アルゴリズム 5－5
（1989. 1. 25）

高階論理におけるユニフィケーション アルゴリズムの複雑さについて

原尾 政輝　岩沼 宏治

山形大学工学部 情報工学科

あらまし ユニフィケーション問題とは、ある言語において項の集合 $\{t_1,t_2,\ldots,t_n\}$が与えられたとき、$t_1\sigma=t_2\sigma=\ldots=t_n\sigma$となる置換（ユニファイア）$\sigma$が存在するかという問題である。ユニフィケーションアルゴリズムは、定理証明の機械化や記号処理等と関連して重要な問題である。本稿では、高階論理におけるユニフィケーションについてアルゴリズム論的立場から考察する。一般に、２階以上の論理式のユニフィケーション問題は決定不能である事が分かっている。しかし、２つの項の間のユニフィアを枚挙する、完全かつ無矛盾で非冗長なアルゴリズムは存在する。本稿では、このアルゴリズムを基に、ユニフィケーション問題が可解となるクラスを示し、実用上有用な２階のクラスでの計算の複雑さを考察する。特に、２階のクラスでＮＰ-完全となるものが存在することを示す。

COMPLEXITY OF UNIFICATION ALGORITHM IN HIGHER-ORDER-LOGICS

Masateru HARAO and Kouji IWANUMA

Department of Information Engineering
Faculty of Engineering, YAMAGATA UNIVERSITY
4-3-16 Zyounan ,Yonezawa,992 JAPAN

Abstract Unification problem is to decide whether for given terms $\{t_1,t_2,\ldots,t_n\}$ in some language there exists some substitution(unifier) such that $t_1\sigma=t_2\sigma=\ldots=t_n\sigma$. Unification algorithm is important concerning with mechanization of theorem proving in logic system,knowledge processing system and so on. It has been already shown that the unifiability of terms in higher order logic (order ≥ 2) is ,in general,undecidable. But,an algorithm which enumerates all unifiers of arbitrary 2 terms is also given. The algorithm ,further,satisfies completeness,consistency and nonredundancy properties. Based on this algorithm,some solvable classes in higher order logic are shown and complexity of 2nd order logic which is important from practical viewpoint are discussed. Especially,it is demonstrated that a NP-complete class in 2nd order logic exists.

1．まえがき ユニフィケーション問題とは、ある言語において項$\{t_1,t_2,\ldots,t_n\}$が与えられたとき、$t_1\sigma=t_2\sigma=\ldots=t_n\sigma$となる置換（ユニファイア）$\sigma$が存在するかという問題である。ユニファイアを求めるアルゴリズムは、知識処理、記号処理、言語処理等などにおいて処理の基本となっている。特に、論理に基づく知識処理においては、導出原理に基づく定理証明の機械化で本質的である。現在、論理型言語の能力を柔軟にするため、高階論理へ拡張する試みがある。そこでは、高階論理のためのユニフィケーションアルゴリズムの性質を解明する事が重要な課題となっている。本稿では、高階論理言語を型付λ-式で定式化し、その言語のユニフィケーション問題について考察する。

一般に、２階以上の論理式のユニファイアが存在するかどうかは決定不能であることが知られている[2]。しかし、２つの項の間のユニファイアを枚挙することは可能である。これまでにも完全かつ無矛盾で非冗長な枚挙アルゴリズムが構成されている[5]。本稿では、このアルゴリズムを基に、いくつかの計算可能なクラスおよびＮＰ-完全なクラスの存在などを示す。

２章では、型付λ-式とユニフィケーション問題の定式化を行なう。３章では、ユニフィケーション問題が可解となるクラスについて考察する。４章では、２階の論理式においてＮＰ-完全となるクラスが存在する事を示す。５章はむすびである。

２．型付λ式とユニフィケーション

２.１ 型付λ式

λ式に型を導入したものを型付λ式という。型付λ式は、常に正規形を持ちセマンティクスも簡潔である。また、自然演繹法などの証明法とも自然な対応があり、高階論理の定式化に用いられる。すべての有限階の論理を含む論理体系は、第ω階の論理あるいは有限タイプ理論と呼ばれている。本稿はこの言語を基本とする。

【定義 2.1】 基本となるタイプの有限集合をT_0とするとき、タイプの集合Ｔとは$T_0\subset T$かつ以下の条件を満たす最小の集合である：$\alpha,\beta\in T$ならば$\alpha\to\beta\in T$.特に、個体定数のタイプνおよび述語定数のタイプoはT_0の要素であるとする。また、タイプ$\alpha\to(\beta\to\gamma)$を$\alpha\to\beta\to\gamma$と略記する。

【定義 2.2】 タイプの階数を次のように定義する：
(a) $\alpha\in T_0$に対し、$Odr(\alpha)=1$.
(b) $\alpha=(\alpha_1\times\alpha_2\times\ldots\times\alpha_n)\in T$ならば
$Odr(\alpha\to\beta)=\max_i\{Odr(\alpha_i)+1,Odr(\beta)\}$

