数理解析研究所講究録 1205 巻 2001 年 65-70

Pre-checkingを用いた効率的2階述語マッチングアルゴリズム

久保 憲吾 Kengo Kubo　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
Department of Artificial Intelligence, Kyushu Institute of Technology

要旨

抽象化あるいは一般化された知識であるスキーマを用いたスキーマ誘導型推論システムは, プログラムの自動生成, 問題解決, 証明システムを始め多くの研究に適用されている. 特に, 高階の論理スキーマを用いると, 類推などの高次の推論機構を簡潔に定式化することができ, したがって強力な知識処理システムを構築できる. その反面, システムの実装で重要となるマッチングが一般に困難となる. 本稿では, LK 類推証明システムを構築する観点から, 効率的なスキーママッチングについて考察する. そして, pre-checking に基づいた, 健全かつ少なくとも 1 つの代入を抽出する意味で完全なアルゴリズムを提案する. さらに, このアルゴリズムが個体自由変数を含まない論理スキーマのクラスでは, 効率が上がることを実験結果を用いて示す.

1 はじめに

これまでに得られている知識や規則を抽象化あるいは一般化したスキーマとして蓄え, このスキーマを援用して問題解決を行う知識処理方式をスキーマ誘導処理という. このスキーマ誘導処理は, プログラムの自動合成 [2], プログラム変換 [6], 証明システム [3] を始め多くの研究に適用されている. 特に, 高階の論理スキーマを用いると, 類推 [1, 3] などの高次の推論機構を簡潔に導入でき, 強力な知識処理システムの構築が期待できる. その反面, この手法では援用するスキーマを決定するための重要な操作であるマッチングが一般に困難となる [8].

スキーママッチングとは, マッチング対 $\langle \Phi, \varphi \rangle$ に対し, $\Phi\theta = \varphi$ となるようなマッチング代入 θ を求める操作である. このスキーママッチングは二階マッチングの一種であり, Huet と Lang[6] による簡単化・模倣・射影を用いた完全かつ健全なアルゴリズムが存在する. 一方, マッチングが成功か否かを判定するスキーママッチング問題は一般に NP 完全であり [4], かつ mgu は存在せず複数の極大な単一化代入が存在することが知られている. この複雑さは個体自由変数と束縛変数の存在にも依存し, 個体自由変数が Φ には含まれない場合には, このマッチング問題は多項式時間で解ける [8]. しかしこの場合でも, マッチングが成功するような全ての代入を抽出しようとすると, 一般に指数時間かかってしまう. そこで本稿では, LK 類推証明システムを構築する観点から, 変数の存在位置を先に調べる pre-checking を用いた最適な代入を抽出するアルゴリズムを提案する. 最後に, アルゴリズムの実験結果について報告し, アルゴリズムの有用性を示す.

2 スキーママッチング

型の有限集合 T_0 を仮定し, その要素を**基本型**という. T_0 に対し, 型の集合 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 $\tau_1 \times \cdots \times \tau_n \to \mu \in T$. $C \cap V = \phi$ となる

可算集合 C,V に対して, C,V の要素をそれぞれ**定数**, **変数**という. 本稿では, すべての $d \in C \cup V$ に対して, d に対応する型 $\tau(d) = \sigma \in T$ が存在すると仮定する. これを $d_i : \sigma$ と書き, σ と型付けされた項という.

定義 1 型付項の集合 T を以下のように帰納的に定義する.

(1) $t \in C \cup V$ かつ $\tau(t) \in T_0$ のとき, $t \in T$ である.

(2) (適用) $\tau(d) = \tau_1 \times \cdots \times \tau_n \to \mu$ なる $d \in T$ と $\tau(t_i) = \tau_i$ なる $t_i \in T$ $(1 \leq i \leq n)$ に対して, $d(t_1, \cdots, t_n) \in T$ であり, $\tau(d(t_1, \cdots, t_n)) = \mu$ である.

