型代入を遅延する最適化型推論アルゴリズム*

上野 雄大† 大堀 淳††
† 北陸先端科学技術大学院大学情報科学研究科
†† 東北大学電気通信研究所
katsuu@jaist.ac.jp
ohori@riec.tohoku.ac.jp

型推論は ML 系型付き言語のコンパイルステップの中でも時間を要する処理であるにもかかわらず，その最適化方式や最適化による効果などはあまり研究されていない．本稿では，従来の型推論アルゴリズム $\mathcal{W}$ と比較して実用上より効率的であると期待でき，かつ宣言的に定義される新たな型推論アルゴリズム $\mathcal{DW}$ を提案し，その正しさを示すとともに，そのアルゴリズムを実用的なコンパイラへ採用する上での課題に関して議論する．

1 はじめに

Standard ML[8] や Haskell[3]，ObjectiveCaml[6] など近代的な関数型言語の大きな特徴の一つに，多相型型推論機構がある．これは，プログラムが持つ最も一般的な型を自動的に推論する機構である．この機構の中核をなすのが，構文解析によって得られた拡張ラムダ式に対する型推論アルゴリズムである．多相型型推論アルゴリズムの対象となる拡張ラムダ式は，通常のラムダ式に加えて多相型を持つ変数束縛を許す多相型 let 構文を含む．この多相型 let 構文の存在によって，関数型言語の型推論問題は DEXPTIME 完全であることが示されており [5, 4]，漸近的な動作を考えた場合，アルゴリズム論の意味において効率的な型推論アルゴリズムの構築は不可能である．しかし，このような論理的な限界が存在する一方，型推論アルゴリズムはコンパイラフロントエンドの中で最も複雑かつ時間がかかる処理であり，その宣言的でかつ実用上より効率的なアルゴリズムの開発は，型推論機構を装備した次世代プログラミング言語の設計と実装にとって重要な課題である．

この重要性にもかかわらず，型推論アルゴリズムの最適化の研究はほとんどなされておらず，多くのコンパイラで現在採用されている型推論アルゴリズムは，Milner の先駆的な研究 [7] によって与えられたアルゴリズムにアドホックな改良を加えたものとなっている．

本研究の一般的な目的は，高度な機能を含む最先端の関数型プログラミング言語のコンパイラの高信頼かつ効率的な実装の基礎として，実用上，より効率的で宣言的記述が可能な型推論アルゴリズムとその実装技術を構築することである．本稿では，その第一歩として新しい型推論アルゴリズムを提案し，その正しさを示し，その拡張可能性や実用的なコンパイラへ応用する上での課題などを議論する．現在，本稿で提案したアルゴリズムの実装や，その性能の定量的な評価，さらにそれらをもとにしたアルゴリズムの改良の研究を行っている．近い将来，これらを含むより完成された研究成果の報告を行う予定である．

次章以降の本稿の構成は以下の通りである．第 2 章では，従来のアルゴリズムの問題点と，より効率的なアルゴリズムを構築する上での我々の基本的なアイデアを概説する．第 3 章では，型推論アルゴリズムを与え，その型健全性を証明する．最後に，第 4 章で種々の拡張の可能性や実用化の上での課題を含む今後の課題に関して議論する．

2 型代入を遅延する型推論アルゴリズム

議論を明確にするために，本稿で分析の対象とする型システムを定義する．以下の文法で定義されるラムダ式の集合を考える．

$$e ::= x \mid \lambda x.e \mid e\ e \mid \mathsf{let}\ x = e\ \mathsf{in}\ e$$

ここで，$\mathsf{let}\ x = e\ \mathsf{in}\ e$ は多相型 let 構文である．単相型（τ）と多相型（σ）の集合は以下の文法で定義