各タイプの変数および定数記号は可算個あるとし、変数の集合をΔ、定数の集合をΓで表す。言語Ｌの原始記号は、論理記号［,］,～,∨,∀,そしてΔ、Γよりなる。

【定義 2.3】 項の集合Terと項ｔのタイプ$\tau(t)$を帰納的に次のように定義する：
(a) $t\in\Delta\cup\Gamma$ならば$t\in Ter$.
(b) $t_1,t_2\in Ter$で$\tau(t_1)=(\alpha_1\to\alpha_2)$,$\tau(t_2)=\alpha_1$ならば、$(t_1\ t_2)\in Ter$で、$\tau((t_1\ t_2))=\alpha_2$.ここで関数適用は左結合的とする。即ち、$(t_1\ t_2\ t_3)=((t_1\ t_2)t_3)$.
(c) $t\in Ter$,$\tau(t)=\alpha_2$で、xがタイプα_1の変数とする。このとき、ｔの**抽象化**$(\lambda x.t)\in Ter$で、$\tau(\lambda x.t)=(\alpha_1\to\alpha_2)$である。

【例 2.1】 $\tau(P)=(\alpha\to\beta\to\gamma)$、$\tau(x)=\alpha$、$\tau(y)=\beta$なる変数とする。このとき次の性質が成立する：
(1)$\tau(Px)=(\beta\to\gamma)$
(2)$\tau(Pxy)=(\gamma)$
(3)$\tau(\lambda x.P)=(\alpha\to\alpha\to\beta\to\gamma)$
(4)$\tau(\lambda x.Px)=(\alpha\to\beta\to\gamma)$
(5)$\tau(\lambda x.Pxy)=(\alpha\to\gamma)$
(6)$\tau(\lambda xy.Pxy)=(\alpha\to\beta\to\gamma)$

【定義2.4】 タイプoを持つ項を**論理式**と言う。もしA,B が論理式であり、u が任意のタイプ変数であれば、～A,[A∨B],∀u.Aは論理式である。

自由変数、閉論理式等は通常の定義に従うとする。項ｔがｋ階の項とは、ｔの中に出現する自由変数が、高々ｋ階のときである。特に論理式は、その中にｍより大きい階数の定数,変数が存在しないとき2(m-1)階の論理式と呼び、そこでもしｍ階の変数が∀，∃記号で束縛されていないとき、(2m-3)階の論理式と呼ぶ。xが変数、ｔおよびＡを項とする。Ａにおいて自由変数xの出現を全てｔで置き換えたものを$A[t/x]$で表す。ただし$\tau(t)=\tau(x)$とする。

【定義 2.5】 **λ-変換**とは次の３つの規則からなる。
(1)α-変換：ｙがＡでxに関して自由なとき$(\lambda x.A)$を$(\lambda y\ A[x/y])$で置き換える。
(2)β-変換：ｔがＡでxに関して自由なとき$(\lambda x.A)t$を$A[x/t]$で置き換える。
(3)η-変換：$\tau(A)=(\alpha\to\beta)$，$\tau(z)=\alpha$でｚはＡで自由ならば、Ａを$\lambda z(Az)$で置き換える。

λ式で、これ以上λ-変換を用いて変形出来ないものを**正規形**とよぶ。型付λ式は、必ず有限回の手数で正規形へ変換出来、しかも名前の付け替えを除いて一意的である事が知られている。正規形は一般に次の形

で表される：$t=\lambda u_1u_2\ldots u_n.\phi(t_1,t_2,\ldots,t_p)$
ϕをヘッドと呼ぶ。ヘッドϕは、$\phi\in\Gamma\cup\{u_1,u_2,\ldots,u_n\}$のとき**確定的(rigid)**、そうでない（自由変数の）とき**不定的（flex)**と言う。またtをそれぞれ**確定項、不定項**と呼ぶ。

2．2高階ユニフィケーション

次の形をした置き換えの有限集合σを代入と呼ぶ：
$\sigma=\{x_1/u_1,x_2/u_2,\ldots,x_n/u_n\}$
ただし、$\tau(x_1)=\tau(u_1),\ldots,\tau(x_n)=\tau(u_n)$.
また、項$t=\lambda x_1x_2\ldots x_n.\phi(t_1,t_2,\ldots,t_p)$に代入$\sigma$を施したものを$\sigma(t)$と書き、$[\lambda x_1x_2\ldots x_n\phi(t_1,t_2,\ldots,t_p)](u_1,u_2,\ldots,u_n)$の正規形として定義する。代入の合成等は通常の通りとする。

【定義2.6】 t_1,t_2を$\tau(t_1)=\tau(t_2)$なる項とする。このとき、$\sigma(t_1)=\sigma(t_2)$（$\sigma(t_1)=t_2$）なる代入σをt_1,t_2のユニファイア（マッチング）と呼ぶ。

λ式におけるユニフィケーションでは、λ変換の操作が入るために非常に複雑になる。2階以上のλ項のユニフィケーション問題は一般に決定不能である事が示されている[2]。しかし、ユニファイアの同値類の代表元を枚挙する部分アルゴリズムが Huet により与えられており、以下そのアルゴリズムを基にユニファイアの存在性に関する性質を調べる。λ式は次の正規形に変換されているとする。

$$t=\lambda u_1u_2\ldots u_n.f(t_1,t_2,\ldots,t_p)$$
$$r=\lambda v_1v_2\ldots v_n.@(r_1,r_2,\ldots,r_q)$$

