Syntactic Unification Problem under Constrained Substitutions

Yuichi Kaji Kazuhiro Takada Tadao Kasami

Nara Institute of Science and Technology,
Graduate School of Information Science
Takayama 8916-5, Ikoma, Nara 630-01, Japan
{kaji,kazuh-t,kasami}@is.aist-nara.ac.jp

Abstract. A syntactic unification problem under constrained substitutions (SUPCS) has been proposed as an extension of the usual unification problem. Given a pair of goal terms and a finite set A of terms, SUPCS is to decide whether goal terms are unifiable by replacing variables in goal terms with terms which are recursively constructed of terms in A. In this paper, the followings are shown: (1) SUPCS is undecidable in general; (2) if terms in A are linear or ground, then SUPCS is decidable; (3) if goal terms are nonlinear or they share same variables, then SUPCS is $\mathcal{NP}$-hard in general even if all terms in A are ground; (4) if terms in A are linear or ground, goal terms are linear and they share no variables, then SUPCS belongs to P.

1 Introduction

A *syntactic unification problem* is the one to decide whether given two (first-order) *terms* g_1 and g_2 have a substitution σ such that $g_1\sigma = g_2\sigma$. The problem often plays an essential role in computational logic, logic programming, and some AI-applications that are based on symbolic computation[1]. Furthermore, the problem is significant since it is a basis of *E-unification problems*. An E-unification problem takes a triple $\langle g_1, g_2, E\rangle$ as an input where E is a set of axioms, and decide whether there is a substitution σ such that $g_1\sigma =_E g_2\sigma$. A syntactic unification can be regarded as a special simple case of E-unification such that $E = \emptyset$ (empty set). Indeed, E-unification procedures such as narrowing[7] and decomposition methods[2] solve an E-unification problem by reducing an instance of E-unification problem into a set of syntactic unification problems. Due to a number of eager studies on syntactic unification problems, a linear time procedure to solve the problem has been proposed[9].

Usually, the substitution σ considered in unification problems is a "free" substitution in the sense that an arbitrary term can substitute for a variable. However, the authors are interested in unification problems with "non-free" substitutions because of the following reason: When one uses terms and axioms to model a practical problem, operations and elementary objects in the practical world are often associated with function symbols. Complex objects are represented by first-order terms, and axioms are used to describe relations among

operations and objects. The situation that an arbitrary term can substitute for a variable corresponds to the situation that we can use arbitrary objects in the practical world to accomplish a goal. But this is not a very common case. It seems more natural and practically important to investigate the case where there are some constraints on operations and objects that we can use. This observation inspired the concept of *unification problems under constrained substitutions*[3, 4, 8].

Unification problems under constrained substitutions are defined to be extensions of the usual unification problems. Thus, it will be convenient if we can extend fruitful results of the usual unification problems to the constrained substitution case. For this purpose, it is significant to clarify basic properties of terms, especially syntactic unification problems, under constrained substitutions.

This paper is devoted to the decidability and computational complexity of the *syntactic unification problem under constrained substitutions* (*SUPCS*). An instance of SUPCS consists of a finite set A of terms and a pair of goal terms g_1 and g_2. The problem is to decide whether there is a substitution σ such that $g_1\sigma = g_2\sigma$ and every term in the co-domain of σ is a term which is recursively constructed of terms in A. After reviewing some basic notations in Sect. 2, we formally define SUPCS in Sect. 3. In Sect. 4, it is shown that the problem is undecidable in general. In Sect. 5, SUPCS is shown to be decidable if terms in A are linear or ground. However, if goal terms are nonlinear or there are variables that occur in both of goal terms, then SUPCS is $\mathcal{NP}$-hard in general even if all terms in A are ground. If goal terms are linear and there is no variable that occurs in both of goal terms, then the problem is shown to belong to $\mathcal{P}$.

