Tractable and Intractable Second-Order Matching Problems*

Kouichi Hirata Keizo Yamada Masateru Harao **

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

Abstract. The *second-order matching problem* is the problem of determining, for a finite set $\{\langle t_i, s_i\rangle \mid i \in I\}$ of pairs of a second-order term t_i and a first-order closed term s_i, called a *matching expression*, whether or not there exists a substitution σ such that $t_i\sigma = s_i$ for each $i \in I$. It is well-known that the second-order matching problem is NP-complete. In this paper, we introduce the following restrictions of a matching expression: *k-ary*, *k-fv*, *predicate*, *ground*, and *function-free*. Then, we show that the second-order matching problem is NP-complete for a unary predicate, a unary ground, a ternary function-free predicate, a binary function-free ground, and an 1-fv predicate matching expressions, while it is solvable in polynomial time for a binary function-free predicate, a unary function-free, a k-fv function-free ($k \geq 0$), and a ground predicate matching expressions.

1 Introduction

The *unification problem* is the problem of determining whether or not any two terms possess a common instance. The *matching problem*, on the other hand, is the problem of determining whether or not a term is an instance of another term. Both the unification and the matching play an important role in many research areas, including theorem proving, term rewriting systems, logic and functional programming, database query language, program synthesis, and so on.

The *second-order unification problem* is formulated as the problem of determining, for a finite set $\{\langle t_i, s_i\rangle \mid i \in I\}$ of pairs of second-order terms t_i and s_i, called a *unification expression*, whether or not there exists a substitution σ such that $t_i\sigma = s_i\sigma$ for each $i \in I$. The *second-order matching problem* is the special unification problem that t_i is a second-order term and s_i is a first-order closed term. A unification expression for the second-order matching problem is called

* This paper is published in Proc. of 5th Annual International Computing and Combinatrics Conference (COCOON'99), Lecture Notes in Computer Science 1627, 432–441, 1999 ©Springer-Verlag.

** This work is partially supported by the Japanese Ministry of Education, Grant-in-Aid for Exploratory Research 10878055 and Kayamori Foundation of Informational Science Advancement.

a *matching expression*. The second-order matching has been applied to program synthesis and transformation, schema-guided proof, analogical reasoning, and machine learning [4, 5, 9, 13, 14].

It is well-known that the second-order unification problem is undecidable [12], and various researchers have separated decidable from undecidable unification problems by introducing several restrictions of a unification expression [1, 6, 7, 8, 11, 12, 15, 16, 17]. It is also well-known that the second-order matching problem is NP-complete [2]. Huet and Lang [14] have designed a complete and nonredundant second-order matching algorithm. However, there exist few researches to analyze deeply the complexity of the matching problem. It is one of the reason that the interest of the researchers [3, 5, 9, 13, 14] is rather the matching algorithm than the matching problem itself. In this paper, by introducing the several restrictions of a matching expression, we give a sharp characterization between tractable and intractable second-order matching problems.

A matching expression is called *k-ary* if any function variable in it is at most k-ary, and *k-fv* if it includes at most k distinct function variables. Furthermore, a matching expression is called *predicate* if any argument's term of function variables in it includes no function variables, *ground* if it includes no individual variables, and *function-free* if it includes no function constants.

In this paper, we show that the second-order matching problem is NP-complete for a unary predicate, a unary ground, a ternary function-free predicate, and a binary function-free ground matching expressions, while it is solvable in polynomial time for a binary function-free predicate and a unary function-free matching expressions. We also show that it is NP-complete for an 1-fv predicate matching expression, while it is solvable in polynomial time for a k-fv function-free matching expression for $k \geq 0$. Furthermore, we show that it is solvable in polynomial time for a ground predicate matching expression.

2 Preliminaries

Instead of considering arbitrary second-order languages, we shall restrict our attention to languages containing just simple terms (i.e., terms without variable-binding operators like the λ operator). Throughout of this paper, we deal with the term languages introduced by Goldfarb [12] and Farmer [8].

Let a *term language* L be a quadruple $(\mathrm{IC}_L, \mathrm{IV}_L, \mathrm{FC}_L, \mathrm{FV}_L)$, where IC_L is a set of *individual constants* (denoted by $a, b, c, \cdots$); IV_L is a set of *individual variables* (denoted by $x, y, z, \cdots$); FC_L is a set of *function constants* (denoted by $f, g, h, \cdots$); FV_L is a set of *function variables* (denoted by $F, G, H, \cdots$). Each element of $\mathrm{FC}_L \cup \mathrm{FV}_L$ has a fixed arity ≥ 1, and IC_L, IV_L, FC_L and FV_L are mutually disjoint. We call an element of $\mathrm{IV}_L \cup \mathrm{FV}_L$ a *variable* simply. Let BV_L be an infinite collection $\{w_i\}_{i \geq 1}$ of symbols not in L called *bound variables*.

