数理解析研究所講究録
1093 巻 1999 年 142-147

スキーママッチングにおける計算の複雑さ

山田 敬三 (Keizo Yamada)　平田 耕一 (Kouichi Hirata)　原尾 政輝 (Masateru Harao)
九州工業大学 情報工学部

要旨

一階論理式に述語変数を許した式を**スキーマ**(*schema*)という．本論文では，スキーマと一階閉論理式のマッチングである**スキーママッチング**(*schema matching*)，および，その判定問題である**スキーママッチング問題**(*schema matching problem*)について議論する．スキーマは二階の変数である述語変数を含むので，スキーママッチングは二階マッチングの特別な場合となる．そこで，まず，スキーママッチングの手続きとして，二階マッチングと同様に，三つの書き換え規則，**簡単化**(*simplification*)，**模倣**(*imitation*)，**射影**(*projection*)を定義する．次に，スキーママッチングを，変数の型によって**述語マッチング**(*predicate matching*)と**項マッチング**(*term matching*)に分ける．そして，述語マッチング問題は表現長の線形時間で解けるが，項マッチング問題は NP 完全になることを示す．一方，個体自由変数を含まないスキーママッチング問題は，多項式時間で解けることを示す．さらに，このアルゴリズムに基づいたマッチング代入の抽出法について議論する．

1　はじめに

これまでに得られている知識や規則を**スキーマ**(*schema*)として蓄え，このスキーマを援用して問題解決を行なう知識処理方式を**スキーマ誘導**(*schema-guided*)処理という．このスキーマ誘導処理は，プログラムの自動合成 [6]，数学の証明問題 [3]，プログラム変換 [12]，類推処理 [9] において広く適用されている．スキーマ誘導処理では，援用するスキーマを決定するための操作として，**マッチング**(*matching*)が重要となる．

マッチングとは，与えられた二つの表現のうち，一方の表現に代入を適用することで他方と一致させる操作，すなわち一方向の**単一化**(*unification*)のことである．このときの代入を**マッチング代入**(*matcher*)といい，マッチング代入が存在する二つの表現は**マッチング可能**(*matchable*)であるという．また，表現がマッチング可能か否かを判定する問題を**マッチング問題**(*matching problem*)という．項のマッチング問題の場合，階数が 3 以下のマッチング問題は決定可能である [4] が，二階マッチング問題でさえ NP 完全である [1]．また，制限つき二階マッチング問題の計算量の解析や二階マッチングの効率的アルゴリズムについての研究がなされている [2, 8, 10]．

本論文では，スキーマ誘導処理におけるスキーマを単純型理論 [11, 12] を用いて定式化する．まず，単純型理論の β 標準形のうち，抽象化を含まない二階の項を**スキーマ項**(*schema term*)，述語変数を含む論理式を**スキーマ** (*schema*) として定義する．ここで，論理式は，論理型をもつスキーマ項として扱われる．さらに，スキーマと自由変数を含まない一階論理式の組の有限集合を**スキーマ表現**(*schema expression*)といい，スキーマ項と変数を含まないスキーマ項の組の有限集合を**項表現**(*term expression*)という．本論文では，スキーマ表現のマッチングとマッチング問題，すなわち**スキーママッチング**(*schema matching*)と**スキーママッチング問題**(*schema matching problem*)について議論する．

スキーマの定義から，スキーママッチングは，二階マッチング [1, 12] の特別な場合となる．二階マッチングの手続きは，**簡単化**(*simplification*)，**模倣**(*imitation*)，**射影**(*projection*) という三つの書き換え規則によって実現される [11, 12, 13]．そこで，本論文でも，スキーママッチングをこの三つの書き換え規則で実現する．

次に，スキーママッチングを，マッチングの対象となる変数の型によって，**述語マッチング**(*predicate matching*)と**項マッチング**(*term matching*)の二つの段階に分ける．述語マッチングは，スキーマ表現を項表現に変換する手続きである．型の条件から，述語マッチングは簡単化と模倣の繰り返しとして実現できる．したがって，述語マッチング問題は表現長の線形時間で解け，解である項表現も一意に求まる．一方，項マッチングは，項表現におけるマッチングであり，簡単化，模倣，射影のすべての手続きを用いて実現される．一般に，二階マッチング問題は NP 完全である [1]．本論文では，項マッチング問題も NP 完全になることを示す．したがって，一般に，スキーママッチング問題は NP 完全となる．

述語マッチングによって得られる項表現は，関数変数の引数に関数変数が出現しないような項表現となる．そこで，項表現 $E = \{\langle F(s_1^i,\cdots,s_{n_i}^i), f(t_1^i,\cdots,t_{m_i}^i)\rangle \mid i \in I\}$ に対して，各 $f(t_1^i,\cdots,t_{m_i}^i)$ に模倣と射影の適用可能性の情報をラベルとして割り当てた**既約判定項**(*reduced judgement term*) を定義する．そして，この既約判定項を用いることにより，個体自由変数を含まない項マッチング問題が多項式時間で解けることを示す．さらに，この既約判定項を用いたマッチング代入の抽出法について議論する．

2 準備

基本型(*elementary type*) の有限集合 T_0 に対して，**型**(*type*) の集合 T を T_0 を含み次の操作について閉じている最小の集合とする: $\tau_1,\cdots,\tau_n \in T$ かつ $\mu \in T_0$ ならば $\tau_1 \times \cdots \times \tau_n \to \mu \in T$.

