# 特徴制約系による第一級メッセージの型推論

Type Inference for First-Class Messages with Feature Constraints

Martin Müller
Universität des Saarlandes

西村 進
Susumu Nishimura
京都大学数理解析研究所
RIMS, Kyoto University

**概 要**

オブジェクト指向プログラミング言語における第一級メッセージの型推論を，特徴制約系 OF の充足可能性問題に還元し，その計算の複雑さを解析することによって，型推論の計算量が多項式時間であることを示す．また，OF を Treinen の特徴制約系 EF と比較することにより，OF に論理否定を追加した制約系の充足可能性問題は NP 完全であることも示す．この NP 完全性に関する考察から，型推論が多項式時間でおさまる理由を明らかにする．

## 1 はじめに: 第一級メッセージの型付け

第一級 (first-class) メッセージとは，関数型言語における第一級関数と同様に，計算の対象となるデータとして扱えるメッセージのことをいう．オブジェクト指向言語において，第一級メッセージは有用性が高い．例えば，関数型言語における map 関数に似たメソッド **map**は次のように定義できる．

method **map**(o,l) = *for each* m *in* l: o←m

ここで，o はオブジェクト，l は第一級メッセージのリストで，o←m は m に束縛された第一級メッセージによるオブジェクト o のメッソド起動をおこなう．

また，第一級メッセージは，分散オブジェクト指向プログラミングでよくみられる．例えば，簡単な proxy サーバは第一級メッセージを使えば次のように自然に定義することができる．

```
let ProxyServer =
      { new(o) = { send(m) = o←m} };
```

オブジェクト ProxyServer は，オブジェクト o をメソッド呼び出し **new**によって受け取り，結果として代理オブジェクトを返す．代理オブジェクトはメソッド **send**によってメッセージ m を受け取ると，そのメッセージをオブジェクト o に投げて，オブジェクト o のメソッドを起動する．

このような第一級メッセージのための型システムを，西村は ML 型システムの拡張として提案した．[3] このシステムにおける型付けを例によってみてみよう．

let o = { **pos**(x) = x>0, **neg**(p) = ¬ p};

このプログラムは，オブジェクト o を定義している．オブジェクト o は 2 つのメソッド **pos**と **neg**を持つ．これらは，それぞれ整数型 int と真偽値型 bool をうけとり，bool を返す．したがって，オブジェクト o の型は，これら 2 つのメソッドの型の積であるオブジェクト型で表される．

obj(**pos**:int → bool, **neg**:bool → bool)

次にメッセージであるが，第一級メッセージの持つメソッドの名前は，一般的には静的に決定することはできない．例えば，

let m = if b then **pos**(42) else **neg**(true);

はメッセージ m を定義しているが，その値は **pos**(42) の **neg**(true) いずれかに，真偽値 b に依存して決まる．したがって，メッセージの型はこれら

二つの可能性の和であるメッセージ型で表される．

$$\mathsf{msg}(\textbf{pos}{:}\mathsf{int}, \textbf{neg}{:}\mathsf{bool})$$

このように型が与えられたとき，メソッド呼び出し $\mathsf{o} \leftarrow \mathsf{m}$ の型は次のように決定される．

1. メッセージ m が持つであろうメソッド名 **pos** と **neg** の両方を，オブジェクト o は定義しており，しかも引数の型がそれぞれ一致している．
2. 2 つのメソッド **pos** と **neg** の両方ともがおなじ型 (bool) を返す．したがって，$\mathsf{o} \leftarrow \mathsf{m}$ は bool であると静的に決定できる．

これが，西村の型システムにおける第一級メッセージに対する型付けの条件である．

西村の型システムの提案においては，その定式化はかなり複雑であり，その型推論に対する解析を行うのは困難であった．本稿では，この型システムを制約系によって定式化し直し，型推論を制約の充足可能性問題に還元することによって，型推論の計算量が多項式時間であることを明らかにする．また，この型システムが通常より多くのプログラムを型付け可能にすることが，型推論が多項式時間であることに深く関係していることも示す．