The *L-terms* and *L^*-terms* are defined inductively by:

1. Each $d \in \mathrm{IC}_L \cup \mathrm{IV}_L$ (*resp.* $\mathrm{IC}_L \cup \mathrm{IV}_L \cup \mathrm{BV}_L$) is an L-term (*resp.* L^*-terms).
2. If $d \in \mathrm{FC}_L \cup \mathrm{FV}_L$ has arity $n \geq 1$ and $t_1, \cdots, t_n$ are L-terms (*resp.* L^*-terms), then $d(t_1, \cdots, t_n)$ is an L-term (*resp.* L^*-terms).

The *rank* of an L^*-term t is the largest n such that w_n occurs in t.

For L^*-terms $t, t_1, \cdots, t_n$, we write $t[t_1, \cdots, t_n]$ for the L^*-term obtained by replacing each occurrence of w_i in t with t_i for all i $(1 \leq i \leq n)$ simultaneously. The *head* of t, denoted by $\mathrm{hd}(t)$, is the outermost symbol occurring in t. An L^*-term is *closed* if it contains no variables.

A *substitution* (in L) is a function σ with a finite domain $\mathrm{dom}(\sigma) \subseteq \mathrm{IV}_L \cup \mathrm{FV}_L$ which maps individual variables to L-terms and n-ary function variables with $n \geq 1$ to L^*-terms of rank $\leq n$. A substitution σ is denoted by $\{s_1/v_1, \cdots, s_m/v_m\}$, where $\mathrm{dom}(\sigma) = \{v_1, \cdots, v_m\}$. Each element s_i/v_i of σ is called a *binding* of σ. The result $t\sigma$ of applying σ to an L^*-term t is defined inductively by:

1. If $t \in \mathrm{IC}_L \cup \mathrm{IV}_L \cup \mathrm{BV}_L$ but $t \notin \mathrm{dom}(\sigma)$, then $t\sigma = t$.
2. If $t = x \in \mathrm{IV}_L$ and $x \in \mathrm{dom}(\sigma)$, then $t\sigma = x\sigma$.
3. If $t = d(t_1, \cdots, t_n)$ $(d \in \mathrm{FC}_L \cup \mathrm{FV}_L)$ but $d \notin \mathrm{dom}(\sigma)$, then $t\sigma = d(t_1\sigma, \cdots, t_n\sigma)$.
4. If $t = F(t_1, \cdots, t_n)$ and $F \in \mathrm{dom}(\sigma)$, then $t\sigma = (F\sigma)[t_1\sigma, \cdots, t_n\sigma]$.

A *matching expression* E (in L) is a finite set $\{\langle t_i, s_i\rangle \mid i \in I\}$, where t_i is an L-term and s_i is a closed L-term for each $i \in I$. For a substitution σ, $E\sigma$ denotes the matching expression $\{\langle t_i\sigma, s_i\rangle \mid i \in I\}$. The *size* of E, denoted by $|E|$, is the number of symbols of L occurring in E. Furthermore, for a function variable F, E_F denotes a matching expression $\{\langle t, s\rangle \in E \mid \mathrm{hd}(t) = F\}$.

A matching expression E is called *matchable* if there exists a substitution σ such that $t_i\sigma = s_i$ for each $i \in I$. Such a σ is called a *matcher* of E.

The *transformation rules* [14] are defined as follows:

1. **simplification**:
   $\{\langle f(t_1, \cdots, t_n), f(s_1, \cdots, s_n)\rangle\} \cup E \Rightarrow \{\langle t_1, s_1\rangle, \cdots, \langle t_n, s_n\rangle\} \cup E \ \ (n \geq 0)$,
2. **projection** (on F): $E \ \Rightarrow \ E\{w_i/F\}$,
   if $\langle F(t_1, \cdots, t_n), s\rangle \in E \ \ (n \geq 1, 1 \leq i \leq n)$,
3. **imitation** (on F): $E \ \Rightarrow E\{f(H_1(w_1, \cdots, w_n), \cdots, H_m(w_1, \cdots, w_n))/F\}$,
   if $\langle F(t_1, \cdots, t_n), f(s_1, \cdots, s_m)\rangle \in E \ \ (n, m \geq 0)$.

Theorem 1 (Huet&Lang [14]). *E is matchable iff $E \Rightarrow^* \emptyset$.*

The *second-order matching problem* is defined as follows:

SECOND-ORDER MATCHING (MATCHING)
INSTANCE: A matching expression E.
QUESTION: Is E matchable?