ここで、@は確定項、f は自由項とする。また、t_i ($1\leqq i\leqq n$)は一般の正規項である。

【定義2.7】 不一致集合Dとは、$\tau(t)=\tau(r)$となる正規形の順序対 <t，r>の有限集合である。D中の対を構成する正規項をDの**構成項**と呼ぶ。また、一方が確定項で他方が自由項であるものを**rigid-flex 対**、両方とも flex 項の場合を **flex-flex 対**と呼ぶ。

対象とする項では、全称記号や存在記号等は適当な操作によって取除かれていると仮定する。ユニファイアは与えられた正規項の対から、次の Simple および Match の手順によって導出される。

《簡約規則 Simple》 Simple は不一致集合Dを受取り、正規項の対のヘッドが同じであればそれを取り除き、新しい簡約された不一致集合を生成する。

Simple(D)＝DとなるDを**規約**であると呼ぶ。以下断らない限り不一致集合と言った時には規約なものを指すとする。

《Match 手順》 Match はDの中の flex-rigid 対を受け取り、自由項のヘッドの変数に可能な代入の有限集合を返す。Match 手順は大別すると、Imitation rule と Projection rule より構成されており、この対から高々p+1個の代入を出力する。

Imitation rule

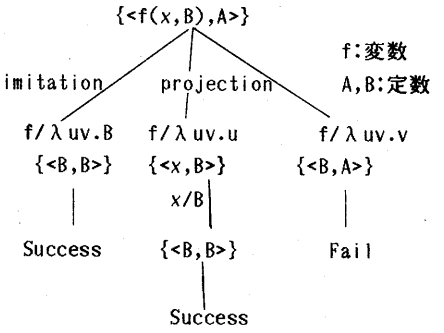
$\sigma=f/\lambda x_1x_2\ldots x_p.@(\underline{h}_1,\underline{h}_2,\ldots,\underline{h}_q)$.
ここで、$\underline{h}_i=h_i(x_1,x_2,\ldots,x_p)(1\leqq i\leqq q)$, $\tau(\underline{h}_i)=\tau(r_i)$、はtやrには現れない新しい関数変数 h_i, $\tau(x_i)=\tau(t_i)$,を用いて定義される新しい関数である。

Projection rule

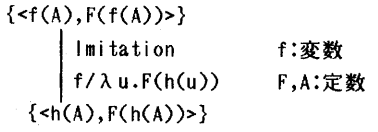
$\sigma=f/\lambda x_1x_2\ldots x_p.x_i(\bar{g}_1,\bar{g}_2,\ldots,\bar{g}_m)$
ここで、$\bar{g}_i=g_i(x_1,x_2,\ldots,x_p)$ $(1\leqq i\leqq q)$, $\tau(g_i)=\alpha_1\times\alpha_2\times\ldots\times\alpha_p\rightarrow\gamma_i$,はtやrには現れない新しい関数変数$g_i$より定義される新しい関数である。ただし、$\tau(x_i)=\tau(t_i)=\alpha_i=\gamma_1\times\gamma_2\times\ldots\times\gamma_m\rightarrow\beta$とする。

マッチング木： $\tau(t)=\tau(r)$とするとき、<t，r>のマッチング木とは、不一致集合、成功(Success)、失敗(Fail)のいずれかを節点のラベルとして持つ次のように構成される木のことである：
(1)木の根は、N_0=Simple<{t,r}>である。
(2)節点NがSでもFでもないなら、Nの要素から任意の1つのflex-rigid対を採り、Match により代入の集合$\{\sigma_1,\sigma_2,\ldots,\sigma_k\}$を構成する。各$\sigma_i$に対してSimple<$\sigma_i$N>を計算し、Nの子節点としてNから$\sigma_i$でラベル付けされたアークを付け加える。

【例 2.2】 Matching 木の構成例



(a)停止するマッチング木



(b)停止しないマッチング木

図1．マッチング木の構成

Imitation 規則は、fの形を@に合わせる操作であり、1階の論理式のユニフィケーションの一般化に対応している。Projection 規則は、t自身の部分項を取り出し、確定項とユニファイ可能な形に変形する操作である。また、マッチング木で、根からS節点へのパスに含まれるアークにラベル付けされた代入の合成は、tとrのユニファイアになることが知られている（健全性）。このマッチング木を幅優先で構成する手続は、独立なクラスのユニファイアを全て重複なしに生成する事が知られている(完全性と非冗長性)[6,10]。一般に、マッチング木は無限木になる場合がある。従って、節点の選びかたや代入規則の適用には非決定的な性質があり、アルゴリズムの効率は適用順序や代入規則に大きく依存する。また高階の場合は、複数のmguが存在し、アルゴリズムも停止しない。

3．ユニフィケーションアルゴリズムの可解性

3.1 制限付ユニフィケーション

実用的観点からは、階数、構文、代入に制限を加えた場合が重要である。ある条件の下では、ユニフィケーションが可解なクラスや見通しの良いクラスが存在する。Imitation,Projection 代入には、次の性質が成立つ。

【命題3.1】 Projection rule $\sigma = f/\lambda x_1 x_2 \ldots x_p.x_i(\bar{g}_1,\bar{g}_2,\ldots,\bar{g}_m)$, $\bar{g}_j = g_j(x_1,x_2,\ldots,x_p)$ $(1 \leqq j \leqq q)$, $\tau(g_j) = \alpha_1 \times \alpha_2 \times \ldots \times \alpha_p \rightarrow \gamma_j$, が適用可能なための必要十分条件は記号の階数が次の各条件を満たすことである。

1) $Odr(g_j) \geqq Odr(f) \geqq \max\{Odr(x_i)\}+1$. $(1 \leqq j \leqq q)$

2) $Odr(x_i) \geqq \max\{Odr(\bar{g}_j)\}+1$.