## 2 特徴木と型の表現

上記型システムを制約によって定式化するために，新しい特徴制約系 OF を定義する．OF は基本的な特徴制約系 [1, 5] を，第一級メッセージによるメソッド呼び出しに対応する新しい制約で拡張したものである．

特徴制約系は，特徴木 **(feature tree)** と呼ばれる木構造に関する制約を記述するシステムである．特徴木とは，各頂点と各辺がそれぞれ名前付けされた木のことをいい，辺につけられた名前のことを特徴 **(feature)**，頂点につけられた名前のことをラベル **(label)**，という．ただし，一つの頂点から出る辺の特徴は互いに異なるとする．

本稿では，型を特徴木で表現する．整数型 int などの基礎型は，その型の名前でラベル付けされた 1 個の頂点のみからなる特徴木で表される．メッセージ型 $\mathsf{msg}(f_1{:}\tau_1, \ldots, f_n{:}\tau_n)$ は，msg でラベル付けされた頂点から，特徴 $f_1, \ldots, f_n$ で名前付けされた辺の先に，それぞれ $\tau_1, \ldots, \tau_n$ に対応する部分木を持つ特徴木で表され，またオブジェクト型 $\mathsf{obj}(f_1{:}\tau_1 \to \tau'_1, \ldots, f_n{:}\tau_n \to \tau'_n)$ は，obj でラベル付けされた頂点から，特徴 $f_1, \ldots, f_n$ で名前付けされた辺の先に，それぞれ $\tau_1 \to \tau'_1, \ldots, \tau_n \to \tau'_n$ に対応する部分木を持つ特徴木で表される．ただし，略記法 $\tau \to \tau'$ は，$\to$ でラベル付けされた頂点から，特別な 2 つの特徴 $d$ と $r$ で名前付けられた 2 つの辺の先に，$\tau$ および $\tau'$ に対応する部分木をそれぞれ持つような特徴木を表す．例として，図 1 に，第 1 節でみたオブジェクト型とメッセージ型の特徴木による表現を示す．

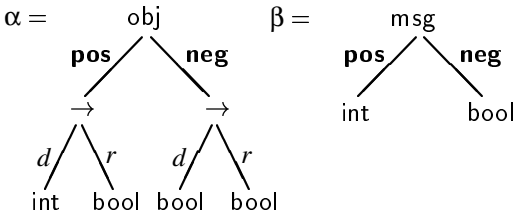


図 1: 特徴木による型の表現

なお，これ以後，型とその特徴木による表現を同一視することとし，$\mathsf{label}(\tau)$ は特徴木 $\tau$ の根の頂点に名前付けらたラベルを，$\mathsf{ar}(\tau)$ は $\tau$ の根から出ていく辺につけられた特徴の集合を，$\tau.f$ は特徴 $f$ を持つ辺の部分木を，それぞれ表すものとする．

## 3 特徴制約系 OF

特徴制約系 OF における式の文法は以下のように与えられる．

$$\begin{array}{rcll} \varphi & ::= & x = y & \text{(Equality)} \\ & | & a(x) & \text{(Labeling)} \\ & | & x[f]y & \text{(Selection)} \\ & | & F(x) & \text{(Arity Bound)} \\ & | & x\langle y\rangle z & \text{(Object Selection)} \\ & | & \varphi \wedge \varphi' & \text{(Conjunction)} \end{array}$$