互いに素である**定数**(*constant*) の集合 $\mathcal{C}$ と**変数**(*variable*) の可算無限集合 $\mathcal{V}$ に対して，すべての $d \in \mathcal{C} \cup \mathcal{V}$ に対して，d に対応する型 $\tau(d) \in T$ が存在すると仮定する．このとき，**項**(*term*) の集合 $\mathcal{T}$ を以下のように帰納的に定義する:

(1) $t \in \mathcal{C} \cup \mathcal{V}$ かつ $\tau(t) \in T_0$ のとき，$t \in \mathcal{T}$.

(2) (**適用**) $\tau(d) = \tau_1 \times \cdots \times \tau_n \to \mu$ となる $d \in \mathcal{C} \cup \mathcal{V}$ と $t_i \in \mathcal{T}$ $(1 \le i \le n)$ に対して，$d(t_1,\cdots,t_n) \in \mathcal{T}$ であり，$\tau(d(t_1,\cdots,t_n)) = \mu$ である．

(3) (**抽象**) $\tau(t) = \mu \in T_0$ となる $t \in \mathcal{T}$ と $\tau(v_i) = \tau_i$ となる $v_i \in \mathcal{V}$ $(1 \le i \le n)$ に対して，$\lambda v_1 \cdots v_n.t \in \mathcal{T}$ であり，$\tau(\lambda v_1 \cdots v_n.t) = \tau_1 \times \cdots \times \tau_n \to \mu$ である．

特に上の (1),(2) だけから構成される項を**スキーマ項**(*schema term*) という．さらに，項を二階に制限するため，変数 $v \in \mathcal{V}$ の型 $\tau(v)$ を $\tau(v) \in T_0$, または $\tau(v) = \tau_1 \times \cdots \times \tau_n \to \mu$ かつ $\tau_i \in T_0$ と仮定する．

$o \in T_0$ を論理型とする．また，$\wedge, \vee, \supset, \neg \in \mathcal{C}$ とし，$\tau(\wedge) = \tau(\vee) = \tau(\supset) = o \times o \to o$ かつ $\tau(\neg) = o \to o$ であるとする．さらに，$\tau(\varphi) = o$ かつ $\tau(x) \in T_0$ ならば $\tau(Qx.\varphi) = o$ $(Q \in \{\forall, \exists\})$ であるとする．このとき，$\tau(t) = o$ となる項 t を**論理式**(*formula*) という．

$\tau(v) = \tau_1 \times \cdots \times \tau_n \to o$ となる変数 $v \in \mathcal{V}$ を**述語変数**(*predicate variable*) といい，$\tau(v) = \tau_1 \times \cdots \times \tau_n \to \tau$ かつ $\tau \neq o$ となる変数 $v \in \mathcal{V}$ を**関数変数**(*function variable*) という．述語変数を含み，関数変数を含まない論理式を**スキーマ**(*schema*) という．以後，スキーマを $\Phi, \Psi, \cdots$, 一階論理式を $\varphi, \psi, \cdots$, 述語変数を $P, Q, R, \cdots$, 関数変数を $F, G, H, \cdots$, 関数定数を $f, g, h, \cdots$ で表す．

項 $\lambda v_1 \cdots v_n.t$ と $\tau(v_i) = \tau(t_i)$ となる項 t_i $(1 \le i \le n)$ に対して，$t[v_1 := t_1, \cdots, v_n := t_n]$ で，t 中の変数 v_i を t_i で同時に置き換えることによって得られる項とする．これは，ラムダ計算 [11, 13] における $(\lambda v_1 \cdots v_n.t)(t_1 \cdots t_n)$ と同等の表現である．

$\tau(v_i) = \tau(t_i)$ となる変数 v_i と項 t_i $(1 \le i \le m)$ に対して，$\{t_1/v_1, \cdots, t_m/v_m\}$ を**代入**(*substitution*) という．スキーマ項 t と代入 θ に対して，$t\theta$ を以下のように帰納的に定義する:

(1) $t = c$ のとき，$t\theta = c$.

(2) $t = x$ に対して，$t'/x \in \theta$ ならば $t\theta = t'$ であり，そうでなければ $t\theta = x$ である．

(3) $t = f(t_1,\cdots,t_n)$ $(f \in \mathcal{C})$ のとき，$t\theta = f(t_1\theta,\cdots,t_n\theta)$.