2 Preliminary

In this section, notations used in this paper are briefly introduced. Readers are assumed to have basic knowledge of the first-order terms and related topics. Throughout this paper, we assume that F and X are sets of *function symbols* and *variables*, respectively. The *arity* of a function symbol f is denoted by $a(f)$. A function symbol c such that $a(c) = 0$ is called a *constant*. The set of *terms* over F and X is denoted by $\mathcal{T}_F[X]$. For a term $t \in \mathcal{T}_F[X]$, the set of variables occurring in t is denoted by $\mathcal{V}(t)$. If $\mathcal{V}(t) = \phi$, then t is called a *ground* term. The set of ground terms is denoted by $\mathcal{T}_F$. A term is *linear* if no variable occurs more than once in the term. An *occurrence* of a term is defined in the usual way, as a concatenation of integers. The root occurrence is denoted by λ, and the set of occurrences of a term t is denoted by $\mathcal{O}(t)$. The *size* of a term t is defined to be the cardinality of $\mathcal{O}(t)$. A *subterm* of t at an occurrence o is denoted by t/o. If a term t is obtained from a term t' by replacing subterms of t' at occurrences $o_1, \ldots, o_m$ ($o_i \in \mathcal{O}(t')$ for $1 \leq i \leq m$, o_i and o_j are disjoint if $i \neq j$) with terms $s_1, \ldots, s_m$, respectively, then we write $t = t'[o_i \leftarrow s_i \mid 1 \leq i \leq m]$.

A *substitution* σ is a mapping from variables in X to terms in $\mathcal{T}_F[X]$, and written as $\sigma = \{x_1 \mapsto t_1, \ldots, x_m \mapsto t_m\}$ where x_i ($1 \leq i \leq m$) is a variable in X and t_i ($1 \leq i \leq m$) is a term in $\mathcal{T}_F[X]$. If a term t is obtained from a term s by replacing variables $x_1, \ldots, x_m$ in s with terms $t_1, \ldots, t_m$, respectively, then

we write $t = s\sigma$ and say t is an *instance* of s. For two substitutions σ_1 and σ_2, if there is a substitution σ_3 such that $\sigma_2 = \sigma_1\sigma_3$, that is, $x\sigma_2 = x\sigma_1\sigma_3$ for every $x \in X$, then we say σ_1 is *more general* than σ_2.

Two terms s and t are *syntactically unifiable* (or simply *unifiable*) if there is a substitution σ such that $s\sigma = t\sigma$. Such a substitution σ is called a *unifier* of s and t. Remark that even if s and t are unifiable, a unifier of s and t is not always unique. Let $U_{s,t}$ be the set of unifiers of s and t. It is known that if $U_{s,t}$ is not empty, then $U_{s,t}$ contains a substitution σ such that σ is more general than any other unifiers in $U_{s,t}$, and $\sigma\sigma' \in U_{s,t}$ for an arbitrary substitution σ'[1]. Such a unifier σ is unique if one ignores variable renaming, and is called the *most general unifier* (*mgu*) of s and t[1]. It is known that mgu is computable in linear time to the size of s and t. An instance of a *syntactic unification problem* is a pair of *goal* terms $\langle g_1, g_2 \rangle$, and the problem is to decide whether g_1 and g_2 are unifiable or not.

3 Unification Problem under Constrained Substitutions

Let $A \subset \mathcal{T}_F[X]$ be a finite set of terms, and let $\mathcal{T}_A[X]$ be the minimal set of terms satisfying the followings.

1. If $x \in X$, then $x \in \mathcal{T}_A[X]$.
2. If $t \in A$, $\mathcal{V}(t) = \{x_1, \ldots, x_n\}$ and $t_1, \ldots, t_n \in \mathcal{T}_A[X]$, then $t\sigma \in \mathcal{T}_A[X]$ where $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

A term in $\mathcal{T}_A[X]$ is called an *A-term* and a substitution $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is called an *A-substitution* if all of $t_1, \ldots, t_n$ are A-terms. If a term t is written as $t = s\sigma$ for a term s and an A-substitution σ, then t is called an *A-instance* of s. Two terms s and t are *A-unifiable* if there is an A-substitution σ such that $s\sigma = t\sigma$. An instance of a *syntactic unification problem under constrained substitutions* (*SUPCS* for short) is a triple $\langle g_1, g_2, A \rangle$ where $g_1, g_2 \in \mathcal{T}_F[X]$ and A is a finite subset of $\mathcal{T}_F[X]$, and the problem is to decide whether g_1 and g_2 are A-unifiable or not.

Example 1. Let f and h be function symbols such that $a(f) = 2$ and $a(h) = 1$, and suppose that $g_1 = f(h(x_1), x_2)$, $g_2 = f(x_3, x_3)$ and $A = \{h(h(x))\}$. The terms g_1 and g_2 are unifiable in the usual sense since there is a substitution $\sigma = \{x_1 \mapsto y, x_2 \mapsto h(y), x_3 \mapsto h(y)\}$ by which $g_1\sigma = g_2\sigma$. However, there is no A-substitution that unifies g_1 and g_2 (remark that $\mathcal{T}_A[X] = X \cup \{\underbrace{h(h(\cdots h}_{2n}(x)\cdots)) \mid x \in X, n \text{ is a nonnegative integer}\}$). □

Consider the following three kinds of conditions on instances of SUPCS.