ただし，$a$ はラベル，$f$ は特徴，$F$ は特徴の有限集合を表す．

OF 制約式の解釈を次のように与える．

$$\begin{array}{rcl} a(\tau) & \text{iff} & \mathsf{label}(\tau) = a \\ \tau[f]\tau' & \text{iff} & \tau.f = \tau' \\ F(\tau) & \text{iff} & \mathsf{ar}(\tau) \subseteq F \\ \tau\langle\tau'\rangle\tau'' & \text{iff} & \text{全ての } f \in \mathsf{ar}(\tau') \text{ に対して} \\ & & f \in \mathsf{ar}(\tau) \text{ かつ } \tau.f = \tau'.f \to \tau'' \end{array}$$

この解釈からわかるように，制約 $x\langle y\rangle z$ は，第一級メッセージによるメソッド呼び出しの型付けのために導入されたものである．例えば，$x, y$ をそれぞれ図 1 における特徴木 $\alpha, \beta$ とせよ．このとき，制約

$$
\begin{array}{lcll}
I(x,\Gamma,b) & = & a(x)\wedge\{\}(x) & \text{if } a=\mathsf{typeof}(b)\\
I(x,\Gamma,\mathsf{y}) & = & x=\Gamma(\mathsf{y}) & \\
I(x,\Gamma,f(M)) & = & \exists y\,(\mathsf{msg}(x)\wedge x[f]y\wedge I(y,\Gamma,M)) & \\
I(x,\Gamma,\{f_1(\mathsf{x}_1)=M_1,\ldots,f_n(\mathsf{x}_n)=M_n\}) & & & \\
\quad = & \mathsf{obj}(x)\wedge\{f_1,\ldots,f_n\}(x)\wedge\bigwedge_{i=1}^{n}\exists x_i\,\exists x'\,\exists z\,(x[f_i]x'\wedge x'=x_i\to z\wedge I(z,\Gamma;\mathsf{x}_i:x_i,M_i)) & & \\
I(x,\Gamma,M\leftarrow N) & = & \exists y\,\exists z\,(y\langle z\rangle x\wedge\mathsf{obj}(y)\wedge I(y,\Gamma,M)\wedge\mathsf{msg}(z)\wedge I(z,\Gamma,N)) & \\
I(x,\mathsf{let\ y}=M\ \mathsf{in}\ N) & = & \exists y\,(I(y,\Gamma;\mathsf{y}:y,M)\wedge I(x,\Gamma;\mathsf{y}:y,N)) &
\end{array}
$$

図 2: OF 制約による第一級メッセージの (単相) 型推論

$x\langle y\rangle z$ がみたされるのは，$z$ がラベル bool で名前付けられた一つの頂点のみからなる特徴木のとき，すなわち型 bool を持つときである．

$\Phi,\Phi'$ を，OF の式から通常の 1 階述語論理の結合子と qunatifier を使って得られる式とする．$\Phi$ が特徴木上での解釈の下で充足可能 (妥当) なとき，$\Phi$ は充足可能 (妥当) であるという．式 $\Phi\to\Phi'$ が妥当なとき，$\Phi$ は $\Phi'$ を伴意 (entail) するといい，$\Phi\models_{\mathrm{OF}}\Phi'$ と書く．

OF の 2 つの選択制約 $x[f]y$ と $x\langle y\rangle z$ の重要な違いは，定数特徴による選択 $x[f]y$ が関数的に働くのに対して，第一級特徴木による選択 $x\langle y\rangle z$ はそうでないことである．すなわち，

$$\models_{\mathrm{OF}}\quad x[f]y\wedge x[f]y'\ \to\ y=y' \qquad (1)$$

であるが一方，

$$\not\models_{\mathrm{OF}}\quad x\langle y\rangle z\wedge x\langle y\rangle z'\ \to\ z=z' \qquad (2)$$

である．なぜなら，式 (2) においては，特徴木 $y$ が部分木を持たない場合，$x\langle y\rangle z$ は $z$ に関しては何の制約も課さないからである．しかし，次のように，特徴木 $y$ が少なくとも一つ特徴を持てば関数的に働くことがわかる．

$$\models_{\mathrm{OF}}\quad y[f]y'\wedge x\langle y\rangle z\wedge x\langle y\rangle z'\ \to\ z=z' \qquad (3)$$

このことは，型推論の計算量が多項式時間であることと深く関係している．

定理 **3.1** *OF* 制約式の充足可能性問題は，多項式時間および多項式領域で決定可能．

証明は，制約解消を書き換え規則として与え，書き換えが多項式ステップで終了することを示すことでできる ( 詳しくは，[2] 参照 .)