(4) $t = F(t_1,\cdots,t_n)$ $(F \in \mathcal{V})$ のとき，$\lambda v_1 \cdots v_n.t'/F \in \theta$ ならば $t\theta = t'[v_1 := t_1\theta, \cdots, v_n := t_n\theta]$ であり，そうでなければ $t\theta = F(t_1\theta,\cdots,t_n\theta)$ である．

(5) $t = Qx.t'$ $(Q \in \{\forall, \exists\})$ のとき，$t\theta = Qy.((t'\{y/x\})\theta)$ である．ただし，y は新しい変数とする．

ラムダ計算 [11, 13] と同様に，スキーマ Φ に対して Φ に現れる $\forall, \exists$ の束縛変数の名前を系統的に付け換えることによって Ψ が得られるとき，Φ は Ψ と α 同値であるといい，$\Phi =_\alpha \Psi$ と表す．

3 スキーママッチング

スキーマ表現と項表現を合わせて単に**表現**(*expression*) という．表現 $E = \{\langle \Phi_i, \varphi_i\rangle \mid i \in I\}$ に対して，$\Sigma_{i\in I}(|\Phi_i| + |\varphi_i|)$ を E の**大きさ**(*size*) といい，$|E|$ と表す．ただし，項の大きさは，その項に出現する (補助記号を除く) 記号の数とする．

表現 $E = \{\langle \Phi_i, \varphi_i\rangle \mid i \in I\}$ に対して，任意の $i \in I$ で $\Phi_i\theta =_\alpha \varphi_i$ となる代入 θ を求める操作を，**スキーママッチング**(*schema matching*) という．そのような代入 θ を E の**マッチング代入**(*matcher*) という．表現 E にマッチング代入が存在するとき，E は**マッチング可能**(*matchable*) であるという．また，表現がマッチング可能か否かを判定する問題を**スキーママッチング問題**(*schema matching problem*) という．

二階マッチングの手続きは，**簡単化** (*simplification*), **模倣** (*imitation*), **射影** (*projection*) の三つの書き換え規則によって実現できる [12]．そこで，本論文でも，スキーママッチングの手続きを三つの書き換え規則で実現する．

スキーママッチングの簡単化では，$Qx.t$ $(Q \in \{\forall, \exists\})$ を t に書き換える．そこで，t に出現する x を**論理束縛記号**($w, w_1, w_2, \cdots$ で表す) という特別な個体定数で置き換える．また，二階マッチング [12] の模倣では，束縛変数を模倣することはできない．そこで，スキーママッチングの模倣でも，論理束縛記号の模倣ができないように定義する．さらに，$Qx.t$ の模倣では，x を後で使えるようにするために，述語変数の引数を一つ増やす．

1. 簡単化 (*simplification*):
$\{\langle c, c\rangle\} \cup E \Rightarrow E \ (c \in \mathcal{C})$,
$\{\langle f(s_1, \cdots, s_n), f(t_1, \cdots, t_n)\rangle\} \cup E$
$\Rightarrow \{\langle s_1, t_1\rangle, \cdots, \langle s_n, t_n\rangle\}$
$\cup E \ (n \geq 1, f \in \mathcal{C})$,
$\{\langle Qx.\Phi, Qy.\varphi\rangle\} \cup E \quad (Q \in \{\forall, \exists\})$
$\Rightarrow (E - \{\langle Qx.\Phi, Qy.\varphi\rangle\})$
$\cup \{\langle \Phi\{w/x\}, \varphi\{w/y\}\rangle\}$.

2. E の F に関する模倣 (*imitation*):
$E \Rightarrow E\{\lambda\overline{v}.c/F\}$,
$\langle F(s_1, \cdots, s_n), c\rangle \in E$ かつ $c \neq w$ のとき,
$E \Rightarrow E\{\lambda\overline{v}.f(H_1(v_1, \cdots, v_n),$
$\cdots, H_m(v_1, \cdots, v_n))/F\}$,
$\langle F(s_1, \cdots, s_n), f(t_1, \cdots, t_m)\rangle$
$\in E \ (n \geq 0, m \geq 1)$,
かつ $\tau(F(s_1, \cdots, s_n))$
$= \tau(f(t_1, \cdots, t_m))$ のとき,
$E \Rightarrow E\{\lambda\overline{v}.Qx.P(v_1, \cdots, v_n, x)/F\}$,
$\langle F(s_1, \cdots, s_n), Qx.\varphi\rangle \in E \ (Q \in \{\forall, \exists\})$
かつ $\tau(F(s_1, \cdots, s_n)) = \tau(Qx.\varphi)$ のとき.

3. E の F に関する射影 (*projection*):
$E \Rightarrow E\{\lambda\overline{v}.v_i/F\}$,
$\langle F(s_1, \cdots, s_n), t\rangle \in E \ (n \geq 1, 1 \leq i \leq n)$
かつ $\tau(s_i) = \tau(t) \ (1 \leq i \leq n)$ のとき.

$\Rightarrow$ の推移閉包を $\Rightarrow^*$ と表す. このとき, 二階マッチング [12] と同様に, 以下の定理が成り立つ:

定理 1 表現 E がマッチング可能であることと, $E \Rightarrow^* \emptyset$ となることは同値である.