3) $Odr(x_i) \geqq 2$ かつ $Odr(g_j) \geqq 3$,

又は $Odr(x_i)=1$ かつ $g_j=\phi$. $(1 \leqq j \leqq q)$ □

Odr (f)	Odr(x) max(x),x_i		Odr(g_i)	Projection 可能性
2	1	1	ϕ	$f/\lambda x_1 x_2 \ldots x_p.x_i$
	≧2	※	※	×①
3	1	※	※	× od(f)≦2
	2	1	ϕ	$f/\lambda x_1 x_2 \ldots x_p.x_i$
	2	2	≦2	× ③
	2	2	3	◎
	2	2	≧4	× ①
	≧3	※	※	×
4	3<	※	※	×
	3	≦3	4	◎

表1 Projection 可能性

可能なProjection代入の例を表1に示す。ただし、maxは$\max\{Odr(x_i)\}$を、◎,×はProjection が可能および不可能な事を、※は任意の条件の下でを、①等は満たされない命題3.1の条件を示す。

【命題3.2】 Imitation $\sigma = f/\lambda x_1 x_2 \ldots x_p.@(\underline{h}_1,\underline{h}_2,\ldots,\underline{h}_q)$, $\underline{h}_i = h_i(x_1,x_2,\ldots,x_p)(1 \leqq i \leqq q)$, $\tau(\underline{h}_i)=\tau(r_i)$, が適用可能なための必要十分条件は、記号の階数が次の式を満たすことである。

1) $Odr(h_i) \geqq Odr(f) \geqq \max\{Odr(x_i)\}+1$,

2) $Odr(@) \geqq \{Odr(\underline{h}_i)\}+1$. □

【系3.1】 $<t,r>$を $t = \lambda u_1 u_2 \ldots u_n.f(t_1,t_2,\ldots,t_p)$, $r = \lambda v_1 v_2 \ldots v_n.@(r_1,r_2,\ldots,r_q)$ なる2階の項対とする。このとき、Imitation rule はtのヘッドの階数によって次の様なものに限られる：

1)階数1のとき：$t = \lambda u_1 u_2 \ldots u_n.x$ だから、

$\sigma = x/@(r_1,r_2,\ldots,r_q)$.

2)階数2のとき：$\sigma = f/\lambda x_1 x_2 \ldots x_p.@(\underline{h}_1,\underline{h}_2,\ldots,\underline{h}_q)$ $\tau(\underline{h}_i)=\tau(r_i)$であって、① $\tau(r_i) \in T_O$なら、

$\underline{h}_i = h_i(x_1,x_2,\ldots,x_p)(1 \leqq i \leqq q)$かつ $Odr(@)=2$.

② $\tau(r_i)=\delta_1 \times \cdots \times \delta_s \rightarrow \delta$ なら、$\underline{h}_i = \lambda w_1 \ldots w_s.h_i(x_1,x_2,\ldots,x_p,w_1,\ldots,w_s)$, $\tau(w_i)=\delta_i (1 \leqq i \leqq s)$.

$Odr(\underline{h}_i) \leqq 2$なら、$Odr(@) \leqq 3$となる． □

【命題 3.3】 $<t,r>$を2階の項対で、2階の変数はその変数を根とする部分項にはその変数を含まないとする。また、4階以上の定数は出現しないとする。そのとき、Match 手順は必ず停止し、$<t,r>$の全てのマッチングを導出する。

（証明）Imitation ,Projection は、共に項の長さ(項に含まれる記号の数)を少なくとも一つ減少させることから明らか。□

3.2 Imitation だけによるユニファイア

Imitation 代入だけによるユニフィケーションは、項依存グラフによってユニファイ可能性を決定出来る。項依存グラフは次のように構成される。

$D = \{<t_1,r_1>,<t_2,r_2>,\ldots,<t_n,r_n>\}$を任意の不一致集合とする。$t_i$は自由項、$r_i$は確定項とする。また、$T_i,R_i$を t_i,r_i の（親から子への有向辺を持つ）構文木で、ヘッドから最初の自由変数節点までの部分を抜出したものとする。R_iの辺をr-辺と呼ぶ。T_iはt_iのヘッドの自由変数節点 $\{f_i\}$だけからなるグラフである事に注意する。このとき、グラフGは、T_i,R_iより次の様に構成されるグラフである。

