Second-Order Schema Matching Based on Projection Point Labeling*

Keizo Yamada, Kouichi Hirata, and Masateru Harao

Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
{yamada,hirata,harao}@ai.kyutech.ac.jp

Summary. In this paper, we propose a second-order schema matching algorithm based on a projection point labeling. This algorithm provides us to construct some suitable matchers efficiently.

1 Introduction

An expression which exhibits abstracted or generalized knowledge is called a *schema*, and the knowledge processing which deals with such a schema as a meta knowledge is called *schema-guided* [2]. The schema-guided knowledge processing is known to be useful for automated program synthesis [2], program transformation [7], analogical reasoning [3] and so on. It is important for the schema-guided knowledge processing to design the matching procedure between a schema and a given problem, which is called a *schema matching*. In this paper, we regard a schema as a second-order formula without quantifiers on predicate and function variables.

Since we can formulate both a schema and a first-order closed formula representing a problem as second-order λ-terms, the schema matching is regarded as a kind of a *second-order matching*, and is known to be intractable in general [1]. On the other hand, our previous work [5] has given a sharp characterization between a tractable and an intractable second-order matching. In particular, we have shown that the schema matching without free individual variables is tractable, and designed a schema matching algorithm by extending the transformations introduced by Huet and Lang [6, 7] and by dividing into two stages, a *predicate* and a *term* matching [4, 8].

In this paper, we divide the schema matching procedure into two processes, a *pre-process* and a *projection position labeling*, instead of the above division. The pre-process consists of a simplification [6, 7]. For an expression E as a set of pairs of schemata and first-order closed formulas, the result applying a pre-process to E is an expression of the form $\{\langle P_i(t^i_1, \dots, t^i_{n_i}), \varphi_i\rangle \mid i \in I\}$, where P_i is a predicate variable and φ_i is a first-order closed formula. The projection position labeling labels to the symbols of a formula φ_i to which a projection is applicable [6, 7]. Then, we determine

* This work is partially supported by Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (B) 13558036 and 11558040.

whether or not E is matchable, by extracting a common part of labeled formulas for each $i \in I$. The present schema matching is more suitable for the implementation by procedural programming languages such as Java than the previous one [4,8].

2 Preliminaries

Let IC, IV, FC, PC, and PV be a set of *individual constants* (denoted by $a, b, c, \ldots$), a set of *individual variables* (denoted by $x, y, x, \ldots$), a set of *function constants* (denoted by $f, g, h, \ldots$), a set of *predicate constants* (denoted by $p, q, r, \ldots$), and a set of *predicate variables* (denoted by $P, Q, R, \ldots$), respectively. We deal with no function variables explicitly, by the definition of schemata denoted below. Throughout of this paper, we assume a set of elementary types containing the Boolean type o. Furthermore, we assume that each $d \in IC \cup IV \cup FC \cup PC \cup PV$ has a type denoted by $\tau(d)$. Here, each $d \in IC \cup IV$ has an elementary type not equal to o, each $d \in FC$ has a type $\mu_1 \times \cdots \times \mu_n \to \mu$ where neither μ_i nor μ is o, and each $d \in PC \cup PV$ has a type $\mu_1 \times \cdots \times \mu_n \to o$.

Terms are defined as usual [1]. Here, we assume that each term contains no λ *abstraction*. The *size* of a term t is the number of occurrences of all symbols in t and is denoted by $|t|$.

In this paper, we deal with the logical connectives $\wedge, \vee, \supset$ as predicate constants satisfying that $\tau(\wedge) = \tau(\vee) = \tau(\supset) = o \times o \to o$ and $\tau(\neg) = o \to o$. For a quantifier Q ($Q \in \{\forall, \exists\}$) we also deal with $Qx.$ as the predicate constant satisfying that $\tau(Qx.) = o \to o$. If $\tau(\varphi) = o$ and $\tau(x) \neq o$, then $Qx.\varphi$ has the type o.