例 1 表現 $E = \{\langle \forall x.P(x), \forall z.\neg(\forall y.p(z, y))\rangle\}$ に対するスキーママッチングは以下のようになる:

$\{\langle \forall x.P(x), \forall z.\neg(\forall y.p(z, y))\rangle\}$ (簡単化)
$\Rightarrow \{\langle P(w_1), \neg\forall y.p(w_1, y)\rangle\}$ (模倣 θ_1)
$\Rightarrow \{\langle \neg P_1(w_1), \neg\forall y.p(w_1, y)\rangle\}$ (簡単化)
$\Rightarrow \{\langle P_1(w_1), \forall y.p(w_1, y)\rangle\}$ (模倣 θ_2)
$\Rightarrow \{\langle \forall y.P_1'(w_1, y), \forall y.p(w_1, y)\rangle\}$ (簡単化)
$\Rightarrow \{\langle P_1'(w_1, w_2), p(w_1, w_2)\rangle\}$ (模倣 θ_3)
$\Rightarrow \{\langle p(H_1(w_1, w_2), H_2(w_1, w_2)), p(w_1, w_2)\rangle\}$
(簡単化)
$\Rightarrow \{\langle H_1(w_1, w_2), w_1\rangle, \langle H_2(w_1, w_2), w_2\rangle\}$
(射影 θ_4)
$\Rightarrow \{\langle w_1, w_1\rangle, \langle H_2(a, w_1, w_2), w_2\rangle\}$
(射影 θ_5)
$\Rightarrow \{\langle w_1, w_1\rangle, \langle w_2, w_2\rangle\}$ (簡単化)
$\Rightarrow^* \emptyset$.

ただし,
$\theta_1 = \{\lambda v_1.\neg P_1(v_1)/P\}$,
$\theta_2 = \{\lambda v_1.\forall y.P_1'(v_1, y)/P_1\}$,
$\theta_3 = \{\lambda v_1 v_2.p(H_1(v_1, v_2), H_2(v_1, v_2))/P_1'\}$,
$\theta_4 = \{\lambda v_1 v_2.v_1/H_1\}$,
$\theta_5 = \{\lambda v_1 v_2.v_2/H_2\}$
とする.
したがって, E のマッチング代入は,
$\{\lambda v_1.\neg\forall y.p(v_1, y)/P\}$ となる. 実際,

$(\forall z.P(z))\{\lambda v_1.\neg\forall y.p(v_1, y)/P\}$
$= \forall u.(P(u)\{\lambda v_1.\neg\forall y.p(v_1, y)/P\})$
$= \forall u.((\neg\forall y.p(v_1, y))[v_1 := u])$
$= \forall u.\neg\forall y.p(u, y)$

となる. また, この例から, 限量子の模倣において引数を増やす必要があることがわかる.

4 スキーママッチング問題の計算量

本論文では, スキーママッチングを, 対象となる変数が述語変数か関数変数かの違いによって, **述語マッチング**と**項マッチング**の二つの段階に分けて考える.

述語マッチング(*predicate matching*) とは, 与えられたスキーマ表現に対して, それに述語変数および述語定数が現れなくなるまで, 模倣と簡単化を適用する手続きである. 述語マッチングには, 射影が適用できない. なぜならば, スキーマ表現 E に対して $\langle P(s_1, \cdots, s_n), \varphi\rangle \in E$ とする. このとき, $\tau(P(s_1, \cdots, s_n)) = \tau(\varphi) = o$ となるが, s_i は項なので $\tau(s_i) \neq o$ となるからである. したがって, 以下の補題が成り立つ:

補題 1 スキーマ表現 E の述語マッチング問題は $O(|E|)$ 時間で解け, その解である項表現も一意である.

スキーマ表現は, 述語マッチングによって, 論理束縛記号を含む項表現に変換される. このようにして得られた項表現に対するマッチングが**項マッチング**(*term matching*) となる. 項マッチングでは, 簡単化と模倣以外に射影も適用される. したがって, 項マッチング問題は, 関数変数の引数に関数変数は出現しないような項表現に対する二階マッチング問題となる. 項マッチング問題に対して, 次の定理が成り立つ:

定理 2 項マッチング問題はNP完全である.

証明 項マッチング問題が NP に属することは, 二階マッチング問題の NP 完全性 [1] より明らか. 完全性は, 項マッチング問題の MONOTONE 1-IN-3 3SATからの還元による. □

したがって，一般に，スキーママッチング問題はNP完全となる．

5 個体自由変数を含まないスキーママッチング

定理2では，個体自由変数の存在が証明の本質である．そこで本節では，個体自由変数を含まないような項表現の項マッチング問題が多項式時間で解けることを示す．ここでのアイデアは，述語マッチングで得られた項表現

$E = \{\langle F(s_1^i, \cdots, s_{n_i}^i), f(t_1^i, \cdots, t_{m_i}^i)\rangle \mid i \in I\}$

の各 $f(t_1^i, \cdots, t_{m_i}^i)$ に模倣と射影の適用可能性の情報をラベルとして割り当てることにある．

定義 1 s をスキーマ項 $H(s_1, \cdots, s_n)$, t を変数を含まないスキーマ項とする．また，$P(t) = \{i \mid t = s_i\ (1 \leq i \leq n)\}$ とする．このとき，t の s に関する**判定項**(*judgement term*) $T(t, s)$ を，以下のようにラベル付けられた項と定義する．ここで，$*$ は新しい個体定数とする．

(1) t が論理束縛記号 w のとき，$T(w, s) = *^{P(w)}$.