①T_i,R_iの同一の自由変数名の節点をまとめて一つにする。

②T_iからR_iのヘッドへ無向辺を付加する。d-辺

（依存辺）と呼ぶ。

③ｄ-辺で連結されている節点間にｄ-辺を付加する（推移的閉包をつくる）。この関係にあるｄ-辺節点集合をＶで表す。

例 3.1 依存グラフの構成例

Ｄ＝{<f(x),R(R(g(x),y),z)>,<f(x),R(g,w)>}

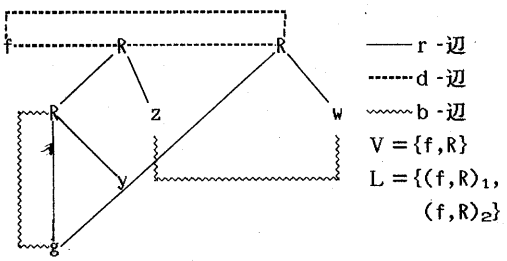


図2 不一致集合と依存グラフ

ユニファイ可能性は、節点集合Ｖのすべての対に対して、次の手続きを適用して決定される。

《Imitation 手続き：Imitate(Ｌ,Ｇ)》

Ｇ:=グラフ,Ｌ:={(u,v)∈Ｖ×Ｖ},Ｂ:=空(ｂ-辺集合).

```
begin
 if Ｌ≠空 then
    Ｌの要素ｐ=(u,v)をとる;
    Ｌ:=Ｌ－{ｐ}と置く;
    Imit(ｐ)を実行する;
    Imitate(Ｌ,Ｇ);
 else
     if ｂ-辺がある（Ｂ≠空） then
        ｂ-辺をｄ-辺に変える（Ｌ":=Ｂ）;
        新たに付け加えられるｄ-辺集合をＬ",
        新らしいグラフをＧ"とする;
        Ｌ:=Ｌ"; Ｇ:=Ｇ";
        Imitate(Ｌ,Ｇ);
     else 停止（Ｇを出力する）;
end

《手続き Imit(ｐ)》   /* ｐ=(u,v) */
begin
 if 両節点のラベルが定数 then
    if ラベルが同じ then
      if 未サーチの子節点がある then
       未サーチの子節点対ｑ=（r,s)へ移る;
       Imit(ｑ);
      else
       節点対にサーチ済の印を付け親節点ｒへ帰る;
       Imit(ｒ);
     if ラベルが違う then 停止(failを返す);
 if どちらかが束縛変数 then 停止(fail）;
 if どちらかが自由項 then
    if {u,v}間に辺がない then
      {u,v}をｂ-辺で結びＢ:=Ｂ∪{u,v}とする;
       出来たグラフをＧ"と置く;
       親節点ｒに帰る;
    else（既に辺がある）直ちに親節点ｒへ帰る;
    Imit(ｒ);
end
```

ここでｄ-辺は、ユニフィケーションの結果一致しなければならない項、変数の間の関係を示す。またｂ-辺は、ｄ-辺の拘束条件より出てくる新たな依存関係を示す。Imitation アルゴリズムはそのすべての可能性を調べ、もし条件を満たせば依存関係を示すグラフＧ"を出力する。これを**依存グラフ**と呼ぶ。もし条件が満たされなければ、Imit アルゴリズムによって fail が出力される。依存グラフは、ユニファイされる頂点間の満たすべき条件を表しており、ユニファイ可能性に関する決定手続きを提供する。

【定義3.1】依存グラフＧ"の**依存閉路**とは、ｒ-辺を少なくとも一つ含むＧ"の閉路のことである。

【定理3.1】　不一致集合Ｄに対して、依存グラフＧ"が構成出来、Ｇ"に依存閉路が存在するとする。このとき、Ｄに対する任意の有限 imitation 代入列σは失敗する(ユニファイアは存在しない）。すなわち、Simple(σＤ)＝fail.

（略証）依存閉路で、自由項ｇと確定項Ｒがｄ-辺で結ばれ、ｒ-辺によりｇとＲのループがあるとする。また、このループ間の関数をＴとする(Ｔは恒等写像の場合もある)。このとき、{ｇ,Ｒ(Ｔ(ｇ))}のユニファイアを求める必要がある。明らかにこれは再帰的になり、ユニファイ不可能である。

【定理3.2】不一致集合Ｄに対して、依存グラフＧ"が構成出来、Ｇ"に依存閉路が存在しないとする。このとき、Ｄに対するある有限 imitation 代入列σが存在し成功する（ユニファイアが存在する）。即ち、Simple(σＤ)＝Success.

（略証）Ｇ"は有限であるから、依存閉路がなければ必ず有限回で停止する。

依存グラフを用いたユニファイ可能性の判定アルゴリズムは、１階論理のユニフィケーションアルゴリズムの手法と基本的には同じである[9]。従って、項対<ｔ,ｒ>がImitation だけを用いてユニファイ可能かどうかはｔとｒの項の長さに関する線形時間で判定で

きる。以下、len(t,r)で、t,rの項の長さの和を表す。

【定理3.3】項対<t,r>がImitation代入規則だけを用いてユニファイ可能かどうかは、len(t,r)に関する線形時間で決定可能である。

4．NP-完全なクラス

実用上重要なクラスとして、2階のクラスがある。本節では、2階の変数しか含まない、狭い範囲のユニフィケーション問題についてもNP-完全なものが存在する事を示す。前節の結果よりProjectionは、$\sigma = \{ f/\lambda x_1 x_2 \ldots x_p . x_i \}$となることに注意する。

