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

Abstract. 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 \{⟨P_i(t_{i1}^{1},\ldots,t_{in}^{i}),\varphi_i⟩ | i \in I⟩, where

* This work is partially supported by Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (B) 13558036 and 11558040. Also this paper will be published in Proc. 15th International Workshop on Unification (UNIF2001) ©Springer-Verlag.
\(P_i\) is a predicate variable and \(\varphi_i\) is a first-order closed formula. The projection position labeling labels to the symbols of a formula \(\varphi_i\) to which a projection is applicable [6, 7]. Then, we determine 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 the paper, we assume a set of elementary types containing the Boolean type \(\text{o}\). Furthermore, we assume that each \(d \in IC \cup IV\) has an elementary type not equal to \(\text{o}\). To express constants satisfying that \(\mu_1 \times \cdots \times \mu_n \to \mu\) where neither \(\mu_1\) nor \(\mu\) is \(\text{o}\), and each \(d \in PC \cup PV\) has a type \(\mu_1 \times \cdots \times \mu_n \to \text{o}\).

Terms are defined as usual [1]. Here, we assume that each term contains no \(\lambda\) 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 \(\land, \lor, \top\) as predicate constants satisfying that \(\tau(\land) = \tau(\lor) = \tau(\top) = \text{o} \times \text{o} \to \text{o}\) and \(\tau(\neg) = \text{o} \to \text{o}\). For a quantifier \(Q\) (\(Q \in \{\forall, \exists\}\)) we also deal with \(Qx.\tau\) as the predicate constant satisfying that \(\tau(Qx.) = \text{o} \to \text{o}\). If \(\tau(\varphi) = \text{o}\) and \(\tau(\varphi) \neq \text{o}\), then \(Qx.\varphi\) has the type \(\text{o}\).

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 \(\Phi\) is a left-most symbol of \(\Phi\) and denoted by \(\text{hd}(\Phi)\).

In contrast to the terms, a substitution-term is just a \(\lambda\)-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 \(\lambda\)-calculus. For a substitution-term \(t_i\) and an individual variable \(v_i (\leq i \leq m)\) 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 \(\theta\), \(t\theta\) 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 \(t = f(t_1, \ldots, t_n)\) and \(f \in FC \cup PC\), then \(t\theta = f(t_1\theta, \ldots, t_n\theta)\).
4. If \( t = P(t_n') \), \( P \in PV \) and \( \lambda \rightarrow \cdot t'/P \in \theta \), then \( \theta = t'[v_1 := t_1 \theta, \ldots, v_n := t_n \theta] \); otherwise \( \theta = P(t_n') \).

5. If \( t = Qx.t' \) and \( Q \in \{ \forall, \exists \} \), then \( \theta = Qy.((t'[y/x])\theta) \), where \( y \) is a new variable.

A finite set of pairs of schemata and formulas is called an expression. An expression of the form \( \{ (P_i(t_{i1}^n), \phi_i) \mid i \in I \} \) is called a reduced expression, where \( P_i \) is a predicate variable and \( \phi_i \) is a formula. For an expression \( E = \{ (\Phi_i, \varphi_i) \mid i \in I \} \), the size of \( E \), denoted by \(|E|\), is defined as \( \sum_{i \in I} (|\Phi_i| + |\varphi_i|) \).

Let \( E \) be an expression \( \{ (\Phi_i, \varphi_i) \mid i \in I \} \). A substitution \( \theta \) such that \( \Phi_i, \varphi_i \) 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. \( \{ (p(t_m), p(t_n)) \} \cup E \Rightarrow \{ (s_i, t_j) \mid 1 \leq i \leq n \} \cup E \), where \( p \in IC \cup FC \cup PC \cup \{ \exists, \land, \lor \} \).
2. \( \{ (Qx.\Phi, Qy.\varphi) \} \cup E \Rightarrow (E - \{ (Qx.\Phi, Qy.\varphi) \}) \cup \{ (\Phi[w/x], \varphi[w/y]) \} \), 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 \( \{ (P_i(\Phi_{i1}^m), \varphi_i) \mid i \in I \} \) 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 \( \Phi \) be a schema \( P(\Phi_1, \ldots, \Phi_n) \), \( \varphi \) a formula, and \( \rho(\varphi) \) a set \( \{ i \mid \varphi = \Phi_i, 1 \leq i \leq n \} \). Then, a judgment term \( T(\Phi, \varphi) \) of \( \varphi \) for \( \Phi \) is a labeled term defined inductively as follows:

1. \( T(\Phi, w) = \ast(\rho(w)), T(\Phi, c) = \rho(c) (c \in IC) \), and \( T(\Phi, x) = x^\emptyset (x \in IV) \).
2. If \( \varphi = f(\varphi_1, \ldots, \varphi_m) \), \( f \in FC \) and \( T(\Phi, \varphi_i) \) is a judgment term of \( \varphi_i \) for \( \Phi \) (\( 1 \leq i \leq m \)), then \( T(\Phi, \varphi) = f^{\rho(t)}(T(\Phi, \varphi_1), \ldots, T(\Phi, \varphi_m)) \).
3. If \( t = p(\varphi_1, \ldots, \varphi_m) \), \( p \in PC \cup \{ \exists, \land, \lor \} \) and \( T(\Phi, \varphi_i) \) is a judgment term of \( \varphi_i \) for \( \Phi \) (\( 1 \leq i \leq m \)), then \( T(\Phi, \varphi) = p^\emptyset(T(\Phi, \varphi_1), \ldots, T(\Phi, \varphi_m)) \).
4. If \( t = Qx.\varphi_1 \) (\( Q \in \{ \forall, \exists \} \)) and \( T(\Phi, \varphi_1) \) is a judgment term of \( \varphi_1 \) for \( \Phi \), then \( T(\Phi, \varphi) = Qx. T(\Phi, \varphi_1) \).
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 \( \{ (\Phi_i, \varphi_i) \mid i \in I \} \) such that \( \text{hd}(\Phi_i) = P_i \), let \( T(\Phi_i, \varphi_i) \) be a judgment term of \( \varphi_i \) for \( \Phi_i \). Then, a common judgment term \( \cap_{i \in I} T(\Phi_i, \varphi_i) \) of \( \{ T(\Phi_i, \varphi_i) \mid i \in I \} \) is defined inductively as follows:

1. If \( T(\Phi_i, \varphi_i) = c^\varphi \) and \( c \in IC \cup IV \cup \{ * \} \) for each \( i \in I \), then:
   \[
   \cap_{i \in I} T(\Phi_i, \varphi_i) = c^\cap_{i \in I} P_i.
   \]
2. If \( T(\Phi_i, \varphi_i) = f^p(\varphi'_1, \ldots, \varphi'_m) \) and \( f \in FC \) for each \( i \in I \), then:
   \[
   \cap_{i \in I} T(\Phi_i, \varphi_i) = f^\cap_{i \in I} P_i(\cap_{i \in I} T(\Phi_i, \varphi'_1), \ldots, \cap_{i \in I} T(\Phi_i, \varphi'_m)).
   \]
3. If \( T(\Phi_i, \varphi_i) = p^\varphi(\varphi'_1, \ldots, \varphi'_m) \) and \( p \in PC \cup \{ \neg, \land, \lor, \} \) for each \( i \in I \), then:
   \[
   \cap_{i \in I} T(\Phi_i, \varphi_i) = p^\cap_{i \in I} P_i(\cap_{i \in I} T(\Phi_i, \varphi'_1), \ldots, \cap_{i \in I} T(\Phi_i, \varphi'_m)).
   \]
4. If \( T(\Phi_i, \varphi_i) = Q^x^\varphi \) and \( Q \in \{ \forall, \exists \} \) for each \( i \in I \), then:
   \[
   \cap_{i \in I} T(\Phi_i, \varphi_i) = Q^x^\cap_{i \in I} P_i \{ T(\Phi_i, \varphi'_i) \{ x \mapsto x \} \}.
   \]
5. If there exist \( k, j \in I \) such that \( \text{hd}(T(\Phi_k, \varphi_k)) \neq \text{hd}(T(\Phi_j, \varphi_j)) \), then:
   \[
   \cap_{i \in I} T(\Phi_i, \varphi_i) = *^\cap_{i \in I} P_i.
   \]

Finally, we introduce a reduced judgment term.

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

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

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

**Theorem 2.** Let \( E \) be a reduced expression. Then, \( \cap E \) can be constructed in \( O(|E|^2) \) time. Furthermore, \( E \) is matchable iff \( \cap E \neq *^\varphi \).

**Example 1.** Consider the following expression \( E_1 \):

\[
E_1 = \left\{ \forall x_1, ((P(x_1, f(x_1)) \land P(x_1, g(x_1))) \lor P(f(x_1), x_1)), \forall x_2, ((\exists z_1, p(z_1, f(x_2)) \land \exists z_2, p(z_2, g(x_2))) \lor \exists z_3, p(z_3, f(x_2))) \right\}.
\]

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

\[
\begin{align*}
E_1 & \Rightarrow \left\{ \{ (P(w, f(w)) \land P(w, g(w))) \land P(f(w), w), \right\} \\
& \quad \left\{ (\exists z_1, p(z_1, f(w)) \land \exists z_2, p(z_2, g(w))) \land \exists z_3, p(z_3, f(w))) \right\} \\
& \Rightarrow \left\{ \{ P(f(w), w), \exists z_2, p(z_2, f(w)) \right\} \\
& \quad \left\{ P(w, g(w)), \exists z_2, p(z_2, g(w)) \right\} \\
& \Rightarrow \left\{ \{ P(w, f(w)), \exists z_2, p(z_2, f(w)) \right\} \\
& \quad \left\{ P(f(w), w), \exists z_3, p(z_3, f(w)) \right\} \right\} = \left\{ \{ \Phi_1, \varphi_1 \}, \{ \Phi_2, \varphi_2 \}, \{ \Phi_3, \varphi_3 \} \right\} = E_1'.
\end{align*}
\]
Next, we apply the projection point labeling to $E'_1$. The judgment terms are:

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

By Definition 2, it holds that $\cap_{i \in \{1, 2, 3\}} T(\Phi_i, \varphi_i) = \exists z_i^\emptyset . p^\emptyset (z_i^\emptyset, *^\emptyset)$. By Definition 3, it holds that $\cap E'_1 = *^\emptyset$, which implies that $E$ is not matchable by Theorem 1 and 2.

On the other hand, consider the following expression $E_2$:

\[ E_2 = \left\{ \langle \forall x_1 (P(x_1, f(x_1)) \land P(x_1, g(x_1)), \langle \forall x_2 (\exists z_1 . p(z_1, f(x_2)) \land \exists z_2 . p(z_2, g(x_2))) \rangle \right\} . \]

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 $\cap_{i \in \{1, 2\}} T(\Phi_i, \varphi_i) = \exists z_i^\emptyset . p^\emptyset (z_i^\emptyset, *^{(i)}) = \cap E'_2 \neq *^\emptyset$, 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 $\cap E'_2$.

### 4 Discussion

We have proposed in this paper a second-order schema matching algorithm which decides matchability in $O(|E'_2|^2)$ 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