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

型 σ の**階数** $ord(\sigma)$ を, $\sigma \in T_0$ のとき 1, $\sigma = \sigma_1 \to \sigma_2$ のとき, $max\{ord(\sigma_1) + 1, ord(\sigma_2)\}$ と定義する. $ord(\sigma) = 2$ のとき, $f : \sigma$ と型付けされる項を**2階の項**という. 本稿では, 以下 σ は $\sigma = \tau_1 \times \cdots \times \sigma_n \to \mu$, $\tau_i, \mu \in T_0$ の形であると仮定する. 特に, 2階の項に含まれる変数 $v, \mu \in V$ は $ord(\tau(\mu)) = 1$ または 2 となることに注意する.

$o \in T_0$ を**論理型**とする. また, $\wedge, \vee, \supset, \neg \in C$ とし, $\tau(\wedge) = \tau(\vee) = \tau(\supset) = o \times o \to o$ かつ $\tau(\neg) = o \to o$ とする. 特に限量子 $\forall$, $\exists$ については, 論理的な意味付けをせず, 構文的に整合するように型付けを行う. $Q \in \{\forall, \exists\}$ と $x \in V$ に対して, $Qx.$ を型 $\tau(Qx.) = o \to o$ の定数と捉え, $\tau(\varphi) = o$, $\tau(x) \in T_0$ かつ $\tau(x) \neq o$ ならば $\tau(Qx.\varphi) = o$ とする. このとき, $\tau(t) = o$ となる項 t を**論理式**という. 一階論理式と同様に, 限量子で束縛されてない個体変数を t の**個体自由変数**という. $\wedge(\phi, \varphi)$ を $\phi \wedge \varphi$, $\neg(\phi)$ を $\neg\phi$, $\forall x.(\phi)$ を $\forall x.\phi$, などと表す.

$\tau_i \neq o$ $(1 \leq i \leq n)$ とする. このとき, $\tau(v) = \tau_1 \times \cdots \times \tau_n \to o$ となる変数 $v \in V$ を**述語変数**といい, $\tau(v) = \tau_1 \times \cdots \times \tau_n \to \mu$ かつ $\mu \neq o$ となる変数 $v \in V$ を**関数変数**という. 述語変数と関数変数をあわせて**スキーマ変数**といい, スキーマ変数を含む論理式を**スキーマ**という. 特に, 述語変数を含み関数変数を含まない論理式を**述語スキーマ**といい, 関数変数を含み述語変数を含まない論理式を**項スキーマ**という. 述語変数も関数変数も含まない論理式は**一階論理式**になる. また, 個体自由変数を含まない一階論理式を**一階閉論理式**という. 以後, 定数を $a, b, c, \cdots$, 個体自由変数を $x, y, z, \cdots$, 関数定数を $f, g, h, \cdots$, 関数変数を $F, G, H, \cdots$, 述語変数を $P, Q, R, \cdots$ で表す. また, 本稿では類推システムへの応用の観点より, 個体自由変数を含まないスキーマのみを考える.

次に, 代入を定義する. まず, **代入項**を以下のように帰納的に定義する:

(1) 項は代入項である.

(2) $\tau(t) = \mu \in T_0$ となる項 t と $\tau(v_i) = \tau_i$ となる変数 $v_i \in V (1 \leq i \leq n)$ に対して, $t' = \lambda v_1 \cdots v_n.t$ は代入項であり, $\tau(v_i) = \tau_1 \times \cdots \times \tau_n \to \mu$ である.

$\tau(v_i) = \tau(t_i)$ となる変数 v_i と代入項 $t_i (1 \leq i \leq m)$ に対して $\{t_1/v_1, \cdots, t_m/v_m\}$ を**代入**という.

定義 2 項 t と代入 θ に対し, $t\theta$ を以下のように定義する:

(1) $t = c$ かつ $c \in C$ のとき, $t\theta = c$.

(2) $t = x$ かつ $x \in V$ 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) $t = f(t_1, \cdots, t_n)$ $(f \in C)$ のとき, $t\theta = f(t_1\theta, \cdots, t_n\theta)$.