【定義4.1】S,T を項の集合とする。代入をθとするとき、$\theta S = \{\theta s \mid s \in S\}$とする。代入$\theta$がSとTをユニファイ（マッチ）するとは、$\theta(S) = \theta(T)$（$\theta(S) = T$）が成立つことである。S,Tがユニファイ（マッチ）可能かどうかを決定する問題を、集合ユニファイ（マッチング）問題と呼び、SUP（SMP）で表す。

【命題4.1】[8] S,T を高々2階の定数および1階の変数だけからなる1階の項の集合とする。このとき、S,TのSMP問題はNP-完全である（Tが変数を含まない場合にも成立つ）。

項tを唯一種の2階の関数変数fを持ち、fをヘッドとする部分項にはfは含まない項とし、rを2階以下の定数だけからなる1階の項とする。このような項対<t,r>のユニフィケーション問題は、NP-hardであることを示す。

命題4.1の条件を満たす項の集合を、$S = \{s_1, s_2, \ldots, s_m\}$、$T = \{t_1, t_2, \ldots, t_n\}$とする。$s_i, t_i$のタイプはすべて$\alpha$であると仮定する。また、S、Tに現れない次の様な新しい変数と定数を考える。

変数：2階の2項変数 f, $\tau(f) = \alpha \times \cdots \times \alpha \to \alpha$

定数：2階の2項関数定数 #

2階のn項関数定数 @

1階の定数記号 d

このとき、項s,tを次のように定義する：

$s = \#(f(s_1, s_2, \ldots, s_m), f(d, \ldots, d))$

$t = \#(@(t_1, t_2, \ldots, t_n)), @(d, \ldots, d))$

【補題4.1】S,T を高々2階の定数および1階の変数だけからなる1階の項の集合とする。$\theta S = T$なる代入θが存在することと、$\sigma s = t$なる代入σが存在することは同値である。

（証明）（→）θが存在し、$\theta S = T$とする。$\xi = f/\lambda w_1 w_2 \ldots w_m . @(w_1, w_2, \ldots, w_m)$とおく。$\xi(f(s_1, s_2, \ldots, s_m)) = @(s_1, s_2, \ldots, s_m)$, $\xi(f(d, \ldots, d)) = @(d, \ldots, d)$,であるから、$\theta' = \theta \cup \xi$はs,tのマッチングとなる。

（←）sとtが代入θでマッチングされているとする。θはImitation代入とProjection代入の組合わせで構成されている。

1)まず、不一致集合として、$\{<f(s_1, s_2, \ldots, s_m), @(t_1, t_2, \ldots, t_n)>, <f(d, \ldots, d)), @(d, \ldots, d)>\}$が導出される。Projection では、第2成分の項対がユニファイ不可能だから、このときの代入はImitationしかない。

2)次の不一致集合は次の様になる。

$\{<h_1(s_1, s_2, \ldots, s_m), t_1>, \ldots, <h_n(s_1, s_2, \ldots, s_m), t_n>, <h_1(d, \ldots, d), d>, \ldots, <h_n(d, \ldots, d), d>\}$.

これ以降、$h_1, \ldots, h_n$に対する代入は、Imitation ではありえない。それは、h_iをdをヘッドとするImitation 代入で置き換えれば、$<s_i, t_i>$なる不一致集合が出てくるが、この置き換えはt_iが変数を持たないので不可能だからである。従って、$h_i (1 \leq i \leq n)$に対して、次のProjection 代入を行なう。

$h_i / \lambda w_1 w_2 \ldots w_m . w_j$ $(1 \leq j \leq m)$

3)この代入の結果次の不一致集合が導出される。

$<\{s_1, s_2, \ldots, s_m\}, \{t_1, \ldots, t_n\}>$.

ここで、s_iはあるjに対して、t_jと一致すればよいことを意味するとする。

4)sとtはユニファイ可能であるから、全ての$t_j (1 \leq j \leq m)$に対してあるs$_i$$(1 \leq j \leq n)$と$\theta_i$が存在して、$\theta_i$ $s_i = t_i$となる。よって、$\theta = \theta_1 \cup \ldots \cup t_n$とすれば、$\theta S = T$。即ちSMP問題は解（ユニファイア$\theta$）を持つ。

従って、次の性質が成立つ。

【定理4.1】項tを唯一種の2階の関数変数fを持ち、fをヘッドとする部分項にはfは含まない項とし、rを2階以下の定数だけからなる1階の項とする。このような項対<t,r>のユニフィケーション問題は、NP-hard である。

次に、この問題がNPである事を示す。

【補題4.2】項tを唯一種の2階の関数変数fを持ち、fをヘッドとする部分項にはfは含まない項とし、rを2階以下の定数だけからなる1階の項とする。このような項対を<t,r>とするとき、<t,r>がImitation のみを用いてユニファイ可能かどうかは、n = len(r)とすると、O(n)で決定出来る。

（証明）Imitation によるユニファイ可能性を、前章のアルゴリズムにより判定すれば良いが、このImitation 手順は、rが変数を含まないので簡単になり、概