– a condition on terms in A;
  (1) terms in A are ground.
  (2) terms in A are ground or linear.
  (3) there is no restriction on terms in A.

– a condition on linearity of g_1 and g_2;
  (A) both of g_1 and g_2 are linear.
  (B) there is no restriction on linearity of goal terms.
– a condition on variables occurring in g_1 and g_2;
  (a) $\mathcal{V}(g_1) \cap \mathcal{V}(g_2) = \emptyset$.
  (b) there is no restriction on variables occurring in goal terms.

According to these conditions, we can define 12 classes of instances of SUPCS. Fig. 1 illustrates relations of these classes of instances of SUPCS with respect to decidability of the problem. In the figure, "1Aa" stands for the class of instances of SUPCS that satisfy the conditions (1), (A) and (a) above. An arc $\mathcal{C}_1 \longrightarrow \mathcal{C}_2$ in the figure means that SUPCS for the class $\mathcal{C}_1$ is not easier than SUPCS for the class $\mathcal{C}_2$. That is, if SUPCS for the class $\mathcal{C}_1$ is decidable, then so is for the class $\mathcal{C}_2$. Conversely, if SUPCS for the class $\mathcal{C}_2$ is undecidable, then so is for the class $\mathcal{C}_1$. Though most relations in the figure are trivial, three arcs 3Ba $\longrightarrow$ 3Bb, 2Ba $\longrightarrow$ 2Bb and 1Ba $\longrightarrow$ 1Bb are not. These relations hold due to the following lemma.

Lemma 1. *SUPCS for the class 3Bb (resp. 2Bb, 1Bb) is polynomial-time reducible to SUPCS for the class 3Ba (resp. 2Ba, 1Ba).*

Proof. Let $\langle g_1, g_2, A\rangle$ be an instance of SUPCS such that $\mathcal{V}(g_1) \cap \mathcal{V}(g_2) \neq \emptyset$. We can construct another instance $\langle g_1', g_2', A\rangle$ of SUPCS such that g_1 and g_2 are A-unifiable if and only if g_1' and g_2' are A-unifiable and $\mathcal{V}(g_1') \cap \mathcal{V}(g_2') = \emptyset$. First, let g_1'' and g_2'' be terms that are obtained from g_1 and g_2 by renaming variables so that $(\mathcal{V}(g_1) \cup \mathcal{V}(g_2)) \cap (\mathcal{V}(g_1'') \cup \mathcal{V}(g_2'')) = \emptyset$. Introduce a new function symbol *dummy* with arity three, and define $g_1' = dummy(g_1, g_1, g_2)$ and $g_2' = dummy(g_1'', g_2'', g_2'')$. It is easily understood that g_1' and g_2' satisfy the above property, and that g_1' and g_2' can be constructed in linear time to the size of g_1 and g_2. □

Though the results of [8] imply that SUPCS for the class 1Aa is decidable, there are no results concerning the decidability of SUPCS for other classes. In this paper, we show the followings (see also Fig. 1):

– SUPCS for the class 3Aa is undecidable. As a corollary, we can conclude that SUPCS for the classes 3Ab, 3Ba and 3Bb is also undecidable.

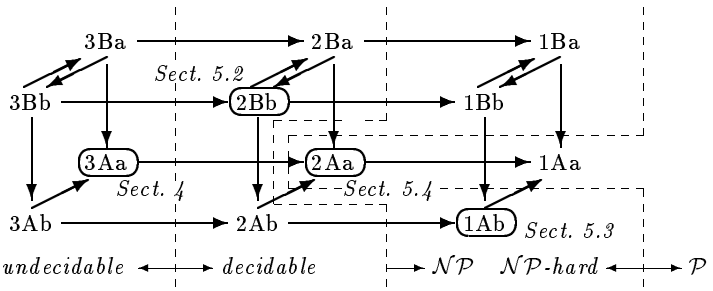

Fig. 1. Relation of 12 classes of instances of SUPCS with respect to decidability.