(4) $t = F(t_1, \cdots, t_n)$ $(F \in V)$ 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.

また, スキーマ t の**頭部** $\mathrm{head}(t)$ を以下のように定義する:

(1) $t \in C, V$ のとき, $\mathrm{head}(t) = t$

(2) $t = f(t_1, \cdots, t_n)$ のとき, $\mathrm{head}(t) = f$

スキーママッチングとは, スキーマ Φ と一階閉論理式 φ の対 (マッチング対) の集合 $E = \{\langle \Phi_i, \varphi_i \rangle \mid 1 \leq i \leq n\}$ に対して, 任意の各 i に対して $\Phi_i\theta = \varphi_i$ をみたす代入 θ を求める操作である. ここで, マッチング対の集合 E のことを**スキーママッチング表現** (または簡単に表現) とよぶ. 代入 θ を**マッチング代入**といい, マッチング代入が存在するか否かを判定する問題を**スキーママッチング問題**という. このスキーママッチングは二階マッチングの一種である [8] ので, スキーママッチングには, Huet と Lang による簡単化・模倣・射影を用いた, 完全かつ健全なアルゴリズムが適用できる [6]. スキーママッチングの書き換え規則には, Huet の手続きの簡単化・模倣に対し, 限量子に関する処理を加える必要がある. 本稿では, 簡単化において $Qx.t(Q \in \{\forall, \exists\})$ を t に書き換え, t に出現する x を**論理束縛記号** w という特別な個体定数で書き換える. また, $Qx.t$ に対する模倣は x を後で使用できるように述語変数の引数を一つ増やすことで実現する. この手続きを以下に示す.

定義 3 スキーママッチングの簡単化, 模倣, 射影の三つの書き換え規則を次のように定義する.

1. 簡単化:
   - $\{\langle c, c\rangle\} \cup E \Rightarrow E$ ($c \in C$ または Cが論理束縛記号)
   - $\{\langle f(s_1, \cdots, s_n), f(t_1, \cdots, t_n)\rangle\} \cup E \Rightarrow \{\langle s_1, t_1\rangle, \cdots, \langle s_n, t_n\rangle\} \cup E$ ($n \geq 1, f \in C$ または fが束縛変数)
   - $\{\langle Qx.\Phi, Qy.\varphi\rangle\} \cup E \Rightarrow (E - \{\langle Qx.\Phi, Qy.\varphi\rangle\}) \cup \{\langle \Phi\{w/x\}, \varphi\{w/y\}\rangle\}$ ($Q \in \{\forall, \exists\}$)
2. E の F に関する模倣:
   - $E \Rightarrow E\{\lambda v_1 \cdots v_n.c/F\}$ ($\langle F(s_1, \cdots, s_n), c\rangle \in E$ かつ $c \neq w$)
   - $E \Rightarrow E\{\lambda v_1 \cdots v_n.f(H_1(v_1, \cdots, v_n), \cdots, H_m(v_1, \cdots, v_n))/F\}$
     ($\langle F(s_1, \cdots, s_n), f(t_1, \cdots, t_m)\rangle \in E$ ($n \geq 0, m \geq 1$))
   - $E \Rightarrow E\{\lambda v_1 \cdots v_n.Qx.P(v_1, \cdots, v_n, x)/F\}$ ($\langle F(s_1, \cdots, s_n), Qx.p(t_1, \cdots, t_m)\rangle \in E$ ($Q \in \{\forall, \exists\}$))
3. E の F に関する射影:
   $E \Rightarrow E\{\lambda v_1 \cdots v_n.v_i/F\}$ ($\langle F(s_1, \cdots, s_n), t\rangle \in E$ ($n \geq 0$, $1 \leq i \leq n$))

スキーママッチングは, スキーマ変数が述語変数か関数変数かの違いにより, **述語マッチング**と**項マッチング**の二段階に分けて考えることができる. 述語マッチングとは, 述語間のマッチングであり, 簡単化と模倣だけが適用できる (型の条件によって射影は適用されないことに注意する). スキーママッチング表現は, 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である.