Theorem 2 (Baxter [2]). MATCHING *is NP-complete.*

In this paper, we reduce the following problem MONOTONE 1-IN-3 3SAT [10] to the restricted problems of MATCHING:

MONOTONE 1-IN-3 3SAT [10]
INSTANCE: A set X of variables and a collection C of monotone 3-clauses (clauses consisting of three positive literals) over X.

QUESTION: Is there a truth assignment to X that makes exactly one literal of each clause in C true?

Hereafter, we refer $X = \{x_1, \cdots, x_n\}$ and $C = \{c_1, \cdots, c_m\}$ to an instance of MONOTONE 1-IN-3 3SAT, where $c_j \in C$ consists of the variables x_1^j, x_2^j and x_3^j.

3 The Restricted Second-Order Matching Problems

Let E be a matching expression. E is *k-ary* if any function variable in E is at most k-ary. E is *k-fv* if E includes at most k distinct function variables. E is *predicate* if any argument's term of function variables in E includes no function variables. E is *ground* if E includes no individual variables. E is *function-free* if E includes no function constants.

We introduce the following restricted problems of MATCHING:

kARY (*resp.* kFV, PRED, GROUND, FFREE) MATCHING
INSTANCE: A k-ary (*resp.* k-fv, predicate, ground, function-free) matching expression E.
QUESTION: Is E matchable?

Theorem 3. UNARYPREDMATCHING *is NP-complete.*

Proof. For each clause $c \in C$, let z_1, z_2 and z_3 be the variables in c. Then, let E_c be the following unary predicate matching expression:

$$E_c = \left\{ \begin{array}{l} \langle F(f(z_3, f(z_2, f(z_1, y)))),\ f(0, f(0, f(1, f(0, f(0, 0))))) \rangle, \\ \langle F(y),\ f(0, f(0, 0)) \rangle \end{array} \right\}.$$

Suppose that c is satisfiable and let (a_1, a_2, a_3) be a truth assignment to (z_1, z_2, z_3) satisfying c, where there exists exactly one index i $(1 \leq i \leq 3)$ such that $a_i = 1$ and $a_l = 0$ $(l \neq i)$. We can construct the matcher σ of E_c as follows:

1. If $(a_1, a_2, a_3) = (1, 0, 0)$, then $\sigma = \{w_1/F, 1/z_1, 0/z_2, 0/z_3, f(0, f(0, 0))/y\}$;
2. If $(a_1, a_2, a_3) = (0, 1, 0)$, then $\sigma = \{f(0, w_1)/F, 0/z_1, 1/z_2, 0/z_3, f(0, 0)/y\}$;
3. If $(a_1, a_2, a_3) = (0, 0, 1)$, then $\sigma = \{f(0, f(0, w_1))/F, 0/z_1, 0/z_2, 1/z_3, 0/y\}$.

Conversely, suppose that E_c is matchable and let σ be a matcher of E_c. Then, σ includes the binding t/F, where t is w_1, $f(0, w_1)$, or $f(0, f(0, w_1))$.

Suppose that $w_1/F \in \sigma$. Since $E_c\{w_1/F\}$ is of the form

$$\{\langle f(z_3, f(z_2, f(z_1, y))), f(0, f(0, f(1, f(0, f(0, 0))))) \rangle, \langle y, f(0, f(0, 0)) \rangle\},$$

and by a simplification, σ includes the bindings $1/z_1$, $0/z_2$ and $0/z_3$.

Suppose that $f(0, w_1)/F \in \sigma$. Since $E_c\{f(0, w_1)/F\}$ is of the form

$$\{\langle f(0, f(z_3, f(z_2, f(z_1, y)))), f(0, f(0, f(1, f(0, f(0, 0))))) \rangle, \langle y, f(0, 0) \rangle\},$$

and by a simplification, σ includes the bindings $0/z_1$, $1/z_2$ and $0/z_3$.

Suppose that $f(0, f(0, w_1))/F \in \sigma$. Since $E_c\{f(0, f(0, w_1))/F\}$ is of the form

$$\{\langle f(0, f(0, f(z_3, f(z_2, f(z_1, y))))), f(0, f(0, f(1, f(0, f(0, 0))))))\rangle, \langle y, 0\rangle\},$$

and by a simplification, σ includes the bindings $0/z_1$, $0/z_2$ and $1/z_3$.

Then, we can construct the truth assignment (a_1, a_2, a_3) to (z_1, z_2, z_3) satisfying c such that $a_i = 1$ if $1/z_i \in \sigma$; $a_i = 0$ if $0/z_i \in \sigma$ $(1 \leq i \leq 3)$. Hence, (a_1, a_2, a_3) satisfies c, where exactly one of a_1, a_2 and a_3 is 1 and others are 0.