We assume that a *schema* is a term containing predicate variables and neither function nor free individual variables, a *first-order formula* is a term containing neither predicate variables nor function variables, and a *first-order closed formula* is a first-order formula containing no individual free variables (or a *formula* in short). Here, *bound* and *free* variables (with respect to quantifiers) are defined as usual. In the following, we denote schemata and formulas by $\Phi, \Psi, \ldots$ and $\varphi, \psi, \ldots$, respectively. A *head* of a schema Φ is a left-most symbol of Φ and denoted by $\mathrm{hd}(\Phi)$.

In contrast to the terms, a *substitution-term* is just a λ-term [6,7]. For a substitution-term $\lambda v_1 \cdots v_n.t$ and a term t_i ($1 \leq i \leq n$), $t[v_1 := t_1, \ldots, v_n := t_n]$ denotes the substitution-term replacing a variable v_i in t to t_i simultaneously, which is the same expression $(\lambda v_1 \cdots v_n.t)t_1 \cdots t_n$ in λ-calculus. For a substitution-term t_i and an individual variable v_i ($1 \leq i \leq m$) such that t_i does not contain v_i as free variables, $\{t_1/v_1, \ldots, t_m/v_m\}$ is called a *substitution*.

For a term t and a substitution θ, $t\theta$ is defined inductively as follows:

1. If $t = c$, then $t\theta = c$.
2. If $t = x$ and $t'/x \in \theta$, then $t\theta = t'$; otherwise $t\theta = x$.
3. If $t = f(\overline{t_n})$ and $f \in FC \cup PC$, then $t\theta = f(\overline{t_n\theta})$.
4. If $t = P(\overline{t_n})$, $P \in PV$ and $\lambda\overline{v_n}.t'/P \in \theta$, then $t\theta = t'[v_1 := t_1\theta, \ldots, v_n := t_n\theta]$; otherwise $t\theta = P(\overline{t_n\theta})$.
5. If $t = Qx.t'$ and $Q \in \{\forall, \exists\}$, then $t\theta = Qy.((t'\{y/x\})\theta)$, where y is a new variable.

A finite set of pairs of schemata and formulas is called an *expression*. An expression of the form $\{\langle P_i(t_1^i,\dots,t_{n_i}^i),\varphi_i\rangle \mid i\in I\}$ is called a *reduced expression*, where P_i is a predicate variable and φ_i is a formula. For an expression $E=\{\langle \Phi_i,\varphi_i\rangle \mid i\in I\}$, the *size* of E, denoted by $|E|$, is defined as $\sum_{i\in I}(|\Phi_i|+|\varphi_i|)$.

Let E be an expression $\{\langle \Phi_i,\varphi_i\rangle \mid i\in I\}$. A substitution θ such that $\Phi_i\theta=\varphi_i$ for all $i\in I$ is called a *matcher* of E. The *schema matching* for E is a procedure to find a matcher of E. If there is a matcher of E, then E is called *matchable*.

3 Schema Matching Based on Projection Point Labeling

For an expression E, our schema matching first applies the following *pre-process* to E until the reduced expression is obtained:

1. $\{\langle p(\overline{s_n}),p(\overline{t_n})\rangle\}\cup E \Rightarrow \{\langle s_i,t_i\rangle \mid 1\le i\le n\}\cup E$, where $p\in IC\cup FC\cup PC\cup\{\neg,\wedge,\vee,\supset\}$.
2. $\{\langle Qx.\Phi,Qy.\varphi\rangle\}\cup E\Rightarrow (E-\{\langle Qx.\Phi,Qy.\varphi\rangle\})\cup\{\langle \Phi\{w/x\},\varphi\{w/y\}\rangle\}$, where $Q\in\{\forall,\exists\}$.

Here, w is a special individual constant (possibly subscribed) representing the bound variables in an expression. The pre-process is corresponding to a *simplification* [6,7]. Then, we can show the following theorem.