*本研究の一部は，科学研究費補助金特定領域研究（課題番号：16016240），基盤研究（B）（課題番号：15300006），および文部科学省委託研究 e-Society 基盤ソフトウェアの総合開発「プログラム自動解析に基づく高信頼ソフトウエアシステムの構築技術」の補助の下に行われたものである．

$$\text{(const)}\quad \Gamma \rhd c^b : b$$

$$\text{(var)}\quad \Gamma\{x:\sigma\} \rhd x : \tau \qquad (\tau < \sigma)$$

$$\text{(abs)}\quad \frac{\Gamma\{x:\tau_1\} \rhd e : \tau_2}{\Gamma \rhd \lambda x.e : \tau_1 \to \tau_2}$$

$$\text{(app)}\quad \frac{\Gamma \rhd e_1 : \tau_1 \to \tau_2 \qquad \Gamma \rhd e_2 : \tau_1}{\Gamma \rhd e_1\ e_2 : \tau_2}$$

$$\text{(let)}\quad \frac{\Gamma \rhd e_1 : \tau_1 \qquad \Gamma\{x:\forall \bar{t}.\tau\} \rhd e_2 : \tau_2}{\Gamma \rhd \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2} \quad (\bar{t} \cap \mathrm{FTV}(\Gamma) = \emptyset)$$

図 1: ML の型システム

される．

$$\tau ::= b \mid t \mid \tau \to \tau$$
$$\sigma ::= \tau \mid \forall \bar{t}.\tau$$

ここで，$\bar{t}$ は型変数の集合を表す．

型システムは，$\Gamma \rhd e : \sigma$ の形の型判定導出システムとして定義される．図 1 に型導出規則の集合を与える．

第 1 章で述べた通り，現在多くのコンパイラに実装されている型推論アルゴリズムは Milner の型推論アルゴリズム $\mathcal{W}$ を基礎とする．$\mathcal{W}$ は，環境 Γ と式 e を受け取り，型変数への代入 S と型 τ を返す関数として定義される．図 2 にその定義を示す．

このアルゴリズムの基本的な流れは以下のように整理することができる．

1. 再帰的に自分自身を部分式に適用し，型と型代入を計算する．
2. 得られた型代入 S を以前の環境 Γ に適用し，型環境に含まれる型情報を更新する．
3. 更新された型環境のもとで，他の部分式の型推論を実行する．

アルゴリズムの効率の点からこの過程を分析すると，ステップ 2 で繰り返し行われる型代入 S の型環境 Γ への適用は，Γ に S には無関係の環境が多数含まれる可能性を考えると無駄な処理が多く非効率的である恐れがある．

この事情は，関数適用の評価を，

$$(\lambda x_1 \ldots \lambda x_n.M)N_1 \cdots N_n \Rightarrow [N_n/x_n] \cdots [N_1/x_1]M$$

$$\mathcal{W}(\Gamma, x) = \text{if } x \notin dom(\Gamma) \text{ then } \mathit{fail} \text{ else } (\emptyset, \Gamma(x))$$

$$\mathcal{W}(\Gamma, \lambda x.e_1) =$$
$$\quad \text{let } (S_1, \tau_1) = \mathcal{W}(\Gamma\{x:t\}, e_1) \quad (t \text{ fresh})$$
$$\quad \text{in } (S_1, S_1(t) \to \tau_1)$$

$$\mathcal{W}(\Gamma, e_1\ e_2) = \text{let } (S_1, \tau_1) = \mathcal{W}(\Gamma, e_1)$$
$$(S_2, \tau_2) = \mathcal{W}(S_1(\Gamma), e_2)$$
$$S_3 = \mathcal{U}(\{(\tau_1, \tau_2 \to t)\})$$
$$\text{in } (S_3 \circ S_2 \circ S_1, S_3(t))$$

$$\mathcal{W}(\Gamma, \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2) =$$
$$\quad \text{let } (S_1, \tau_1) = \mathcal{W}(\Gamma, e_1)$$
$$\quad\quad \bar{t} = \mathrm{FTV}(\tau_1) \setminus \mathrm{FTV}(\Gamma)$$
$$\quad\quad (S_2, \tau_2) = \mathcal{W}(S_1(\Gamma)\{x:\forall \bar{t}.\tau_1\}, e_2)$$
$$\quad \text{in } (S_2 \circ S_1, \tau_1)$$

図 2: Milner の型推論アルゴリズム

のような代入を繰り返し行うことによって実現することに相当すると見なすことができる．関数型言語のコンパイラでは，このような実引数の代入を実行せずにその効果を環境として保存し，その環境の下で M を評価するという戦略をとることによって実行効率の良い実装を実現している．そこで，型推論においても，部分式に対して推論される型代入を現在の環境に適用した上で推論を実行するのではなく，型代入を型環境を評価する際に使用すべき明示的な環境として保存するようにアルゴリズムを再編成すれば，型代入を環境に繰り返し適用することを抑止でき，より効率の良いアルゴリズムが実現できると期待される．