略次の手順に従えばよい。まず、項tは、t＝#(...,f(t_1,t_2,...,t_p),$(...,f($u_1$,$u_2$,...,$u_p$),...)のような形で、fはネストしない形でしか現れない。また、rは変数を含まない。従って、依存グラフは図３の様になる。そこで、r_1,r_2 等は適当な定数で、r_1(㊛)，r_2(㊛)は f(t_1,t_2,...,t_p),f(u_1,u_2,...,u_p)と同じタイプを持つ。

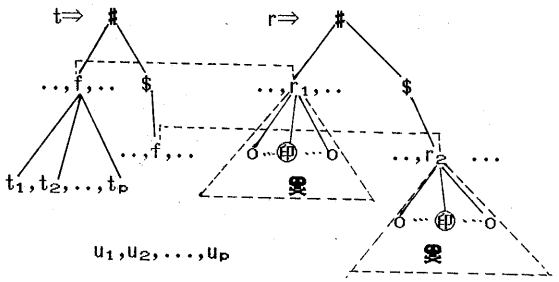


図３ <t，r>の依存グラフ

rは変数を含まないから、ユニファイ可能なためにはd-辺（図３の┈┈）で結合されたヘッドr$_i$以下の部分木（図３の㊛）は全く同一である必要がある。この部分木を一致項と呼び、その構文木を一致木と呼ぶ。ユニファイ可能性は、一致木が全く同じかどうかをチェックすればよい。一致木は変数を含まないから、高々O(n)で検査出来るのは明らか。㊐

全ての一致項の集合において、不一致集合Dが空でないとする。すると、一致木の対応する各節点で少なくとも一つ異なる定数記号を持つ節点が存在する。その節点に㊙を付けそれを一致木の不一致節点とよぶ。

【補題4.3】 一致木が不一致節点㊙を持つとする。このとき、ユニファイ可能であるための必要十分条件は、各不一致㊙節点に対し、ある祖先が存在し、そこでProjectionによってユニファイ可能な事である。

（略証）一つのImitation代入で異なった定数の置き換えは出来ないので、㊙の節点以下ではImitation が失敗する事は明らかである。従って、ユニファイ可能なためにはそれ以前にProjection で変形する必要がある。逆は明らか。 ㊐

Imitation 代入を用いてマッチング木を途中まで構成しており、いま図4のように変数fとヘッド@を持つ一致木$\alpha,\beta,...,\gamma$のImitation 代入 $\sigma=f/\lambda x_1 x_2...x_p.@(\underline{h}_1,\underline{h}_2,...,\underline{h}_q)$を行なう状況とする。$\sigma$を施すと$\sigma$ (f(t_1,t_2,...,t_p)) ＝@ (h_1(t_1,t_2,...,t_p),...,h_q(t_1,t_2,...,t_p)) となり、新しい不一致集合は

D(α)＝{<h_i(v_1,v_2,...,v_p),α_i>},(1≦i≦q)

D(β)＝{<h_i(u_1,u_2,...,u_p),β_i>},(1≦i≦q)

...,

D(γ)＝{<h_i(v_1,v_2,...,v_p),γ_i>},(1≦i≦q)

となる。ただし、$\alpha_i,\beta_i,...,\gamma_i$は一致項$\alpha,\beta,...,\gamma$のi成分とする。即ち、変数 h_iは{$\alpha_i,\beta_i,...,\gamma_i$}すべてとユニファイ可能でなければならない。また、この様な不一致集合対<h_i,α_j>,...等の数は、一致木の葉の数を越える事はないことに注意する。

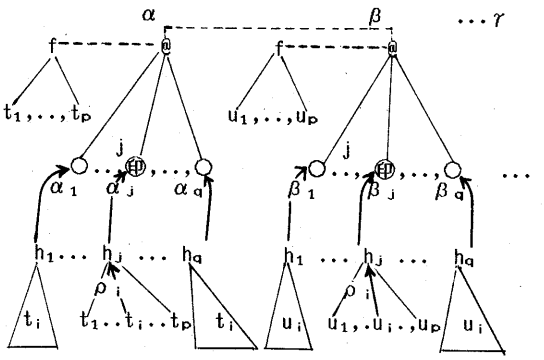


図４ Imitation,Projection による不一致集合の変化

ここで、一致木の不一致節点 j にProjection ρ_i= f/λ$x_1x_2...x_p.x_i$ (i=1,2,...,p) を適用するとする。この時新たに次の依存関係が出てくる：

D_j={<t_i,α_j>,<u_i,β_j>,...,<v_i,γ_j>},(i=1,2,...,p)

即ち、D_jは同時にユニファイ可能でなければならない。τ_i=&(s_i,t_i,...,u_i),(i=1,2,...,p) と定義し、Projection tupleと呼ぶ。τ_iと&(α_j,β_j,...,γ_j)がユニファイ可能であれば、頂点 j におけるユニファイはρ_iの下で成功する事に注意する。

【補題 4.4】 一致項$\alpha,\beta,...,\gamma$の節点jを根とする部分項を$\alpha_j,\beta_j,...,\gamma_j$とし、項$term_j$ = &($\alpha_j,\beta_j,...,\gamma_j$)とする。このとき、この節点 j に関してProjection でユニファイ可能なための必要十分条件は、<τ_i,term>,i=1,2,...,p,のどれか一つがユニファイ可能な事である。㊐