For C, let E be the unary predicate matching expression $\bigcup_{j=1}^{m}(E_{c_j}\{F_j(w_1)/F, y_j/y\})$. Then, C is satisfiable by a truth assignment that makes exactly one literal in each clause in C true iff E is matchable. □

Theorem 4. UnaryGroundMatching *is NP-complete.*

Proof. For each clause $c \in C$, let z_1, z_2 and z_3 be the variables in c. Then, let E_c be the following unary ground matching expression:

$$E_c = \{\langle F_{z_1}(F_{z_2}(F_{z_3}(0))), f(0)\rangle, \langle F_{z_1}(F_{z_2}(F_{z_3}(1))), f(1)\rangle\}.$$

For a truth assignment (a_1, a_2, a_3) to (z_1, z_2, z_3) and a matcher σ of E_c, there exists exactly one index i $(1 \leq i \leq 3)$ such that $a_i = 1$ iff $f(w_1)/F_{z_i} \in \sigma$, and $a_l = 0$ iff $w_1/F_{z_l} \in \sigma$ $(l \neq i)$. Then, c is satisfiable by a truth assignment that makes exactly one literal true iff E_c is matchable.

For C, let E be the unary ground matching expression $\bigcup_{j=1}^{m} E_{c_j}$. Then, C is satisfiable by a truth assignment that makes exactly one literal in each clause in C true iff E is matchable. □

Theorem 5. TernaryFfreePredMatching *is NP-complete.*

Proof. For each clause $c \in C$, let z_1, z_2 and z_3 be the variables in c. Then, let E_c be the following ternary predicate function-free matching expression:

$$E_c = \{\langle F(z_1, z_2, z_3), 1\rangle, \langle F(z_2, z_3, z_1), 0\rangle, \langle F(z_3, z_1, z_2), 0\rangle\}.$$

For a truth assignment (a_1, a_2, a_3) to (z_1, z_2, z_3) and a matcher σ of E_c, there exists exactly one index i $(1 \leq i \leq 3)$ such that $a_i = 1$ iff $1/z_i \in \sigma$, and $a_l = 0$ iff $0/z_l \in \sigma$ $(l \neq i)$. Then, c is satisfiable by a truth assignment that makes exactly one literal true iff E_c is matchable.

For C, let E be the ternary function-free predicate matching expression $\bigcup_{j=1}^{m}(E_{c_j}\{F_j(w_1, w_2, w_3)/F\})$. Then, C is satisfiable by a truth assignment that makes exactly one literal in each clause in C true iff E is matchable. □

Theorem 6. BinaryFfreePredMatching *is solvable in polynomial time.*

Proof. We reduce BinaryFfreePredMatching to 2SAT [10]. Let E be a binary function-free predicate matching expression. Without loss of generality, we can suppose that E_F includes pairs $\langle t_1, s_1\rangle, \langle t_2, s_2\rangle$ such that $s_1 \neq s_2$.

Let IC_E, IV_E, and FV_E be the sets of all individual constants, individual variables, and function variables in E, respectively. Suppose that E_F is of the form $\{\langle F(t_1^i, t_2^i), s_i\rangle \mid i \in I\}$. Note that each s_i is in IC_L. For each pair $\langle F(t_1^i, t_2^i), s_i\rangle \in E_F$, construct the following formula T_j^i $(j = 1, 2)$:

1. If $t^i_j \in \mathrm{IC}_E$ and $t^i_j = s_i$, then $T^i_j = \mathtt{true}$;
2. If $t^i_j \in \mathrm{IC}_E$ and $t^i_j \neq s_i$, then $T^i_j = \mathtt{false}$;
3. If $t^i_j = v \in \mathrm{IV}_E$, then $T^i_j = x_{vs_i} \wedge (\bigwedge_{c \in \mathrm{IC}_E - \{s_i\}} \overline{x_{vc}})$.

For E_F, the DNF formula $(\bigwedge_{i \in I} T^i_1) \vee (\bigwedge_{i \in I} T^i_2)$ is denoted by T_{E_F}. The 2CNF formula equivalent to T_{E_F} is denoted by C_{E_F}. For E, $\bigwedge_{F \in \mathrm{FV}_E} C_{E_F}$ is denoted by C_E. The number of clauses in C_E is at most $(\#\mathrm{IC}_E \times \#E)^2 \times \#\mathrm{FV}_E \leq |E|^5$.