$\mathcal{W}$ の非効率のもう一つの要因は，let 式の型推論において行われる $\mathrm{FTV}(\tau) \setminus \mathrm{FTV}(\Gamma)$ の計算である．Γ から到着可能な型変数の集合を常に正確に把握するならば，Γ 全体のスキャンを let 式の型推論の際に毎回実行するような無駄を省くことができるはずである．

そこで，本研究では，以上の洞察に基づき，型推論アルゴリズムの引数として型環境 Γ に加え，Γ に対する型代入環境 S と Γ から到着可能な型変数集合 Δ を導入し，これらの情報から新たな型代入 S'，新たに型環境から到着可能になった型変数の集合 Δ'，および式の型 τ を計算する

$$\mathcal{DW}(\Gamma, S, \Delta, e) = (S', \Delta', \tau)$$

という形のアルゴリズム $\mathcal{DW}$ を構築する．

$$\text{(u-i)}\quad (S, S', E \cup \{(\tau, \tau)\}) \Longrightarrow (S, S', E)$$
$$\text{(u-ii)}\quad (S, S', E \cup \{(\tau_{11} \to \tau_{12}, \tau_{21} \to \tau_{22})\}) \Longrightarrow (S, S', E \cup \{(\tau_{11}, \tau_{21}), (\tau_{12}, \tau_{22})\})$$
$$\text{(u-iii)}\quad (S, S', E \cup \{(t, \tau)\}) \Longrightarrow (S, S' \cup \{(t, S' \circ S(\tau))\}, E) \quad \text{if } t \notin \mathrm{dom}(S' \circ S),\ t \notin \mathrm{FTV}(S' \circ S(\tau))$$
$$\text{(u-iv)}\quad (S, S', E \cup \{(t, \tau)\}) \Longrightarrow (S, S', E \cup \{(S' \circ S(t), \tau)\}) \quad \text{if } t \in \mathrm{dom}(S' \circ S)$$

図 3: 単一化アルゴリズム $\mathcal{DU}$ の変形規則

3 $\mathcal{DW}$ の定義とその健全性

$\mathcal{DW}$ は，従来の $\mathcal{W}$ 同様，型の制約を満たす最も一般的な解を，型等式に対する最も一般的な単一化を計算することによって求めることで型推論問題を解決する．このために単一化アルゴリズムを必要とする．型代入を遅延するという $\mathcal{DW}$ の戦略を実現するためには，単一化アルゴリズムも型代入 S の下で型等式 E の単一化を計算するアルゴリズムへと洗練する必要がある．本節ではこのように洗練された単一化アルゴリズム $\mathcal{DU}$ とそれを用いた $\mathcal{DW}$ の 2 つのアルゴリズムの定義を与え，その正しさをそれぞれ証明する．

3.1 単一化アルゴリズム

単一化アルゴリズム $\mathcal{DU}$ は，[2] の考え方に従い，型等式間の変形関係 $\Longrightarrow$ を通じて以下のような関数として定義される．

$$\mathcal{DU}(S, E) = \begin{cases} S' & \text{if } (S, \emptyset, E) \overset{*}{\Longrightarrow} (S, S', \emptyset), \\ \mathit{failure} & \text{otherwise.} \end{cases}$$

ここで，S および S' は型代入，E は型等式の集合である．変形規則を図 3 に示す．各変形規則は，必要に応じて E に含まれる型変数を S で解決しながら，型等式 E の単一化を求めるステップを実現する．

型代入 S は，代入の対象となる型変数 t と，それに代入される型 τ のペア (t, τ) を要素とする集合で与えられる．以下では，S は型変数から型への関数，またはその定義域を任意の型構造に拡張した準同型写像とみなし，S に含まれる代入の対象となる型変数の集合を $\mathrm{dom}(S)$，任意の型構造 α に対して，α に含まれる自由な型変数を S によって対応付けられている型で置き換えた型構造 α' を求める操作を $S(\alpha) = \alpha'$ と書く．また，型代入 $S = \{(t_1, \tau_1), \ldots, (t_n, \tau_n)\}$ が以下の条件を全て満たすとき，S **は整形されている** *(well-formed)* という．