【補題4.5】 <τ_i,$term_j$>のユニファイ可能性は、len(τ_i,$term_j$)に関する線形時間で決定可能である。

（略証）τ_iの各成分$s_i,t_i,...,u_i$は１階の項でしかも２階の変数は持たない。従って、<τ_i,$term_j$>のユニファイ可能性は１階述語論理の項のユニファイ可能性と等価である。即ち、線形時間で決定出来る[91]。㊐

依存グラフのサイズをnとする。len(τ_i)＝len(t_i,u_i,...,v_i) ,len($term_j$)＝len($\alpha_j,\beta_j,...,\gamma_j$)である。従って次の関係が成立つ。

$$\sum_{i}^{P} len(\tau_i, term_j) = (\sum_{i} len(\tau_i)) + len(term_j)$$
$$= \sum_{i} (len(t_i, u_i, \ldots, v_i) + len(term_j)) < n .$$

【定理4.2】項ｔを唯一種の２階の関数変数ｆを持ち、ｆをヘッドとする部分項にはｆは含まない項とし、ｒを２階以下の定数だけからなる１階の項とする。このような項対＜ｔ，ｒ＞のユニフィケーション問題は、ＮＰである。

（証明）1)項対＜ｔ，ｒ＞に対して、依存グラフ、一致グラフを構成する。依存グラフのサイズをｎ、一致木グラフのサイズをｍとする。ｎ＞ｍに注意する。
2)Imitation だけでユニファイ可能かどうかを判定する。定理3.3より、高々Ｏ(ｎ)の手数で決定出来る。
3)不一致節点があれば、全ての不一致節点の祖先が存在するような節点集合Ｖをゲスする。これは、高々Ｏ(2^m)の可能性が存在する。
4)ゲスしたＶの各要素に対して、Projection 可能性を検査する。Ｖの位数は高々Ｏ(ｍ)で、一つの頂点に対するProjection 可能性は補題4.5より高々Ｏ(ｎ)で検査出来る。従って、全てのＶの頂点に対するProjection 可能性の検査の手数は高々Ｏ(ｍｎ)である。
5)従って、1)～4)までに要する手数は高々Ｏ(n^2)、即ち多項式オーダ、で決定可能である。□

【定理4.3】項ｔを唯一種の２階の関数変数ｆを持ち、ｆをヘッドとする部分項にはｆは含まない項とし、ｒを２階以下の定数だけからなる１階の項とする。このような項対＜ｔ，ｒ＞のユニフィケーション問題は、ＮＰ-完全である。□

５．むすび

本稿では、高階論理を型付λ式を用いて定式化し、ユニフィケーション問題を考察した。特に、１階述語論理の場合の拡張である Imitation 規則によるユニファイ可能性の判定問題、２階のλ項に制限した場合の性質についてアルゴリズム論的立場から幾つかの性質を明らかにした。結果としては、高階の場合のユニフィケーションは難しいことが分かった。

現在一般的ユニフィケーションアルゴリズムを論理型言語 Prolog を用いて実装している。このユニフィケーションアルゴリズムは効率を考えて深さ優先の戦略を用いているが、これまでの議論で明らかなように一般には停止しない。したがって、そのアルゴリズムは不完全になっていまう。しかし、一方の項には高階の変数が存在せずかつ関数および述語変数の階数を２階に限った場合にはそのアルゴリズムは必ず停止する。この様に、用法や構文に適当な制限を付ける事によって、論理型言語に取り込む事が可能と思われる。

文献

[1]P.B.Andrew : An Introduction to Mathematical Logic and Type Theory , Academic Press.Inc, 1986.

[2]W.D.Goldfarb:The Undecidability of The Second-Order Unification Problem,TCS,Vol.13,pp.225～230,1981.

[3]L.Henkin:Completeness in The Theory of Types, JSL,Vol.15,pp81～91,1986.

[4]G.Huet and B.Lang: Proving and Applying Program Transformations Expressed with Second-Order Patterns,Acta Informatica,11, pp31-55, 1978

[5]G.Huet: A Mechaniztion of Type Theory,Proc.of 3rd IJCAI,pp139～146,1973.

[6]G.P.Huet:A Unification Algorithm for typed λ-calculus,TCS,Vol.1,pp27～57,1975.

[7]岩沼,原尾：高階論理におけるユニフィケーションについて,信情学技報 COMP87-27,pp13～20,1987.

[8]D.Kapur,P.Narendran: NP-Completeness of The Set Unification and Matching Problems, LNS. 230,pp.489～ 495,1986.

[9]M.S.Paterson,M.Wegman : Linear Unification, JCSS 16,pp.158～167,1978.

[10]D.C.Jensen ,T.Pietrozykowski: Mechanizing ω-order Type Theory Through Unification,Theoretical Computer Science 3,pp123～171,1976.

[11]J.H.Siekmann : Universal Unification, 7th International Conf.on Automated Deduction, Lecture note,No.170,pp1～42,1984.

[12]D.Simon : A Linear Time Algorithm for a subclass of Second Order Instantiation,7th International Conf.on Automated Deduction, Lecture note, No.170,pp209～223,1984.