## 4 型推論

第 1 節にみたような，第一級メッセージの型付けの条件を，OF の制約式として表すことによって，型推論を定式化する．ここでは，次のような簡単なオブジェクト指向言語の型推論を考える．

$$
\begin{array}{lll}
M ::= & b & \text{(Constant)}\\
\mid & \mathsf{x} & \text{(Variable)}\\
\mid & f(M) & \text{(Message)}\\
\mid & \{f_1(\mathsf{x}_1)=M_1,\ldots,f_n(\mathsf{x}_n)=M_n\} & \text{(Object)}\\
\mid & M\leftarrow N & \text{(Message Passing)}\\
\mid & \mathsf{let\ y}=M\ \mathsf{in}\ N & \text{(Let Binding)}
\end{array}
$$

図 2 に，型推論アルゴリズム $I$ を示す．このアルゴリズムは，プログラムが充たすべき型付け条件を OF 制約式として生成する．実際の型付けは，その制約を解くことによって得られる．

型環境 ($\Gamma$ で表す) を，プログラム中の自由変数に型を表す制約変数を割り当てる関数とする．アルゴリズム $I$ は，変数 $x$，型環境 $\Gamma$，プログラム $M$ を入力として，OF 制約式を生成する．ここで，変数 $x$ はプログラム $M$ の型を表し，したがって，生成される制約 $I(x,\Gamma,M)$ のいかなる解の $x$ の値も，型環境 $\Gamma$ の下での $M$ の正しい型となっている．なお，$\Gamma;\mathsf{x}:x$ は，プログラム変数 x を制約変数 $x$ に割り当てるような型環境 $\Gamma$ の拡張であり，typeof は定数をそれに対応する基礎型に割り当てる関数である．

明らかに，$I$ が生成する制約の大きさはプログラムの大きさに比例するから，OF の充足可能性問題の計算量から次の定理が得られる．

定理 **4.1** 第一級メッセージの型推論は多項式計算量および領域量を持つ．

なお，ここで示した型推論は単相型のものであるが，Hindley/Milner 流の制約多相型システムの一般的な枠組みである HM(X) [4] に，OF 制約系を当てはめることによって多相型システムに拡張することができる．

## 5 空メッセージ型と型推論の NP 完全性

OF に類似した特徴制約系として Treinen の EF [6] がある．EF はおおよそ，OF から制約 $x\langle y\rangle z$ を除き，かわりに $u = f$ と $x[u]y$ の 2 つの制約を加えたものに等しい．ここで，$u$ は特徴の上を動く特徴変数であり，$u = f$ は $u$ が特徴 $f$ を表すことを，$x[u]y$ は定数特徴による選択 $x[f]y$ を，特徴変数による選択に拡張したものである．

Treinen は，EF の充足可能性問題が NP 完全であることを示した．NP 完全性を示す鍵となるのが次の式が EF では妥当であるという事実である．

$$\models_{\mathrm{EF}} \quad \{f_1, \ldots, f_n\}x \wedge x[u]y \;\rightarrow\; \bigvee_{i=1}^{n} u = f_i$$

OF でこれと同等のことを表現しようとすると，制約 $x\langle y\rangle z$ は $y$ に関して関数的でなかった ( 3 節，式 (2) ) から，次のように $\neg\{\}(y)$ の形の論理否定を用いる必要がある．

$$\models_{\mathrm{OF}} \{f_1, \ldots, f_n\}(x) \wedge x\langle y\rangle z \wedge \neg\{\}(y) \;\rightarrow\; \bigvee_{i=1}^{n} \exists z_i\, y[f_i]z_i$$