Suppose that C_E is satisfiable and let a be a truth assignment to variables $\{x_{vc} \mid v \in \mathrm{IV}_E, c \in \mathrm{IC}_E\}$ satisfying C_E. By the definition of C_E, a satisfies C_{E_F} for any $F \in \mathrm{FV}_E$, so it satisfies T_{E_F}. Then, it also satisfies $\bigwedge_{i \in I} T^i_1$, $\bigwedge_{i \in I} T^i_2$, or both. If a satisfies $\bigwedge_{i \in I} T^i_j$ $(j = 1, 2)$, then we add the bindings w_j/F and c/v to σ for each positive literal $x_{vc} \in \bigwedge_{i \in I} T^i_j$. By the construction of σ and by the definition of $\bigwedge_{i \in I} T^i_j$, σ is a matcher of E_F for any $F \in \mathrm{FV}_E$. Hence, σ is a matcher of E.

Conversely, suppose that E is matchable and let σ be a matcher of E. For $v \in \mathrm{IV}_E$ and $c \in \mathrm{IC}_E$, let a truth assignment a_{vc} to the variable x_{vc} be 1 if $c/v \in \sigma$; 0 otherwise. By the supposition, σ includes the binding either w_1/F or w_2/F for any function variable $F \in \mathrm{FV}_E$. Suppose that $w_j/F \in \sigma$ $(j = 1, 2)$. Since $E_F\{w_j/F\}$ is of the form $\{\langle t^i_j, s_i\rangle \mid i \in I\}$, it holds that $t^i_j\sigma = s_i$. If $t^i_j \in \mathrm{IC}_E$, then it holds that $T^i_j = \mathtt{true}$, since $t^i_j\sigma = t^i_j = s_i$. Then, T^i_j is always satisfiable. If $t^i_j = v_i \in \mathrm{IV}_E$, then it holds that $s_i/v_i \in \sigma$, since $t^i_j\sigma = s_i$. Since T^i_j is of the form $x_{v_i s_i} \wedge (\bigwedge_{c \in \mathrm{IC} - \{s_i\}} \overline{x_{v_i c}})$, the truth assignment $\{a_{v_i c} \mid c \in \mathrm{IC}_E\}$ satisfies T^i_j. By the definition of T_{E_F}, the truth assignment $\{a_{v_i c} \mid c \in \mathrm{IC}_E, i \in I\}$ satisfies T_{E_F}, so it satisfies C_{E_F}. Then, by collecting the truth assignment $\{a_{v_i c} \mid c \in \mathrm{IC}_E, i \in I\}$ for every function variable $F \in \mathrm{FV}_E$, C_E is satisfiable. □

Theorem 7. BinaryFFreeGroundMatching *is NP-complete.*

Proof. For each clause $c \in C$, let z_1, z_2 and z_3 be the variables in c. Then, let E_c be the following binary function-free matching expression:

$$E_c = \left\{ \begin{array}{ll} \langle F_{z_1}(G_{z_1}(H_{z_2}(0), H_{z_3}(0)), H_{z_1}(0)), 1\rangle, & \\ \langle F_{z_1}(H_{z_1}(1), H_{z_2}(1)), 0\rangle, & \langle G_{z_1}(0,0), 0\rangle, \\ \langle F_{z_2}(G_{z_2}(H_{z_3}(0), H_{z_1}(0)), H_{z_2}(0)), 1\rangle, & \\ \langle F_{z_2}(H_{z_2}(1), H_{z_3}(1)), 0\rangle, & \langle G_{z_2}(0,0), 0\rangle, \\ \langle F_{z_3}(G_{z_3}(H_{z_1}(0), H_{z_2}(0)), H_{z_3}(0)), 1\rangle, & \\ \langle F_{z_3}(H_{z_3}(1), H_{z_1}(1)), 0\rangle, & \langle G_{z_3}(0,0), 0\rangle \end{array} \right\}.$$

For a truth assignment (a_1, a_2, a_3) to (z_1, z_2, z_3) and a matcher σ of E_c, there exists exactly one index i $(1 \leq i \leq 3)$ such that $a_i = 1$ iff $1/H_{z_i} \in \sigma$, and $a_l = 0$ iff $0/H_{z_l} \in \sigma$ $(l \neq i)$. Then, c is satisfiable by a truth assignment that makes exactly one literal true iff E_c is matchable.