Theorem 1. *For an expression E, let E' be a reduced expression obtained by applying the pre-process to E. Then, E is matchable iff E' is matchable.*

Let E be a reduced expression $\{\langle P_i(\Phi_1^i,\dots,\Phi_{n_i}^i),\varphi_i\rangle \mid i\in I\}$ obtained by applying the pre-process. Then, our schema matching applies the *projection position labeling* to E. Intuitively, each label presents the positions to which a projection can be applicable. Note that a projection cannot be applicable to all individual variables in a reduced expression, because they are always bound.

Definition 1. Let Φ be a schema $P(\Phi_1,\dots,\Phi_n)$, φ a formula, and $\rho(\varphi)$ a set $\{i\mid \varphi=\Phi_i,1\le i\le n\}$. Then, a *judgment term* $T(\Phi,\varphi)$ of φ for Φ is a labeled term defined inductively as follows:

1. $T(\Phi,w)=*^{\rho(w)}$, $T(\Phi,c)=c^{\rho(c)}$ $(c\in IC)$, and $T(\Phi,x)=x^{\emptyset}$ $(x\in IV)$.
2. If $\varphi=f(\varphi_1,\dots,\varphi_m)$, $f\in FC$ and $T(\Phi,\varphi_i)$ is a judgment term of φ_i for Φ $(1\le i\le m)$, then $T(\Phi,\varphi)=f^{\rho(t)}(T(\Phi,\varphi_1),\dots,T(\Phi,\varphi_m))$.
3. If $t=p(\varphi_1,\dots,\varphi_m)$, $p\in PC\cup\{\neg,\wedge,\vee,\supset\}$ and $T(\Phi,\varphi_i)$ is a judgment term of φ_i for Φ $(1\le i\le m)$, then $T(\Phi,\varphi)=p^{\emptyset}(T(\Phi,\varphi_1),\dots,T(\Phi,\varphi_m))$.
4. If $t=Qx.\varphi_1$ $(Q\in\{\forall,\exists\})$ and $T(\Phi,\varphi_1)$ is a judgment term of φ_1 for Φ, then $T(\Phi,\varphi)=Qx^{\emptyset}.T(\Phi,\varphi_1)$.

Here, $*$ is a new symbol to which no imitations can be applicable.

Furthermore, we introduce a *common judgment term* as a common part of judgment terms.

Definition 2. For a reduced expression $\{\langle \Phi_i, \varphi_i \rangle \mid i \in I\}$ such that $\mathrm{hd}(\Phi_i) = P$, let $T(\Phi_i, \varphi_i)$ be a judgment term of φ_i for Φ_i. Then, a *common judgment term* $\sqcap_{i \in I} T(\Phi_i, \varphi_i)$ of $\{T(\Phi_i, \varphi_i) \mid i \in I\}$ is defined inductively as follows:

1. If $T(\Phi_i, \varphi_i) = c^{\rho_i}$ and $c \in IC \cup IV \cup \{*\}$ for each $i \in I$, then:
$$\sqcap_{i \in I} T(\Phi_i, \varphi_i) = c^{\cap_{i \in I} \rho_i}.$$
2. If $T(\Phi_i, \varphi_i) = f^{\rho_i}(\varphi_1^i, \dots, \varphi_m^i)$ and $f \in FC$ for each $i \in I$, then:
$$\sqcap_{i \in I} T(\Phi_i, \varphi_i) = f^{\cap_{i \in I} \rho_i}(\sqcap_{i \in I} T(\Phi_i, \varphi_1^i), \dots, \sqcap_{i \in I} T(\Phi_i, \varphi_m^i)).$$
3. If $T(\Phi_i, \varphi_i) = p^{\emptyset}(\varphi_1^i, \dots, \varphi_m^i)$ and $p \in PC \cup \{\neg, \wedge, \vee, \supset\}$ for each $i \in I$, then:
$$\sqcap_{i \in I} T(\Phi_i, \varphi_i) = p^{\emptyset}(\sqcap_{i \in I} T(\Phi_i, \varphi_1^i), \dots, \sqcap_{i \in I} T(\Phi_i, \varphi_m^i)).$$
4. If $T(\Phi_i, \varphi_i) = Qx_i^{\emptyset}.\varphi_i'$ and $Q \in \{\forall, \exists\}$ for each $i \in I$, then:
$$\sqcap_{i \in I} T(\Phi_i, \varphi_i) = Qx^{\emptyset}.\sqcap_{i \in I} (T(\Phi_i, \varphi_i')\{x/x_i\}).$$
5. If there exist $k, j \in I$ such that $\mathrm{hd}(T(\Phi_k, \varphi_k)) \neq \mathrm{hd}(T(\Phi_j, \varphi_j))$, then:
$$\sqcap_{i \in I} T(\Phi_i, \varphi_i) = *^{\cap_{i \in I} \rho_i}.$$