以上の観察から，次の命題を示すことができる．

**命題 5.1** *OF* に $\neg\{\}(x)$ の形の制約を加えた制約系の充足性問題は *NP* 完全である .( [2] 参照 .)

この命題が型推論に関して意味するところは何であろうか．まず，特徴制約系 OF で重要な式 (2) がプログラムの型付けにおいて意味するところは，メソッド呼び出し $M \leftarrow N$ において，メッセージ $N$ の型が空メッセージ型 msg() を持っていたとしても，型付け可能であるということである．空メッセージ型はメッセージの名前や引数の型などの情報をまったく持たない．これは，空メッセージ型はプログラム中，文法的にはメッセージとして使われているものの，実行時にはまったく使われないような式に与えられる型であることを意味する．

空メッセージ型のこの性質は，通常の型システムではみられない現象を引き起こす．例えば，次のようなプログラムを考える．

```
let o1 = {a(x)=x+1} in
let o2 = {c(x)=x*2} in
let o3 = {foo(m)= begin o1←m; o2←m end}
```

明らかに，メソッド **foo**の型安全な実行は存在しない．なぜなら，オブジェクト o1 と o2 に共通するメソッド名がないからである．しかし，このプログラムは我々のシステムでは型付け可能である．なぜなら，対応する OF 制約

$$o_1 = \mathsf{obj}(\mathbf{a} : \mathsf{int}\rightarrow\mathsf{int}) \wedge o_2 = \mathsf{obj}(\mathbf{c} : \mathsf{int}\rightarrow\mathsf{int}) \wedge$$
$$o_3 = \mathsf{obj}(\mathbf{foo} : m\rightarrow z) \wedge o_1\langle m\rangle z_1 \wedge o_2\langle m\rangle z_2$$

は $m$ を空メッセージ型とすることによって充足可能だからである．

このように，実際には型安全な実行がないにも関わらず，型付け可能なプログラムが存在するというのは，型システムとして不適切であるように見えるかもしれない．しかし，我々の型システムは全ての型付け可能なプログラムの実行は型安全であるという点で型システムとしての重要な要件である健全性を満たしている．上の例でも，メッセージ m は絶対にプログラム中で使用されないので，型安全性は保たれている．

この型システムで最も重要なのは，空メッセージ型を許すことで，型推論を多項式時間内におさめることができることである．命題 5.1 より，もし空メッセージ型を禁止すれば型推論は NP 完全となる．すなわち，空メッセージ型は，型安全な実行が存在するか否かのチェックを省略する代りに，第一級メッセージの型推論を多項式時間でおこなうことを可能にしている．本研究の結果から，空メッセージ型の禁止と多項式時間の型推論は両立しないと考えられる。

## 参考文献


[1] R. Backofen and G. Smolka. A complete and recursive feature theory. *Theoretical Computer Science*, 146(1–2):243–268, 1995.

[2] M. Müller and S. Nishimura. Type inference for first-class messages with feature constraints. *ASIAN98 Computing Science Conference*, LNCS 1538, pp. 169–187, 1998.

[3] S. Nishimura. Static typing for dynamic messages. *ACM Symposium on Principles of Programming Languages*, pp. 266–278. ACM Press, 1998.

[4] M. Odersky, M. Sulzmann, and M. Wehr. Type inference with constrained types. *Theory and Practice of Object Systems*, 5(1):35–55, 1999.

[5] G. Smolka and R. Treinen. Records for logic programming. *The Journal of Logic Programming*, 18(3):229–258, Apr. 1994.

[6] R. Treinen. Feature constraints with first-class features. *International Symposium on Mathematical Foundations of Computer Science*, LNCS 711, pp. 734–743. Springer-Verlag, 1993.