For C, let E be the binary function-free matching expression $\bigcup_{j=1}^{m} E_{c_j}$. Then, C is satisfiable by a truth assignment that makes exactly one literal in each clause in C true iff E is matchable. □

Theorem 8. UNARYFFREEMATCHING *is solvable in polynomial time.*

Proof. Let E be a unary function-free matching expression. For the transformation rules, we adopt the constraint that a projection on F is applied to E if there exist pairs $\langle t_1, s_1\rangle, \langle t_2, s_2\rangle \in E_F$ such that $s_1 \neq s_2$. Since E is unary, the transformation rules can be applied deterministically to E. □

Theorem 9. 1FVPREDMATCHING *is NP-complete.*

Proof. Let q_1, q_2 and q_3 be terms $g(1,0,0)$, $g(0,1,0)$ and $g(0,0,1)$, respectively. Then, let E be the following 1-fv matching expression:

$$E = \left\{ \begin{array}{l} \langle F(g(x_1^1, x_2^1, x_3^1), y_1, z_1, \cdots g(x_1^m, x_2^m, x_3^m), y_m, z_m), \\ \quad f(q_1, q_2, q_3, \cdots, q_1, q_2, q_3)\rangle, \\ \langle F(d_1, d_1, d_1, \cdots, d_m, d_m, d_m),\ f(d_1, d_1, d_1, \cdots, d_m, d_m, d_m)\rangle \end{array} \right\}.$$

Suppose that C is satisfiable and let $(a_1, \cdots, a_n)$ be a truth assignment to X satisfying C. From $(a_1, \cdots, a_n)$, we obtain the m 3-tuples (a_1^j, a_2^j, a_3^j) $(1 \leq j \leq m)$ assigned to the variables x_1^j, x_2^j, x_3^j in c_j. For each j $(1 \leq j \leq m)$, there exists exactly one index i_j $(1 \leq i_j \leq 3)$ such that $a_{i_j}^j = 1$ and $a_l^j = 0$ $(l \neq i_j)$. Then, E is matchable by the following substitution σ:

$$\begin{array}{l} \sigma = \{f(w_{\mu(1,i_1,1)}, w_{\mu(1,i_1,2)}, w_{\mu(1,i_1,3)}, \cdots, w_{\mu(m,i_m,1)}, w_{\mu(m,i_m,2)}, w_{\mu(m,i_m,3)})/F\} \\ \qquad \cup \{a_1^j/x_1^j, a_2^j/x_2^j, a_3^j/x_3^j, q_{\rho(i_j,2)}/y_j, q_{\rho(i_j,3)}/z_j \mid 1 \leq j \leq m\}, \end{array}$$

where $\rho(l,n) = ((l+n-2) \bmod 3) + 1$ and $\mu(j,l,n) = 3(j-1) + ((3-l+n) \bmod 3) + 1$ for $1 \leq l, n \leq 3$ and $1 \leq j \leq m$.

Conversely, suppose that E is matchable. By Theorem 1, E is matchable iff so is the following matching expression E':

$$E' = \left\{ \left. \begin{array}{l} \langle H_1^j(g(x_1^1, x_2^1, x_3^1), y_1, z_1, \cdots, g(x_1^m, x_2^m, x_3^m), y_m, z_m), q_1\rangle, \\ \langle H_1^j(d_1, d_1, d_1, \cdots d_m, d_m, d_m), d_j\rangle, \\ \langle H_2^j(g(x_1^1, x_2^1, x_3^1), y_1, z_1, \cdots, g(x_1^m, x_2^m, x_3^m), y_m, z_m), q_2\rangle, \\ \langle H_2^j(d_1, d_1, d_1, \cdots d_m, d_m, d_m), d_j\rangle, \\ \langle H_3^j(g(x_1^1, x_2^1, x_3^1), y_1, z_1, \cdots, g(x_1^m, x_2^m, x_3^m), y_m, z_m), q_3\rangle, \\ \langle H_3^j(d_1, d_1, d_1, \cdots d_m, d_m, d_m), d_j\rangle \end{array} \right| 1 \leq j \leq m \right\}.$$

Let σ be a matcher of E'. Then, σ includes the bindings w_{r_1}/H_1^j, w_{r_2}/H_2^j, and w_{r_3}/H_3^j such that $3(j-1)+1 \leq r_1, r_2, r_3 \leq 3j$ for each j $(1 \leq j \leq m)$. Let t_i^j $(1 \leq i \leq 3, 1 \leq j \leq m)$ be the following L-term:

$$H_i^j(g(x_1^1, x_2^1, x_3^1), y_1, z_1, \cdots, g(x_1^m, x_2^m, x_3^m), y_m, z_m),$$

By the definition of t_i^j, for each j, there exists exactly one index i_j $(1 \leq i_j \leq 3)$ such that $t_{i_j}^j \sigma = g(x_1^j, x_2^j, x_3^j)\sigma = q_{i_j}$. Then, we obtain the truth assignment (a_1^j, a_2^j, a_3^j) to (x_1^j, x_2^j, x_3^j) as $q_{i_j} = g(a_1^j, a_2^j, a_3^j)$. By collecting all of (a_1^j, a_2^j, a_3^j) from q_{i_j} $(1 \leq j \leq m)$, we obtain the truth assignment $(a_1, \cdots, a_n)$ to X satisfying C such that $(a_1, \cdots, a_n)$ makes exactly one literal in each clause in C true. □

Theorem 10. kFVFFREEMATCHING *is solvable in polynomial time for* $k \geq 0$.