(2) t が個体定数 c のとき，$T(c, s) = c^{P(c)}$.

(3) $t = f(t_1, \cdots, t_m)$ であり，t_i の s に関する判定項が $T_i(t_i, s)$ $(1 \leq i \leq m)$ のとき，$T(t, s) = f^{P(t)}(T_1(t_1, s), \cdots, T_m(t_m, s))$.

例 2 スキーマ項 s_1, s_2, s_3 および変数を含まないスキーマ項 t_1, t_2, t_3 を以下のように定義する:

$$s_1 = H(w_1, f(a, w_1), b),$$
$$s_2 = H(w_2, g(w_2, a, b), a),$$
$$s_3 = H(w_1, f(a, w_1), b)\ (= s_1),$$
$$t_1 = f(f(a, w_1), f(b, w_1)),$$
$$t_2 = f(g(w_2, a, b), f(a, w_2)),$$
$$t_3 = f(f(a, w_1), f(a, w_2)).$$

このとき，t_i の s_i に関する判定項 $T(t_i, s_i)$ $(i \in \{1, 2, 3\})$ は以下のようになる:

$$T(t_1, s_1) = f^{\emptyset}(f^{\{2\}}(a^{\emptyset}, *^{\{1\}}), f^{\emptyset}(b^{\{3\}}, *^{\{1\}})),$$
$$T(t_2, s_2) = f^{\emptyset}(g^{\{2\}}(*^{\{1\}}, a^{\{3\}}, b^{\emptyset}), f^{\emptyset}(a^{\{3\}}, *^{\{1\}})),$$
$$T(t_3, s_3) = f^{\emptyset}(f^{\{2\}}(a^{\emptyset}, *^{\{1\}}), f^{\emptyset}(a^{\emptyset}, *^{\emptyset})).$$

判定項のラベルは，その位置において射影が適用できる引数を表しており，もしラベルが $\emptyset$ であるならば，その位置には射影は適用できない．

スキーマ項 t の**頭部**(*head*) $\mathrm{hd}(t)$ を以下のように定義する: $t \in \mathcal{C} \cup \mathcal{V}$ のとき $\mathrm{hd}(t) = t$; $t = f(t_1, \cdots, t_n)$ のとき $\mathrm{hd}(t) = f$. 同様に，判定項 T のラベル付き頭部(*labeled head*) $\mathrm{lhd}(T)$ を以下のように定義する: $T = c^P$ または $T = *^P$ のとき $\mathrm{lhd}(T) = T$; $T = f^P(t_1, \cdots, t_n)$ のとき $\mathrm{lhd}(T) = f^P$.

定義 2 項表現 $\{\langle s_i, t_i\rangle \mid i \in I\}$ に対して，$\mathrm{hd}(s_i) = F$ とし，t_i の s_i に関する判定項を $T(t_i, s_i)$ とする．このとき，$\{T(t_i, s_i) \mid i \in I\}$ の**共通判定項**(*common judgement term*) $\sqcap_{i \in I} T(t_i, s_i)$ を次のように定義する．

(1) 任意の $i \in I$ に対して，$\mathrm{lhd}(T(t_i, s_i)) = c^{P_i}$ (c は個体定数か $*$) ならば，$\sqcap_{i \in I} T(t_i, s_i) = c^{\cap_{i \in I} P_i}$.

(2) 任意の $i \in I$ に対して，
$T(t_i, s_i) = f^{P_i}(t_1^i, \cdots, t_m^i)$ ならば，
$\sqcap_{i \in I} T(t_i, s_i) = f^{\cap_{i \in I} P_i}(\sqcap_{i \in I} T(t_1^i, s_i), \cdots, \sqcap_{i \in I} T(t_m^i, s_i))$.

(3) $\mathrm{lhd}(T(t_i, s_i)) = f_i^{P_i}$ かつ $\mathrm{hd}(T(t_k, s_k)) \neq \mathrm{hd}(T(t_j, s_j))$ となる $k, j \in I$ が存在するとき，
$\sqcap_{i \in I} T(t_i, s_i) = *^{\cap_{i \in I} P_i}$.