- $t_1, \ldots, t_n$ は相異なる．
- $t_1, \ldots, t_n$ は $\tau_1, \ldots, \tau_n$ に現れない．

この単一化アルゴリズム $\mathcal{DU}$ について，以下の定理が成り立つ．

定理 3.1. 任意の整形されている型代入 S と任意の型等式の集合 E について，もし $S(E)$ が単一化を持つならば，アルゴリズム $\mathcal{DU}$ は $S(E)$ の最も一般的な単一化 *(most general unifier)* を返す．$S(E)$ が単一化を持たないならば $\mathcal{DU}$ は失敗を報告する．

証明. まず，$(S, S'_1, E_1) \Longrightarrow (S, S'_2, E_2)$ でかつ S'_1 が整形されているならば，型代入 S'_2 も整形されていることを，変形の長さに関する帰納法で示す．

次に，各変形規則は単一化の集合を保存すること，すなわち，$(S, S'_1, E_1) \Longrightarrow (S, S'_2, E_2)$ ならば，任意の型代入 S_0 について，S_0 が $S'_1 \circ S(E_1) \cup S'_1$ の単一化であるとき, かつそのときに限り，S_0 は $S'_2 \circ S(E_2) \cup S'_2$ の単一化であることを示す．この性質は変形規則 (u-i) および (u-ii) については明らかであり，変形規則 (u-iii) については条件 $t \notin \mathrm{dom}(S' \circ S)$ から，(u-iv) については S が整形されていることから容易に示すことができる．この性質より，$\mathcal{DU}(S, E) = S'$ ならば，$(S, \emptyset, E) \overset{*}{\Longrightarrow} (S, S'_2, \emptyset)$ であるから S'_2 の最も一般的な単一化は $S(E)$ の最も一般的な単一化である．一方，S'_2 は整形されているから，S'_2 の最も一般的な単一化は S'_2 自身である．従って，$S(E_2)$ の最も一般的な単一化は S'_2 である．

一方，アルゴリズムが失敗を報告する場合を考える．$\mathcal{DU}(S, E) = \mathit{failure}$ と仮定する．このときアルゴリズムの定義から $(S, \emptyset, E) \Longrightarrow (S, S'_1, E_1) \not\Longrightarrow (S, S'_2, E_2)$ となる $E_1 \neq \emptyset$ が存在する．ところが，変形規則の定義より，$(S, S'_1, E_1) \not\Longrightarrow (S, S'_2, E_2)$ ならば $S'_1 \circ S(E_1) \cup S'_1$ は単一化を持たない．変形規則は単一化の集合を保存するから，$S(E)$ も単一化を持たない．

アルゴリズムの停止性については，(S, S', E) の複雑さの量を $S' \circ S(E)$ 中の型変数の数，E に含まれる $S' \circ S$ によって解決された型変数の数，および E に含まれる型の大きさの総和で構成される 3 つ組で表現するとき，各変形規則が複雑さの量を必ず減少

$$\mathcal{DW}(\Gamma, S, \Delta, c^b) = (\emptyset, \emptyset, b)$$

$$\mathcal{DW}(\Gamma\{x:\tau\}, S, \Delta, x) = (\emptyset, \emptyset, \tau)$$

$$\begin{aligned}\mathcal{DW}(\Gamma\{x:\forall(t_1,\ldots,t_n).\tau\}, S, \Delta, x) = \text{let } &\{t'_1,\ldots,t'_n\} = \mathit{newvars}(\mathrm{dom}(S)\cup\Delta, \{t_1,\ldots,t_n\})\\ &\tau_1 = [t'_1/t_1,\ldots,t'_n/t_n]S(\tau)\\ \text{in } &(\emptyset, \{t'_1,\ldots,t'_n\}, \tau_1)\end{aligned}$$

$$\begin{aligned}\mathcal{DW}(\Gamma, S, \Delta, \lambda x.e) = \text{let } &t = \mathit{newvar}(\mathrm{dom}(S)\cup\Delta)\\ &(S_1, \Delta_1, \tau_1) = \mathcal{DW}(\Gamma\{x:t\}, S, \Delta\cup\{t\}, e)\\ \text{in } &(S_1, \Delta_1 \cup (\{t\}\setminus\mathrm{dom}(S_1)), t\to\tau_1)\end{aligned}$$