3 Pre-checking を用いた代入抽出アルゴリズム

類推証明では閉論理式の証明を扱うため, 自由変数はすべて束縛されていると仮定してよい. そこで, スキーマは述語変数しか含まないと仮定できる. このとき, スキーママッチングの決定問題は多項式時間で解けることが知られている [8] が, 全ての解を求めようとすると一般には指数時間かかる. ここで, 述語マッチングの解である項表現は一意であるため, 解を求めるときの困難さは項マッチングによるものであるといえる. そこで本稿では, 以下項マッチングにおける代入抽出について議論する.

類推証明では, 全てのマッチング代入を求める必要はなく, 類似性判定の意味で最適なものを求めればよい. 本稿ではこの制限の下で, スキーママッチング特有の効率的な解を抽出するためのアルゴリズムを考える.

スキーマの定義より述語マッチングで得られる項表現の関数変数は, 引数に関数変数を持たない. そこで, スキーマ表現 E を $\{\langle F(s_1^i, \cdots, s_n^i), f(t_1^i, \cdots, t_m^i)\rangle \mid i \in I\}$ とすると, s_j^i と $f(t_1^i, \cdots, t_m^i)$ が同一であるかどうかを判定することで, F に関する第 j 引数の射影によって E がマッチング可能かどうか判定することができる. これより, 各 $f(t_1^i, \cdots, t_m^i)$ に射影によるマッチング可能性のラベルを割り当てることができる. また, 模倣と簡単化により, E は $\{\langle H_j(s_1^i, \cdots, s_n^i), t_j^i\rangle \mid i \in I, 1 \leq j \leq m\}$ となる. つまり, $f(t_1^i, \cdots, t_m^i)$ は部分項 $t_1^i, \cdots, t_m^i$ に分解される. H_j は t_j^i にしか依存しないことから, $f(t_1^i, \cdots, t_m^i)$ の部分項 t_j^i にも同様に射影によるマッチン

グ可能性のラベルを割り当てることができる．以下にラベル割り当てについて定義する．なお，* は射影しか適用できないことを表す．

定義 4 s をスキーマ項 $H(s_1,\cdots,s_n)$, t を一階閉論理式とする．また，$P(t)=\{i \mid t=s_i(1\leq i\leq n)\}$ とする．このとき，t の s に関するラベル付けされた項 $T(s,t)$ を以下のように定義する:

(1) t が論理束縛記号のとき，$T(s,w)=*P(w)$

(2) t が定数 c のとき，$T(s,c)=cP(c)$

(3) $t=f(t_1,\cdots,t_m)$ であり，t_i の s に関するラベル付けされた項が $T_i(s,t_i)$ のとき，

$T(s,t)=fP(t)(T_1(s,t_1),\cdots,T_n(s,t_n))$

本稿では，定義 4 のように各 s_i の t における射影可能位置をあらかじめ検査する (pre-checking) ことにより効率化を図る．

次に，ラベル付けを行う際の戦略について述べる．t の木表現において，親子の関係にある射影可能な位置が存在する場合には，親の位置をラベル付けする，これは t の最外位置を選択することになる．これを**最外射影優先戦略**とよぶ．pre-checking においてラベル付けされなかった項，つまり，射影が行えない位置には模倣を行う．

Huet の方法に基づく探索木生成アルゴリズムは，射影可能な位置があればそれら全てにラベル付けを行う．しかし，表 1のように，射影できる位置が多数存在する場合には効率化が図れない．そこで，代表元を最左でラベル付けを行うように改良を加える．つまり，最外射影位置に対して，複数の $s_i=s_j$ $(1\leq i\leq j\leq n)$ となる射影が可能な場合には，最左である s_i を選択する．これを**最左最外射影優先戦略**とよぶ．

また，与えられたスキーママッチング表現に同じスキーマ変数が複数個存在する場合，射影可能性に制約を加える必要がある．以下は同じスキーマ変数を持つマッチング対に対し行う処理の概要である．