例 3 例2の判定項 $T(t_i, s_i)$ $(i \in \{1, 2, 3\})$ に対して，$\{T(t_1, s_1), T(t_2, s_2)\}$ と $\{T(t_2, s_2), T(t_3, s_3)\}$ の共通判定項 $\sqcap_{i \in \{1,2\}} T(t_i, s_i)$ と $\sqcap_{i \in \{2,3\}} T(t_i, s_i)$ はそれぞれ以下のようになる:

$$\sqcap_{i \in \{1,2\}} T(t_i, s_i) = f^{\emptyset}(*^{\{2\}}, f^{\emptyset}(*^{\{3\}}, *^{\{1\}})),$$
$$\sqcap_{i \in \{2,3\}} T(t_i, s_i) = f^{\emptyset}(*^{\{2\}}, f^{\emptyset}(a^{\emptyset}, *^{\emptyset})).$$

定義 3 項表現 $E = \{\langle s_i, t_i\rangle \mid i \in I\}$ に対して，$\mathrm{hd}(s_i) = F$ とする．このとき，E の**既約判定項**(*reduced judgement term*) $\sqcap E$ を，$\sqcap_{i \in I} T(t_i, s_i)$ に以下の規則を可能な限り適用して得られるラベル付き項とする:

$\sqcap_{i \in I} T(t_i, s_i)$ の部分項 t' に対して，$t' = f^{\emptyset}(t_1', \cdots, t_m')$ かつ $t_i' = *^{\emptyset}$ となる i $(1 \leq i \leq m)$ が存在するとき，$\sqcap_{i \in I} T(t_i, s_i)$ 中の t' を $*^{\emptyset}$ で置き換える．

例 4 例2の s_i と t_i に対して，$E_1 = \{\langle s_1, t_1\rangle, \langle s_2, t_2\rangle\}$, $E_2 = \{\langle s_2, t_2\rangle, \langle s_3, t_3\rangle\}$ とすると，例3より $\sqcap E_1 = f^{\emptyset}(*^{\{2\}}, f^{\emptyset}(*^{\{3\}}, *^{\{1\}}))$, $\sqcap E_2 = *^{\emptyset}$ となる．

補題 2 項表現 $E = \{\langle s_i, t_i\rangle \mid i \in I\}$ に対して，$s_i = F(s_1^i, \cdots, s_m^i)$ とする．このとき，$\sqcap E$ は $O(|E|^2)$ 時間で構成できる．

証明 スキーマ項 s と t に対して，$T(t, s)$ は $O(|s| \times |t|)$ 時間で構成できるので，$\sqcap_{i \in I} T(t_i, s_i)$ は $O(|E|^2)$ 時間で構成できる．また，定義2より，$|\sqcap_{i \in I} T(t_i, s_i)| \leq \min_{i \in I}\{|t_i|\}$ となる．さらに，定義3より，$\sqcap E$ は，$\sqcap_{i \in I} T(t_i, s_i)$ から $O(|\sqcap_{i \in I} T(t_i, s_i)|)$ 時間で構成できる．したがって，$\sqcap E$ は E から $O(|E|^2)$ 時間で構成できる．□

補題 3 項表現 $E = \{\langle s_i, t_i\rangle \mid i \in I\}$ に対して, $s_i = F(s_1^i, \cdots, s_n^i)$ とする. このとき, E がマッチング可能であることと, $\sqcap E \neq *^{\emptyset}$ であることは同値である.

証明 ($\rightarrow$) 補題の対偶に対する $\sqcap E$ の構造に関する帰納法による. ($\leftarrow$) は明らか. □

定理 3 個体自由変数を含まないスキーマ表現 E のスキーママッチング問題は $O(|E|^2)$ 時間で解ける.

証明 E に対して, 以下のような手続き Matchを考える:

Procedure Match(E)
E_1 := PredMat(E);
  while $E_1 \neq \emptyset$ *do begin*
    select $\langle s, t\rangle \in E_1$;
    $F := \mathrm{hd}(s)$;
    $E_2 := \{\langle t, s\rangle \in E_1 \mid \mathrm{hd}(s) = F\}$;
    $T := \sqcap E_2$;
    if $T = *^{\emptyset}$ *then*
      return "not matchable" and *halt*;
  end
return "matchable";

ここで, PredMat(E) は E の述語マッチングである. よって, E_1 は個体自由変数を含まない項表現となる. したがって, 補題3より, Match(E) が "matchable" と返すことと, E がマッチング可能であることは同値である. また, 補題1より, PredMat(E) の計算時間は $O(|E|)$ である. さらに, E_2 を構成するための計算時間は高々 $O(|E|)$ である. $E_2 = \{\langle s_i, t_i\rangle \mid i \in I\}$ とすると, $\sqcap E_2$ を構成するための計算時間は, 補題2より $O(|E_2|^2)$ となる. E_1 の異なる関数変数の数を k とし, その関数変数によって E_1 を $E_1^1 \cup \cdots \cup E_1^k$ と分割する. それぞれの E_1^j $(1 \leq j \leq k)$ が上の E_2 に対応する. したがって, Match(E) の計算時間は, $O(|E|) + \Sigma_{j=1}^{k} O(|E_1^j|^2) \leq O((|E_1^1| + \cdots + |E_1^k|)^2) \leq O(|E|^2)$ となる. □

6 マッチング代入の抽出

前節では, 与えられたスキーマ表現にマッチング代入が存在するか, というスキーママッチング問題を考察した. 一方, 二階マッチングおよび本論文のスキーママッチングでは, マッチング代入が存在しても, それが一意とは限らない [12]. そこで本節では, 個体自由変数を含まないスキーマ表現におけるマッチング代入の抽出法について議論する.