- SUPCS for the class 2Bb is decidable. We can conclude that SUPCS for the classes 2Ba, 2Ab, 2Aa, 1Bb, 1Ba, 1Ab and 1Aa is also decidable.
- SUPCS for the class 1Ab is $\mathcal{NP}$-hard. Thus, SUPCS for 2Bb, 2Ba, 2Ab, 1Bb and 1Ba is also $\mathcal{NP}$-hard. Furthermore, it is easily shown that SUPCS for 1Bb, 1Ba and 1Ab is $\mathcal{NP}$-complete.
- SUPCS for the class 2Aa is solvable in deterministic polynomial time, and so is for the class 1Aa.

4 Undecidability Result for the Class 3Aa

The accepting problem of Turing machines is reducible to SUPCS for the class 3Aa. See [6] for the definitions of Turing machines and the accepting problem of Turing machines. Given a Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_s, B, Q_F)$ where $\Gamma = \Sigma \cup \{B\}$ and a sequence $w \in \Sigma^*$ of input symbols, we construct a set A of terms and goal terms g_1 and g_2 that satisfy the following property.

Property 2. *Turing machine M accepts w if and only if g_1 and g_2 are A-unifiable.*

First, the set F of function symbols is defined to be $F = Q \cup \Gamma \cup \{\not\epsilon, \vdash\}$ where $\not\epsilon$ and $\vdash$ are newly introduced symbols such that $a(\not\epsilon) = 0$ and $a(\vdash) = 2$. Arities of $q \in Q \subset F$ and $c \in \Gamma \subset F$ are defined to be two and one, respectively. The set A of terms is defined as follows. In this definition, let x, y, u and v be variables.

Type 1: $c(x) \in A$ for every $c \in \Gamma \subset F$.

Type 2: $\not\epsilon \in A$.

For a string $\alpha = c_1 c_2 \cdots c_n$ $(n \geq 0, c_i \in \Gamma, 1 \leq i \leq n)$, we use $\overleftarrow{\alpha}$ to denote a term $c_n(\cdots(c_2(c_1(\not\epsilon)))\cdots)$, and $\overrightarrow{\alpha}$ to denote a term $c_1(c_2(\cdots c_n(\not\epsilon)\cdots))$. It is obvious that, for an arbitrary string $\alpha \in \Gamma^*$, both of $\overleftarrow{\alpha}$ and $\overrightarrow{\alpha}$ are A-terms. We would like to associate an ID (instantaneous description) $\alpha q \beta$ $(\alpha, \beta \in \Gamma^*, q \in Q)$ of M with a term $q(\overleftarrow{\alpha}, \overrightarrow{\beta})$, and associate a computation sequence of M $\alpha_1 q_1 \beta_1 \vdash \alpha_2 q_2 \beta_2 \vdash \cdots$ with a term $\vdash(q_0(\overleftarrow{\alpha_0}, \overrightarrow{\beta_0}), \vdash(q_1(\overleftarrow{\alpha_1}, \overrightarrow{\beta_1}), \cdots))$. Terms of the following Types 3 to 6 are introduced to realize one step move of M. Let $q_i, q_j \in Q$, $b \in \Sigma$, and $c \in \Gamma$.

Type 3: If $\delta(q_i, b) = (q_j, c, R)$, then $\vdash(q_i(x, b(y)), \vdash(q_j(c(x), y), u)) \in A$.

Type 4: If $\delta(q_i, b) = (q_j, c, L)$, then $\vdash(q_i(b'(x), b(y)), \vdash(q_j(x, b'(c(y))), u)) \in A$ for every $b' \in \Gamma$.

Type 5: If $\delta(q_i, B) = (q_j, c, R)$, then $\vdash(q_i(x, \not\epsilon), \vdash(q_j(c(x), \not\epsilon), u)) \in A$.

Type 6: If $\delta(q_i, B) = (q_j, c, L)$, then $\vdash(q_i(b'(x), \not\epsilon), \vdash(q_j(x, b'(c(\not\epsilon))), u)) \in A$ for every $b' \in \Gamma$.

We need special treatments for final states.

Type 7: $q_F(u,v) \in A$ for every final state $q_F \in Q_F$.

Type 8: $\vdash(q_F(u,v), q_F(u,v)) \in A$ for every final state $q_F \in Q_F$.