(1) 各マッチング対に対し定義 4 に従ってラベル付けする．

(2) 共通木をつくる．成分が異なる節点は射影節点とし，* 印を付ける．

(3) 各射影成分の共通成分をとる．複数の共通成分が存在する場合，最左の成分を選択する．

(4) 木表現に対し，最外戦略を適用しラベル付けをする．

模倣優先戦略を適用する場合には * でない限り模倣できるのでそれを利用する．図 1は，スキーママッチング表現に同じスキーマ変数を複数含む場合の pre-checking の例である．

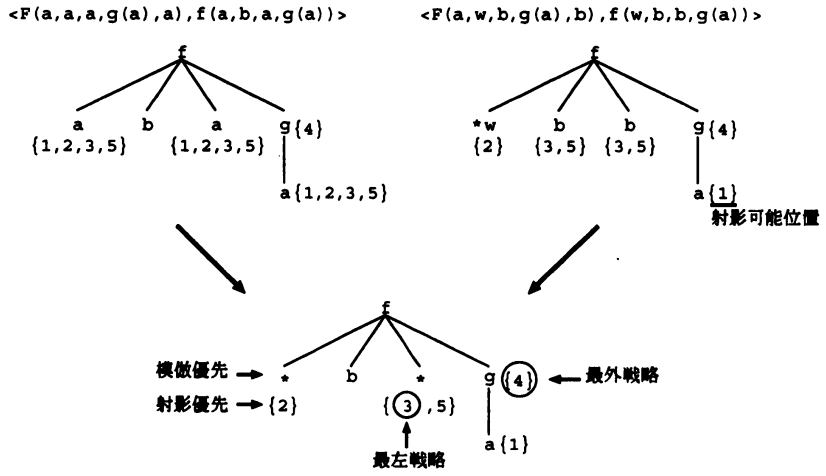

図 1: Pre-checking

射影は，元のスキーマの構文情報を保存するため，射影優先戦略を適用することにより，類推システムで扱う

```
match(s,t) /* 探索木を生成しないスキーママッチングアルゴリズム */
E:={⟨s,t⟩}; /* 探索木を生成する場合には E は木構造で表現 */
if |V_b(s)| > |V_b(t)| then output Fail;
else
  while E に自明でない対が存在 do
    if E に述語変数または述語定数が存在する then
      E := E の述語マッチング [8]; /* このとき E = {⟨H_i(s^i_1,...,s^i_{l_i}), t_i⟩ | 1 ≤ i ≤ n} の形になる */
    else /* 項マッチング [8] */
      for E の全てのマッチング対に対して do /* W に関する pre-checking */
        if |E_H| > 1 then
          各マッチング対 {⟨H(s_1,···,s_n), t_k⟩ | 1 ≤ k ≤ n} の木構造を作成し, 射影可能性のラベル付け;
          P := 共通木を作成し最左最外射影可能位置にラベル付け;
        else P := ラベル付け(H);
            /* このときラベル付けは最左で同値類の代表元を選択 */
      for i = 1 to n do
        for j = 1 to l_i do
          if j ∈ P then E := 射影(H_i, j); /* ラベル付けされた項のとき射影 */
          else E := 模倣(H_i, t_i); /* ラベル付けされた項を取り出すまで模倣 */
          簡単化 (E);
```

図 2: Pre-checking を用いたスキーママッチング

のに適した代入を取り出すことができる. また, 項は個体自由変数を含まないため, 各マッチング対に対して独立に得られた代入は, それらの和集合をとっても代入としての条件を損なうことはない. 図 2は, pre-checking を行う際, 最左最外戦略の下でのスキーママッチングアルゴリズムである. ここで, $|V_b(s)|$ は s 中の束縛変数の種類の数を表す. また, $E_H = \{\langle s,t\rangle \mid \mathrm{head}(s) = H\}$ とし, $|E_H|$ を E 中の H の数と定義する.