手続き Match と同様に, 述語マッチングで得られた項表現を関数変数によって分割して考える. 項表現 $E = \{\langle s_i, t_i\rangle \mid i \in I\}$ に対して, $s_i = F(s_1^i, \cdots, s_n^i)$ とする. このとき, $\sqcap E$ を用いてマッチング代入を抽出する.

定義 4 項表現 $E = \{\langle s_i, t_i\rangle \mid i \in I\}$ に対して, $s_i = F(s_1^i, \cdots, s_n^i)$ とする. $\sqcap E \neq *^{\emptyset}$ のとき, $\sqcap E$ の**項集合**(*term set*) $T_{\sqcap E}$ を以下のように帰納的に定義する:

(1) $\sqcap E = *^P$ のとき, $T_{\sqcap E} = \{\lambda\overline{v}.v_j \mid j \in P\}$.
(2) $\sqcap E = c^P$ のとき,
  $T_{\sqcap E} = \{\lambda\overline{v}.c\} \cup \{\lambda\overline{v}.v_j \mid j \in P\}$.
(3) $\sqcap E = f^P(\sqcap E_1, \cdots, \sqcap E_m)$ のとき,
  $T_{\sqcap E} = \{\lambda\overline{v}.f(t_1, \cdots, t_m) \mid \lambda\overline{v}.t_i \in T_{\sqcap E_i}\}$
    $\cup \{\lambda\overline{v}.v_j \mid j \in P\}$.

例 5 例2 の s_1, s_2, t_1, t_2 に対して, $E_1' = \{\langle s_1, t_1\rangle\}$, $E_2' = \{\langle s_2, t_2\rangle\}$ とすると, 例2より
$\sqcap E_1' = f^{\emptyset}(f^{\{2\}}(a^{\emptyset}, *^{\{1\}}), f^{\emptyset}(b^{\{3\}}, *^{\{1\}}))$,
$\sqcap E_2' = f^{\emptyset}(g^{\{2\}}(*^{\{1\}}, a^{\{3\}}, b^{\emptyset}), f^{\emptyset}(a^{\{3\}}, *^{\{1\}}))$
となるので, $T_{\sqcap E_1'}$ と $T_{\sqcap E_2'}$ は以下のようになる.

$$T_{\sqcap E_1'} = \left\{ \begin{array}{l} \lambda v_1 v_2 v_3.f(v_2, f(v_3, v_1)), \\ \lambda v_1 v_2 v_3.f(v_2, f(b, v_1)), \\ \lambda v_1 v_2 v_3.f(f(a, v_1), f(v_3, v_1)), \\ \lambda v_1 v_2 v_3.f(f(a, v_1), f(b, v_1)) \end{array} \right\},$$

$$T_{\sqcap E_2'} = \left\{ \begin{array}{l} \lambda v_1 v_2 v_3.f(v_2, f(v_3, v_1)), \\ \lambda v_1 v_2 v_3.f(v_2, f(a, v_1)), \\ \lambda v_1 v_2 v_3.f(g(v_1, v_3, b), f(v_3, v_1)), \\ \lambda v_1 v_2 v_3.f(g(v_1, v_3, b), f(a, v_1)), \\ \lambda v_1 v_2 v_3.f(g(v_1, a, b), f(v_3, v_1)), \\ \lambda v_1 v_2 v_3.f(g(v_1, a, b), f(a, v_1)) \end{array} \right\}.$$

また, 例4の E_2 に対して, $\sqcap E_2 = f^{\emptyset}(*^{\{2\}}, f^{\emptyset}(*^{\{3\}}, *^{\{1\}}))$ となるので, $T_{\sqcap E_2} = \{\lambda\overline{v_3}.f(v_2, f(v_3, v_1))\}$ となる.

定理 4 項表現 $E = \{\langle s_i, t_i\rangle \mid i \in I\}$ に対して, $s_i = F(s_1^i, \cdots, s_n^i)$ とする. このとき, $\sqcap E \neq *^{\emptyset}$ ならば, 代入 $\theta \in \{\{t/F\} \mid t \in T_{\sqcap E}\}$ は E のマッチング代入となる.

定理4を用いることにより, 以下のような E のマッチング代入の抽出法を与えることができる: 項表現 $E = \{\langle s_i, t_i\rangle \mid i \in I\}$ に対して, $s_i = F(s_1^i, \cdots, s_n^i)$ とする.

E の**模倣優先**(*imitation-first*) マッチング代入は, 以下のように定義される単一要素項集合 $T_{\sqcap E}^I = \{t\}$ に対して, $\{t/F\}$ となるマッチング代入である:

(1) $\sqcap E = *^P$ のとき,
  $T_{\sqcap E}^I = \{\lambda\overline{v}.v_j \mid j = \min\{k \mid k \in P\}\}$.
(2) $\sqcap E = c^P$ のとき, $T_{\sqcap E}^I = \{\lambda\overline{v}.c\}$.
(3) $\sqcap E = f^P(\sqcap E_1, \cdots, \sqcap E_m)$ のとき,
  $T_{\sqcap E}^I = \{\lambda\overline{v}.f(t_1, \cdots, t_m) \mid \lambda\overline{v}.t_i \in T_{\sqcap E_i}^I\}$.