$$\begin{aligned}\mathcal{DW}(\Gamma, S, \Delta, e_1\ e_2) = \text{let } &(S_1, \Delta_1, \tau_1) = \mathcal{DW}(\Gamma, S, \Delta, e_1)\\ &(S_2, \Delta_2, \tau_2) = \mathcal{DW}(\Gamma, S_1\circ S, (\Delta\setminus\mathrm{dom}(S_1))\cup\Delta_1, e_2)\\ &t = \mathit{newvar}(\mathrm{dom}(S_2\circ S_1\circ S)\cup(((\Delta\setminus\mathrm{dom}(S_1))\cup\Delta_1)\setminus\mathrm{dom}(S_2))\cup\Delta_2)\\ &S_3 = \mathcal{DU}(S_2\circ S_1\circ S, \{(\tau_1, \tau_2\to t)\})\\ \text{in } &(S_3\circ S_2\circ S_1, (\Delta_1\setminus\mathrm{dom}(S_3\circ S_2))\cup((\Delta_2\cup\{t\})\setminus\mathrm{dom}(S_3)), t)\end{aligned}$$

$$\begin{aligned}\mathcal{DW}(\Gamma, S, \Delta, \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2) = \text{let } &(S_1, \Delta_1, \tau_1) = \mathcal{DW}(\Gamma, S, \Delta, e_1)\\ &\sigma = \forall(((\Delta\setminus\mathrm{dom}(S_1))\cup\Delta_1)\setminus\mathrm{FTV}(S_1(\Delta))).\tau_1\\ &(S_2, \Delta_2, \tau_2) = \mathcal{DW}(\Gamma\{x:\sigma\}, S_1\circ S, (\Delta\setminus\mathrm{dom}(S_1))\cup\Delta_1, e_2)\\ \text{in } &(S_2\circ S_1, (\Delta_1\setminus\mathrm{dom}(S_2))\cup\Delta_2, \tau_2)\end{aligned}$$

図 4: 型推論アルゴリズム

させることによって示される． □

さらに，この単一化アルゴリズムに関して，その構造から以下の性質を証明することができる．これらの補題は型推論アルゴリズムの健全性の証明に必要となる．

補題 3.2. 任意の整形されている型代入 S，任意の型等式集合 E について，$\mathcal{DU}(S, E) = S'$ ならば，$S'\circ S$ も整形されている．

補題 3.3. 任意の型代入 S，任意の型等式集合 E について，$\mathcal{DU}(S, E) = S'$ ならば，$\mathrm{dom}(S)\cap\mathrm{dom}(S') = \emptyset$ である．

補題 3.4. S を任意の整形されている型代入，E を任意の型等式集合 E とする．もし $\mathcal{DU}(S, E) = S'$ ならば，任意の型 τ について $\mathrm{FTV}(S'(\tau)) \subseteq (\mathrm{FTV}(\tau)\cup\mathrm{FTV}(S(E)))\setminus\mathrm{dom}(S')$ である．

3.2 型推論アルゴリズム

型推論アルゴリズム $\mathcal{DW}$ は以下のような形の関数として定義される．

$$\mathcal{DW}(\Gamma, S, \Delta, e) = (S', \Delta', \tau)$$

$\mathcal{DW}$ は，型環境 Γ と型環境に対する型代入 S，S の下で Γ から到達可能な型変数を全て含む型変数の集合 Δ，およびラムダ式 e を受けとり，新たな型代入 S'，S' と S によって解決されていない新たな型変数の集合 Δ'，および S'，S，Γ の下でのラムダ式 e の型 τ を返す．型推論アルゴリズム $\mathcal{DW}$ を図 4 に示す．また，$\mathcal{DW}$ の定義で用いている補助関数を図 5 に与える．

以下では，アルゴリズム $\mathcal{DW}$ の健全性を証明する．まず，型代入に関する以下の補題を導入する．

補題 3.5. 任意の型環境 Γ，任意の式 e，任意の型 τ，任意の型代入 S について，$\Gamma\triangleright e:\tau$ ならば $S(\Gamma)\triangleright e: S(\tau)$ である．