Proof. Let E be a k-fv function-free matching expression with k function variables $F_1, \cdots, F_k$ and n be the maximum arity of F_i $(1 \leq i \leq k)$. We adopt the same constraint of Theorem 8. Since E is function-free, once applying an imitation or a projection to E decreases at least one function variable in E. Furthermore, a projection is applied to E at most n times for every function variable. Then, we can determine whether E is matchable by checking at most n^k first-order matching expressions. □

Theorem 11. GROUNDPREDMATCHING *is solvable in polynomial time.*

Proof. Let E be a ground predicate matching expression. Consider the following two projections, instead of a projection:

1. **projection 1** (on F): $E \Rightarrow E\{w_i/F\}$,
   if E_F is of the form $\{\langle F(t_1^1, \cdots, t_n^1), t_i^1\rangle, \cdots, \langle F(t_1^m, \cdots, t_n^m), t_i^m\rangle\}$,
2. **projection 2** (on F): $E \Rightarrow$ `fail`,
   if a projection 1 on F cannot be applied to E and there exists pairs $\langle F(t_1, \cdots, t_n), s_1\rangle, \langle F(u_1, \cdots, u_n), s_2\rangle \in E_F$ but $\mathrm{hd}(s_1) \neq \mathrm{hd}(s_2)$.

An imitation on F is applied to E if the above projections 1 and 2 on F cannot be applied. Then, the transformation rules is applied deterministically to E.

Since E is ground and predicate, E is transformed to `fail` by a projection 2 iff $E \not\Rightarrow^* \emptyset$ by only an imitation and a simplification. Furthermore, by an imitation on F and a simplification, the right-hand term of pairs in E_F is decomposed into the subterms. By Theorem 1, the statement holds. □

4 The Comparison between Second-Order Matching and Unification Problems

In this section, we compare the tractability/intractability of the restricted second-order matching problems with the decidability/undecidability of the restricted second-order unification problems.

1. Amiot [1] (and implicitly Farmer [8]) has shown that the unification problem is undecidable for an unary predicate unification expression with at least one binary function constant. On the other hand, by Theorem 3, the matching problem is NP-complete for a unary predicate matching expression with at least one binary function constant.

2. A matching expression is *monadic* if any function constant in it is unary, and *nonmonadic* if it is not monadic. Farmer has shown that the unification problem is decidable for a monadic unification expression [7], but undecidable for a nonmonadic unary one with at least one binary function constant [8]. On the other hand, by Theorem 4, the matching problem is NP-complete for a monadic matching expression. Also by Theorem 3, it is NP-complete for a nonmonadic unary one with at least one binary function constant.

3. Goldfarb [12] has shown that the unification problem is undecidable for a ternary ground unification expression. On the other hand, by Theorem 4, the matching problem is NP-complete for a unary ground matching expression. Note that Amiot's and Farmer's results [1, 8] do not imply that the unification problem is undecidable for a unary ground unification expression, because the existence of individual variables is essential in their proofs.

4. As pointed by Goldfarb [12], the unification problem is decidable for a function-free unification expression. On the other hand, by Theorem 5 or 7, the matching problem is NP-complete for a function-free matching expression. However, if a function-free matching expression is binary predicate, unary, or k-fv ($k \geq 0$), then it is solvable in polynomial time by Theorem 6, 8 or 10.

5. Ganzinger *et al.* [11] have shown that the unification problem is undecidable for an 1-fv unification expression, where the function variable occurs at most twice. On the other hand, by Theorem 9, the matching problem is NP-complete for an 1-fv matching expression, where the function variable occurs at most twice. Also Levy and Veanes [17] have shown that the unification problem is undecidable for an 1-fv ground and an 1-fv unary unification expressions. Whether the corresponding matching problems are NP-complete is still open.

6. A matching expression is *k-linear* if any function variable occurs at most k times. Dowek [6] has shown that the unification problem is decidable for an 1-linear unification expression, and the matching problem is solvable in linear time for an 1-linear matching expression. Furthermore, Levy [16] has shown that the unification problem is undecidable for a 2-linear unification expression. On the other hand, Theorem 3 or 9 claims that the matching problem is NP-complete for a 2-linear matching expression.