E の**最左射影優先**(*left-most projection-first*)マッチング代入は, 以下のように定義される単一要素項集合 $T^{LP}_{\sqcap E} = \{t\}$ に対して, $\{t/F\}$ となるマッチング代入である:

(1) $\sqcap E = *^P$ のとき,
$T^{LP}_{\sqcap E} = \{\lambda\overline{v}.v_j \mid j = \min\{k \mid k \in P\}\}$.
(2) それ以外のとき, $P \neq \emptyset$ ならば
$T^{LP}_{\sqcap E} = \{\lambda\overline{v}.v_j \mid j = \min\{k \mid k \in P\}\}$,
$P = \emptyset$ ならば
(a) $\sqcap E = c^P$ のとき, $T^{LP}_{\sqcap E} = \{\lambda\overline{v}.c\}$,
(b) $\sqcap E = f^P(\sqcap E_1, \cdots, \sqcap E_m)$ のとき,
$T^{LP}_{\sqcap E} = \{\lambda\overline{v}.f(t_1, \cdots, t_m) \mid \lambda\overline{v}.t_i \in T^{LP}_{\sqcap E_i}\}$.

例 6 例5の E'_1, E'_2 に対する模倣優先マッチング代入はそれぞれ $\{\lambda\overline{v_3}.f(f(a, v_1), f(b, v_1))/H\}$, $\{\lambda\overline{v_3}.f(g(v_1, a, b), f(a, v_1))/H\}$, となる. 一方, E'_1, E'_2 の最左射影マッチング代入は共に $\{\lambda\overline{v_3}.f(v_2, f(v_3, v_1))/H\}$ となる.

7 おわりに

本論文では, スキーマと一階閉論理式マッチングであるスキーママッチングについて考察した. まず, スキーママッチングの手続きを, 簡単化, 模倣, 射影という三つの書き換え規則で実現した. そして, スキーママッチング問題が一般に NP 完全になることを示した. さらに, 個体自由変数を含まないスキーママッチング問題は, スキーマ表現 E に対して $O(|E|^2)$ 時間で解けることを示した. さらに, このスキーママッチングにおけるマッチング代入の抽出についても議論した.

本論文の結果は, スキーマ誘導知識処理 [6], 特に類推 [9] におけるマッチングが多項式時間で解けることを示唆している. 本論文の手法を用いて, 類推システムを構築することは今後の課題である. また, どのようなマッチング代入を抽出すべきか, という問題も今後の課題である.

スキーママッチングにおいて, 複数存在するマッチング代入のうち, どのマッチング代入を利用するかの基準を導入することも今後の課題である. さらに, マッチング代入が一意に存在する, もしくは, すべてのマッチング代入が多項式時間で求まるようなスキーマの特徴づけも今後の課題である.

謝辞 本研究の一部は, 文部省科学研究補助金萌芽的研究 10878055, 基盤 (C) 07680405, および柏森情報科学振興財団の助成金によるものである.

参考文献

[1] Baxter, L. D.: *The complexity of unification*, Doctoral Thesis, Department of Computer Science, University of Waterloo, 1977.

[2] Curien, R., Qian, Z. and Shi, H.: *Efficient second-order matching*, Proc. of the 7th International Conference on Rewriting Techniques and Applications, LNCS **1103**, 317–331, 1996.

[3] Donat, M. R. and Wallen, L. A.: *Learning and applying generalized solutions using higher order resolution*, Proc. 9th International Conference on Automated Deduction, LNCS **310**, 41–60, 1988.

[4] Dowek, G.: *Third order matching is decidable*, Proc. 7th Annual IEEE Symposium on Logic in Computer Science, 2–10, 1992.

[5] Farmer, W. M.: *Simple second-order languages for which unification is undecidable*, Theoretical Computer Science **87**, 25–41, 1991.

[6] Flener, P.: *Logic program synthesis from incomplete information*, Kluwer Academic Press, 1995.

[7] Garey, M. R. and Johnson, D. S.: *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman and Company, 1979.

[8] 原尾, 岩沼：高階ユニフィケーションアルゴリズムの複雑さについて, 日本ソフトウェア科学会論文誌 **8**, 41–53, 1991.

[9] Harao, M.: *Proof discovery in LK system by analogy*, Proc. 3rd Asian Computing Science Conference, LNCS **1345**, 197–211, 1997.

[10] Hirata, K., Yamada, K. and Harao, M.: *Tractable and intractable second-order matching problems*, 情報基礎理論ワークショップ予稿集 1–6, 1998.

[11] Huet, G. P.: *A unification algorithm for typed λ-calculus*, Theoretical Computer Science **1**, 27–57, 1975.

[12] Huet, G. P. and Lang, B.: *Proving and applying program transformations expressed with second-order patterns*, Acta Informatica **11**, 31–55, 1978.

[13] Snyder, W. and Gallier, J.: *Higher-order unification revisited: Complete sets of transformations*, Journal of Symbolic Computation **8**, 101–140, 1989.