補題 3.6. 任意の型代入 S，任意の型変数の集合 Δ_1，Δ_2 について，$\Delta_1\subseteq\Delta_2$ ならば $\mathrm{FTV}(S(\Delta_1))\subseteq\mathrm{FTV}(S(\Delta_2))$ である．

それぞれの補題の証明は省略する．

定理 3.7 ($\mathcal{DW}$ の健全性). S を任意の整形されている置換，Γ を任意の型環境，Δ を任意の型変数の集合，e を任意のラムダ式とする．もし $\mathrm{FTV}(S(\Gamma))\subseteq\Delta$，$\mathrm{dom}(S)\cap\Delta = \emptyset$，かつ $\mathcal{DW}(\Gamma, S, \Delta, e) = (S', \Delta', \tau)$ ならば，$S'\circ S(\Gamma)\triangleright e: S'\circ S(\tau)$ である．

$$
\begin{aligned}
&newvar(\Delta) = x \quad \text{such that } x \notin \Delta\\
&newvars(\Delta, \{t_1, \ldots, t_n\}) =\\
&\quad \text{let } t'_1 = newvar(\Delta)\\
&\qquad t'_2 = newvar(\Delta \cup \{t'_1\})\\
&\qquad \cdots\\
&\qquad t'_n = newvar(\Delta \cup \{t'_1, \ldots, t'_{n-1}\})\\
&\quad \text{in } \{t'_1, t'_2, \ldots, t'_n\}
\end{aligned}
$$

図 5: 型推論アルゴリズムの補助関数

証明. 任意の整形されている置換 S，任意の型環境 Γ，任意の型変数の集合 Δ，任意のラムダ式 e について，$\mathrm{FTV}(S(\Gamma)) \subseteq \Delta$，$\mathrm{dom}(S) \cap \Delta = \emptyset$，かつ $\mathcal{DW}(\Gamma, S, \Delta, e) = (S', \Delta', \tau)$ ならば，以下の性質が全て成り立つことを示す.

1. $S' \circ S(\Gamma) \triangleright e : S' \circ S(\tau)$,
2. $\mathrm{FTV}(S' \circ S(\tau)) \subseteq (\Delta \setminus \mathrm{dom}(S')) \cup \Delta'$,
3. $\mathrm{FTV}(S'(\Delta)) \subseteq (\Delta \setminus \mathrm{dom}(S')) \cup \Delta'$,
4. $\mathrm{dom}(S') \cap \mathrm{dom}(S) = \emptyset$,
5. $\mathrm{dom}(S') \cap \Delta' = \emptyset$,
6. $\mathrm{dom}(S) \cap \Delta' = \emptyset$,
7. $\Delta \cap \Delta' = \emptyset$,
8. $S' \circ S$ は整形されている.

証明は，e の構造に関する帰納法による.

全ての場合について，前提条件，帰納法の仮定，単一化アルゴリズム $\mathcal{DU}$ に関する定理と補題，型代入に関する補題，および各補助関数の定義から，以上の性質を全て満たすことを証明することができる．ここでは，多相型が関わる場合，すなわち $\Gamma = \Gamma'\{x : \forall \bar{t}.\tau\}$ であるときの式 x の型推論と，let 式の型推論を行う場合についての略証を示す.

$\Gamma = \Gamma'\{x : \forall \bar{t}.\tau\}$，$e = x$ の場合．アルゴリズムの定義より明らかに τ_1 は $\forall \bar{t}.S(\tau)$ の例である．一方，$newvars$ の定義より $\{t'_1, \ldots, t'_n\} \cap \mathrm{dom}(S) = \emptyset$ であるから $\tau_1 = S(\tau_1)$，よって $S(\tau_1)$ は $\forall \bar{t}.S(\tau)$ の例である．従って，補題 3.5 および型システムの定義より $S' \circ S(\Gamma'\{\forall \bar{t}.\tau\}) \rhd x : S' \circ S(\tau_1)$.

$e = \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$ の場合．まず, σ が型システムのルール (let) に定められている条件を満たすことを示す．帰納法の仮定より $\Delta \cap \Delta_1 \cap \Delta_2 = \emptyset$ であるから，$\mathrm{FTV}(S_2 \circ S_1 \circ S(\Gamma)) \subseteq \mathrm{FTV}(S_1(\Delta)) \cup \Delta_2$ を示せばよい.

