズムの正しさ） 本章で述べたアルゴリズムは、計算体系 CoreML の型推論問題に関して健全かつ完全である。

すなわち、Γ と M が与えられたとき、本章で述べたアルゴリズムは必ず停止し、型付け Θ と A が存在するならば、その代表的な型付けを返す。また、型付け Θ と A が存在するならば、型付けが存在しない旨を返す。