Finally, we introduce a *reduced judgment term*.

Definition 3. For a reduced expression $E = \{\langle \Phi_i, \varphi_i \rangle \mid i \in I\}$, a *reduced judgment term* $\sqcap E$ of E is a labeled term obtained by applying the following rule to $\sqcap_{i \in I} T(\Phi_i, \varphi_i)$ as possible:

> For a term t in $\sqcap_{i \in I} T(\Phi_i, \varphi_i)$, if there exists a j $(1 \leq j \leq m)$ such that $t = f^{\emptyset}(t_1, \dots, t_m)$ and $t_j = *^{\emptyset}$, then replace t with $*^{\emptyset}$ in $\sqcap_{i \in I} T(\Phi_i, \varphi_i)$.

As similar as our previous work [8], the following theorem holds.

Theorem 2. *Let E be a reduced expression. Then, $\sqcap E$ can be constructed in $O(|E|^2)$ time. Furthermore, E is matchable iff $\sqcap E \neq *^{\emptyset}$.*

Example 1. Consider the following expression E_1:

$$E_1 = \left\{ \begin{array}{l} \langle \forall x_1.((P(x_1, f(x_1)) \wedge P(x_1, g(x_1))) \supset P(f(x_1), x_1)), \\ \ \forall x_2.((\exists z_1.p(z_1, f(x_2)) \wedge \exists z_2.p(z_2, g(x_2))) \supset \exists z_3.p(z_3, f(x_2)))\rangle \end{array} \right\}.$$

At first, we apply the pre-process to E_1 as follows.

$$\begin{array}{rl} E_1 \Rightarrow & \left\{ \begin{array}{l} \langle (P(w, f(w)) \wedge P(w, g(w))) \supset P(f(w), w), \\ \ (\exists z_1.p(z_1, f(w)) \wedge \exists z_2.p(z_2, g(w))) \supset \exists z_3.p(z_3, f(w)))\rangle \end{array} \right\} \\ \Rightarrow & \left\{ \begin{array}{l} \langle P(w, f(w)) \wedge P(w, g(w)),\ \exists z_1.p(z_1, f(w)) \wedge \exists z_2.p(z_2, g(w))\rangle, \\ \langle P(f(w), w),\ \exists z_3.p(z_3, f(w))\rangle \end{array} \right\} \\ \Rightarrow & \left\{ \begin{array}{l} \langle P(w, f(w)),\ \exists z_1.p(z_1, f(w))\rangle, \\ \langle P(w, g(w)),\ \exists z_2.p(z_2, g(w))\rangle, \\ \langle P(f(w), w),\ \exists z_3.p(z_3, f(w))\rangle \end{array} \right\} \quad \left(= \left\{ \begin{array}{l} \langle \Phi_1, \varphi_1 \rangle, \\ \langle \Phi_2, \varphi_2 \rangle, \\ \langle \Phi_3, \varphi_3 \rangle \end{array} \right\} = E_1' \right). \end{array}$$