5 Conclusion

We summarize the results obtained by this paper in the following table.

expression	matching	ref.	unification	ref.
UnaryPred	NP-complete	Theorem 3	undecidable	[1, 8]
TernaryGround	NP-complete	Theorem 4	undecidable	[12]
UnaryGround	NP-complete	Theorem 4		
TernaryFfreePred	NP-complete	Theorem 5	decidable	[12]
BinaryFfreePred	polynomial	Theorem 6	decidable	[12]
BinaryFfreeGround	NP-complete	Theorem 7	decidable	[12]
UnaryFfree	polynomial	Theorem 8	decidable	[12]
1FVPred	NP-complete	Theorem 9	undecidable	[11]
kFVFfree ($k \geq 0$)	polynomial	Theorem 10	decidable	[12]
GroundPred	polynomial	Theorem 11		
Monadic	NP-complete	Theorem 4	decidable	[7]
Nonmonadic	NP-complete	Theorem 3	undecidable	[8]
1Linear	linear	[6]	decidable	[6]
2Linear	NP-complete	Theorem 3,9	undecidable	[16]

In this paper, we have dealt with the second-order matching problem for a matching expression consisting of L-terms, not L^*-terms. The matching expression consisting of L^*-terms follows the matching problem of *second-order patterns* [18, 19], which is related to the problem GroundPredMatching.

Curien *et al.* [3] have designed a complete second-order matching algorithm which works more efficient than the one of Huet and Lang [14] in most cases. When it is necessary to obtain the complete set of matchers for a given matching expression, we know no more efficient algorithm than their algorithms although it is not a polynomial-time algorithm. It is a future work to give the trade-off between completeness and efficiency of the second-order matching adequate for each research field.

References

1. Amiot, G.: *The undecidability of the second order predicate unification problem*, Archive for Mathematical Logic **30**, 193–199, 1990.
2. Baxter, L. D.: *The complexity of unification*, Doctoral Thesis, Department of Computer Science, University of Waterloo, 1977.
3. Curien, R., Qian, Z. and Shi, H.: *Efficient second-order matching*, Proc. 7th International Conference on Rewriting Techniques and Applications, LNCS **1103**, 317–331, 1996.
4. Défourneaux, G., Bourely, C. and Peltier, N.: *Semantic generalizations for proving and disproving conjectures by analogy*, Journal of Automated Reasoning **20**, 27–45, 1998.
5. 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.
6. Dowek, G.: *A unification algorithm for second-order linear terms*, manuscript, 1993, also available at `http://coq.inria.fr/~dowek/`.
7. Farmer, W. M.: *A unification algorithm for second-order monadic terms*, Annals of Pure and Applied Logic **39**, 131–174, 1988.
8. Farmer, W. M.: *Simple second-order languages for which unification is undecidable*, Theoretical Computer Science **87**, 25–41, 1991.
9. Flener, P.: *Logic program synthesis from incomplete information*, Kluwer Academic Press, 1995.
10. Garey, M. R. and Johnson, D. S.: *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman and Company, 1979.
11. Ganzinger, H., Jacquemard, F. and Veanes, M.: *Rigid reachability*, Proc. 4th Asian Computing Science Conference, LNCS **1538**, 1998, also available at `http://www.mpi-sb.mpg.de/~hg/`.
12. Goldfarb, W. D.: *The undecidability of the second-order unification problem*, Theoretical Computer Science **13**, 225–230, 1981.
13. Harao, M.: *Proof discovery in LK system by analogy*, Proc. 3rd Asian Computing Science Conference, LNCS **1345**, 197–211, 1997.
14. Huet, G. P. and Lang, B.: *Proving and applying program transformations expressed with second-order patterns*, Acta Informatica **11**, 31–55, 1978.
15. Levy, J.: *Linear second-order unification*, Proc. 7th International Conference on Rewriting Techniques and Applications, LNCS **1103**, 332–346, 1996.

16. Levy, J.: *Decidable and undecidable second-order unification problem*, Proc. 9th International Conference on Rewriting Techniques and Applications, LNCS **1379**, 47–60, 1998.
17. Levy, L. and Veanes, M.: *On unification problems in restricted second-order languages*, Proc. Annual Conference of the European Association for Computer Science Logic (CSL'98), 1998, also available at `http://www.iiia.csic.es/~levy/`.
18. Miller, D.: *A logic programming language with lambda-abstraction, function variables, and simple unification*, Journal of Logic and Computation **1**, 497–536, 1991.
19. Prehofer, C.: *Decidable higher-order unification problems*, Proc. 12th International Conference on Automated Deduction (CADE 12), LNAI **814**, 635–649, 1994.

This article was processed using the LaTeX macro package with LLNCS style