前提，帰納法の仮定，および補題 3.6 より $\mathrm{FTV}(S_2 \circ S_1 \circ S(\Gamma)) \subseteq \mathrm{FTV}(S_2 \circ S_1(\Delta)) \subseteq \mathrm{FTV}(S_2((\Delta \setminus \mathrm{dom}(S_1)) \cup \Delta_1)) \subseteq ((\Delta \setminus \mathrm{dom}(S_1)) \setminus \mathrm{dom}(S_2)) \cup \Delta_2$．ここで，補題 3.6 より $\Delta \setminus \mathrm{dom}(S_2 \circ S_1) = \mathrm{FTV}(S_1(\Delta \setminus \mathrm{dom}(S_2 \circ S_1))) \subseteq \mathrm{FTV}(S_1(\Delta))$ であるから，$\mathrm{FTV}(S_2 \circ S_1 \circ S(\Gamma)) \subseteq \mathrm{FTV}(S_1(\Delta)) \cup (\Delta_1 \setminus \mathrm{dom}(S_2)) \cup \Delta_2$.

一方，前提および帰納法の仮定より $\mathrm{FTV}(S(\Gamma)) \cap \Delta_1 = \emptyset$ かつ $\mathrm{dom}(S_2 \circ S_1) \cap \Delta_1 = \emptyset$．従って補題 3.6 より $\mathrm{FTV}(S_2 \circ S_1 \circ S(\Gamma)) \cap \Delta_1 = \emptyset$．ここで，$\Delta_1 \setminus \mathrm{dom}(S_2) \subseteq \Delta_1$ であるから $\mathrm{FTV}(S_2 \circ S_1 \circ S(\Gamma)) \cap S_2(\Delta_1 \setminus \mathrm{dom}(S_2)) = \mathrm{FTV}(S_2 \circ S_1 \circ S(\Gamma)) \cap (\Delta_1 \setminus \mathrm{dom}(S_2)) = \emptyset$.

以上より，$\mathrm{FTV}(S_2 \circ S_1 \circ S(\Gamma)) \subseteq \mathrm{FTV}(S_1(\Delta)) \cup \Delta_2$．よって，$\sigma$ が型システム Λ のルール (let) の条件を満たすことが示された．従って，帰納法の仮定，補題 3.5，および型システムの定義より $S_2 \circ S_1 \circ S(\Gamma) \triangleright \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : S_2 \circ S_1 \circ S(\tau_2)$ である． □

4 アルゴリズムの評価と拡張

第 1 章で指摘した通り，let 式を含む型推論問題の複雑さは DEXPTIME 完全であることが示されており，アルゴリズム論的により高速な型推論アルゴリズムは構成できない．以上で構築したアルゴリズム $\mathcal{DW}$ に対しても，[4] の結果などを参考に，指数関数的な時間を要する例を容易に構築することができる．しかし，実用的な観点からは，アルゴリズム $\mathcal{DW}$ は型代入を遅延することで大きな型項や型環境を頻繁に操作することを避けているため，多くの場合，従来の型推論アルゴリズムと比較して実用上はより高速であると期待できる．例えば，図 6 に示す例の場合[1]，従来のアルゴリズムでは型代入の適用がおよそ n^2 に比例する量に対して行われると見込まれるのに対して，我々のアルゴリズムでは，n に比例する量にとどまり，格段の高速化が見込める.

しかしながら，アルゴリズム $\mathcal{DW}$ の実用上の優位性の議論は，多くの典型的なプログラムに対する型推論に要する時間の測定や，測定結果を元にした従来の型推論アルゴリズムとの比較に基づく分析が必要である．さらに，現実の実用コンパイラにおいては，型推論アルゴリズムは明示的な型情報の構築やコンパイラが管理する種々の環境との複雑な相互作用を行っているため，比較評価のためのデータは，Standard

[1] この例では，説明を簡単にするために組を使っているが，組の使用は本質的ではなく，同様の例を組を使わずに構築可能である.

```
fn x => (x 1,
  fn x2 => (x2 1,
    fn x3 => (x3 1,
      ···
        fn x_n => x_n 1
        ··· )))))
```