Next, we apply the projection point labeling to E_1'. The judgment terms are:

$$T(\Phi_1,\varphi_1) = \exists z_1^{\emptyset}.p^{\emptyset}(z_1^{\emptyset}, f^{\{2\}}(w^{\{1\}})), T(\Phi_2,\varphi_2) = \exists z_2^{\emptyset}.p^{\emptyset}(z_2^{\emptyset}, g^{\{2\}}(w^{\{1\}})),$$
$$T(\Phi_3,\varphi_3) = \exists z_3^{\emptyset}.p^{\emptyset}(z_3^{\emptyset}, f^{\{1\}}(w^{\{2\}})).$$

By Definition 2, it holds that $\sqcap_{i\in\{1,2,3\}} T(\Phi_i,\varphi_i) = \exists z^{\emptyset}.p^{\emptyset}(z^{\emptyset}, *^{\emptyset})$. By Definition 3, it holds that $\sqcap E_1' = *^{\emptyset}$, which implies that E is not matchable by Theorem 1 and 2. On the other hand, consider the following expression E_2:

$$E_2 = \left\{ \begin{array}{l} \langle \forall x_1(P(x_1,f(x_1)) \wedge P(x_1,g(x_1)), \\ \ \forall x_2(\exists z_1.p(z_1,f(x_2)) \wedge \exists z_2.p(z_2,g(x_2)))\rangle \end{array} \right\}.$$

By the pre-process for E_2, we obtain the following reduced expression E_2'.

$$E_2' = \{\langle \Phi_1,\varphi_1\rangle, \langle \Phi_2,\varphi_2\rangle\}.$$

It holds that $\sqcap_{i\in\{1,2\}} T(\Phi_i,\varphi_i) = \exists z^{\emptyset}.p^{\emptyset}(z^{\emptyset}, *^{\{2\}}) = \sqcap E_2' \neq *^{\emptyset}$, which implies that E_2 is matchable by Theorem 1 and 2. Furthermore, we can extract the matcher of E_2 as $\{\lambda xy.\exists z.p(z,y)/P\}$, by using $\sqcap E_2'$

4 Discussion

We have proposed in this paper a second-order schema matching algorithm which decides matchability in $O(|E|^2)$ time. From the viewpoint of applications, it is important to design an algorithm which does not only decide matchability but also produces matchers efficiently. Since the expression by projection point labeling holds all the information for the applicability of the rules, we are able to design an algorithm which extracts some optimal matcher, by introducing moderate heuristics as for the application of rules. We have also shown that some efficient matcher extracting algorithms are available [8].

References

1. L. D. Baxter, *The complexity of unification*, Doctoral Thesis, Department of Computer Science, University of Waterloo, 1977.
2. P. Flener, *Logic program synthesis from incomplete information* (Kluwer Academic Press, 1995).
3. M. Harao, *Proof discovery in LK system by analogy*, in: Proc. 3rd Asian Computing Science Conf., LNCS **1345** (Springer, 1997) 197–211.
4. M. Harao, K. Yamada, K. Hirata, *Efficient second order predicate matching algorithm*, in: Proc. Korea-Japan Joint Workshop on Algorithm and Computations, 31–39, 1999.
5. K. Hirata, K. Yamada, M. Harao, *Tractable and intractable second-order matching problems*, in: Proc. 5th Annual Intern. Computing and Combinatorics Conf., LNCS **1627** (Springer, 1999) 432–442.
6. G. P. Huet, *A unification algorithm for typed λ-calculus*, Theor. Comput. Sci. **1** (1975) 27–57.
7. G. P. Huet, B. Lang, *Proving and applying program transformations expressed with second-order patterns*, Acta Inform. **11** (1978) 31–55.
8. K. Yamada, K. Hirata, M. Harao, *Schema matching and its complexity*, Trans. IEICE **J82-D-I** (1999) 1307–1316 (in Japanese).