最左最外射影優先戦略においては, 唯一のマッチング代入が求まり, マッチング可能なら常にマッチング代入が求まるため, 次の命題 1 が成り立つ. なお, マッチング可能位置の探索は, 個体自由変数を含まないクラスにおいて多項式時間で調べることができる [8].

命題 1 E をスキーママッチング表現とする. アルゴリズム *match*は, E がマッチング可能ならば唯一のマッチング代入を出力する (解の抽出に関する *match*の完全性). また, 求めたマッチング代入は, E の正しいマッチング代入になっている (解の健全性).

4 実装結果

表 1は, [8] の完全なスキーママッチングアルゴリズム, 探索木を生成する pre-checking アルゴリズムおよび図 2のアルゴリズムを StandardML で実装し, マッチング代入を求める実験をした結果である. 表の計算時間はマッチング代入を求めるのにかかる時間を表す.

結果 1 は pre-checking によって効率が上がることを表している. 一方, 結果 2 では生成される探索木が小さいために, pre-checking のオーバーヘッドの分だけ Al2 が遅くなる. しかし, Al3 については探索木を生成せずに代入を出力するため, オーバーヘッドに関係なく速くなる. 結果 3 はマッチング不可能な例であるが, Al1 は探索木を全て生成してマッチング不可能ということを返すため, Al2 および Al3 の方が速くなっている. 結果 4 と 5 にある×はメモリが不足し, 処理が途中で停止してしまうことを表す. Al3 は同値類の代表元しかラベル付

表 1: 実装結果

	マッチング対	計算時間 (msec)		
		Al1	Al2	Al3
1	$\langle P(a), p(a) \wedge q(a) \vee r(b) \rangle$	13.87	6.85	4.01
2	$\langle \forall x.P(a,x), \forall z.\neg(\forall y.p(z,y)) \rangle$	6.03	6.28	3.73
3	$\langle \forall xyz.P(x,y,z),$ $\forall yz.(p(z,y) \wedge q(a,y) \vee r(z)) \rangle$	1.31	0.45	0.45
4	$\langle P(a,a,a,a,a), p(f(a,a,a,a,a,a,a,a)) \rangle$	×	34000	2.24
5	$\langle P(a,a,a,a,a), p(f(a,a,a,a,a,a,a,a,a)) \rangle$	×	×	2.37

Al1 : Huet のアルゴリズム
Al2 : Precheck アルゴリズム (探索木あり)
Al3 : Precheck アルゴリズム (探索木なし)

けしないため, 代入を 1 つしか出力しない. また, 探索木を生成せず, メモリを殆ど必要としないため, 表のような結果が得られた.

5 まとめ

本稿では, pre-checking を用いて, 射影優先戦略に基づいたスキーママッチングアルゴリズムを定式化し実装した. 実装結果から, 個体自由変数を仮定しない類推処理に導入するアルゴリズムとしてはかなり有用なものだといえる.

今後の課題として, 本稿の pre-checking を述語マッチングに拡張して, 同様のヒューリスティックを導入し実装することが挙げられる. さらに, 本稿の pre-checking の手法は個体自由変数を許した場合にも適用できる. 今後これらの観点から拡張を行う予定である.

参考文献

[1] Brock, B., Cooper, S., Pierce, W.: *Analogical reasoning and proof discovery*, LNCS **310**, 454-468, 1988.

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

[3] Harao, M.: *Proof discovery in LK system by Analogy*, Proc. Asian'97, LNCS **1345**, 197-211, 1995.

[4] Hirata, K., Yamada, K., Harao, M.: *Tractable and intractable second-order matching problems*, Proc. COCOON'99, LNCS **1627**, 432-441, 1999.

[5] Huet, G. P.: *A unification algorithm for typed λ-calculus*, Theoretical Computer Science **1**, 27-57, 1978.

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

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

[8] 山田, 平田, 原尾: スキーママッチングとその計算量, 電子情報通信学会 **J82-A**, 1-9, 1999.