図 6: 冗長な型代入の適応を引き起こす例

ML など現実の言語に対して，実用コンパイラによるコンパイル過程において取得する必要がある．現在，本稿で提案したアルゴリズムを，大堀が提案し開発を進めている Standard ML の拡張言語 SML♯のコンパイラ上に実装し，性能評価を行い，従来のアルゴリズムや手続き的処理によって型代入を実現しているアルゴリズムとの比較等の研究を行っている．

それらの評価結果を基に，アルゴリズムの方式やデータ構造に更なる改良などを加え，より効率の良い実用的な型推論アルゴリズムの構築技術の確立を目指す．アルゴリズム自身の改良点としては，例えば以下のような可能性が考えられる．$\mathcal{DW}$ では，部分式の型推論の結果得られた差分 S'，Δ' から，新しい S，Δ を構築する処理を頻繁に行っている．しかし，実際に型代入や型変数を生成するのは単一化を行う場合など，アルゴリズムの一部に過ぎない．そこで，$\mathcal{DW}$ を S，Δ の差分ではなく新しい S，Δ 全体を返すように変形し，S，Δ に対する変更は実際に型代入や型変数を生成している部分に集約すれば，より一層の効率の向上が得られる可能性がある．

また，アルゴリズム $\mathcal{DW}$ で扱う型代入はすべて整形されているものであるため，$\mathcal{DW}$ では型変数への型代入の適用は常に 1 回で停止する．しかし一方で，受け取る型代入が整形されていることを必要条件とせず，またアルゴリズム内部で型代入が整形されていることを保存しないようなアルゴリズム $\mathcal{DW}$ の変形も考えられる．これは，明示的代入操作をもつラムダ計算（例えば [1] などを参照）などで行われている代入に関する簡約規則に相当する操作と見なすことができる．整形されていない型代入の型項への適用には型代入の推移的な適用が必要である反面，このようにアルゴリズムを変形することで vacuous な型変数 (vacuous type variable)[9] に対する不要な型代入の適用が抑止できることが期待でき，全体として効率が向上する可能性がある．

5 結論

本稿では，宣言的で実用上より効率的であると思われる型推論アルゴリズム $\mathcal{DW}$ を提案した．アルゴリズム $\mathcal{DW}$ では，従来の型推論アルゴリズム $\mathcal{W}$ を元に，項や環境への型代入の適用を遅延させるなどして大きな項や項の集合を操作する回数を減らすことで，アルゴリズムの実用上の性能の改善を試みた．さらに，アルゴリズム $\mathcal{DW}$ が健全であることを示した．

構築したアルゴリズムは，我々が開発中の Standard ML を拡張した言語 SML♯のコンパイラに実装し，その性能評価を行う予定である．

参考文献

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J Levy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[2] J. Gallier and W. Snyder. Complete sets of transformations for general E-unification. *Theoretical Computer Science*, 67(2):203–260, 1989.

[3] P. Hudak, S. Peyton Jones, P. Wadler, B. Boutel, J. Fairbairn, J. Fasel, M. Guzman, K. Hammond, J. Hughes, T. Johnsson, D. Kieburtz, R. Nikhil, W. Partain, and J. Perterson. Report on programming language Haskel a non-strict, purely functional language version 1.2. *SIGPLAN Notices, Haskell special issue*, 27(5), 1992.

[4] P. Kanellakis, H.G. Mairson, and J. Mitchell. Unification and ML type reconstruction. In J-L Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. MIT Press, 1990.

[5] A.J. Kfoury, J. Tiuryn, and P. Urzyczyn. An analysis of ML typability. *Journal of the ACM*, 41(2):368–398, 1994.

[6] X. Leroy. *The Objective Caml User's Manual*. INRIA Rocquencourt, B.P. 105 78153 Le Chesnay France, 1997.

[7] R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[8] R. Milner, R. Tofte, M. Harper, and D. MacQueen. *The Definition of Standard ML*. The MIT Press, revised edition, 1997.

[9] A. Ohori. A polymorphic record calculus and its compilation. *ACM Transactions on Programming Languages and Systems*, 17(6):844–895, 1995. A preliminary summary appeared at ACM POPL, 1992 under the title "A compilation method for ML-style polymorphic record calculi".