Goal terms g_1 and g_2 are defined as $g_1 = \vdash(q_s(\overleftarrow{\varepsilon}, \overrightarrow{w}), x)$ and $g_2 = y$ where ε is an empty string. Remark that the triple $\langle g_1, g_2, A\rangle$ belongs to the class 3Aa of SUPCS. In the following, we show that the set A and goal terms g_1 and g_2 constructed as above satisfy Prop. 2.

Lemma 3. *If $\alpha q\beta \vdash \alpha' q' \beta'$ $(\alpha, \beta, \alpha', \beta' \in \Gamma^*, q, q' \in Q)$, then there is an A-term $\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), \vdash(q'(\overleftarrow{\alpha'}, \overrightarrow{\beta'}), u))$.*

Proof. Obvious from the definition of A. □

Proof of Prop. 2 (Only if part). It suffices to show that if there is an accepting computation $\alpha q\beta \vdash^* \alpha_F q_F \beta_F$ $(\alpha, \beta, \alpha_F, \beta_F \in \Gamma^*, q \in Q, q \in Q_F)$ of M, then there are A-terms t_x and t_y such that $\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t_x) = t_y$. The proof is by induction on the length of the computation $\alpha q\beta \vdash^* \alpha_F q_F \beta_F$. If the length is zero, then $q = q_F \in Q_F$, and hence there are terms $q(u,v)$ and $\vdash(q(u,v), q(u,v))$ in A (see Types 7 and 8 of the definition of A). By substituting u and v in these terms by A-terms $\overleftarrow{\alpha}$ and $\overrightarrow{\beta}$, respectively, we have A-terms $q(\overleftarrow{\alpha}, \overrightarrow{\beta})$ and $\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), q(\overleftarrow{\alpha}, \overrightarrow{\beta}))$. By putting $t_x = q(\overleftarrow{\alpha}, \overrightarrow{\beta})$ and $t_y = \vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), q(\overleftarrow{\alpha}, \overrightarrow{\beta}))$, the only if part holds in this case.

Next, suppose that the only if part holds for any computation with length k or less, and consider that the length of $\alpha q\beta \vdash^* \alpha_F q_F \beta_F$ is $k+1$. Such a computation is written as $\alpha q\beta \vdash \alpha_1 q_1 \beta_1 \vdash^* \alpha_F q_F \beta_F$ where $\alpha_1, \beta_1 \in \Gamma^*$ and $q_1 \in Q$. By $\alpha q\beta \vdash \alpha_1 q_1 \beta_1$ and Lemma 3, there is an A-term

$$\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), \vdash(q_1(\overleftarrow{\alpha_1}, \overrightarrow{\beta_1}), u)). \tag{1}$$

Since the length of the computation $\alpha_1 q_1 \beta_1 \vdash^* \alpha_F q_F \beta_F$ is k, there are A-terms t'_x and t'_y such that $\vdash(q_1(\overleftarrow{\alpha_1}, \overrightarrow{\beta_1}), t'_x) = t'_y$ by the inductive hypothesis. Let $t_x = t'_y$ and $t_y = \vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), \vdash(q_1(\overleftarrow{\alpha_1}, \overrightarrow{\beta_1}), t'_x))$ (the term that is obtained from (1) by substituting t'_x for u), then both of t_x and t_y are A-terms, and

$$\begin{aligned}\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t_x) &= \vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t'_y) \\ &= \vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), \vdash(q_1(\overleftarrow{\alpha_1}, \overrightarrow{\beta_1}), t'_x)) = t_y.\end{aligned}$$

This completes the induction, and the only if part is shown.

(If part). We show that if there are A-terms t_x and t_y such that $\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t_x) = t_y$, then there are $\alpha_F, \beta_F \in \Gamma^*$ and $q_F \in Q_F$ such that $\alpha q\beta \vdash^* \alpha_F q_F \beta_F$. Since t_y is an A-term, there are a term $t_0 \in A$ and an A-substitution σ_y such that $t_y = t_0\sigma_y$. Due to the construction of A, t_0 must be any of Type 1 to 8. If t_0 is either one of Type 1, 2 or 7, then the function symbol of the term

$t_y = t_0\sigma_y$ at the root occurrence is not "$\vdash$", which contradicts the fact that the root symbol of $\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t_x)$ is "$\vdash$". Therefore t_0 must be any of Type 3, 4, 5, 6 or 8. The if part is shown by induction on the number of occurrences of the symbol $\vdash$ in t_y $(= \vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t_x))$. First, consider the case that the number of occurrences of $\vdash$ is one. If t_0 is of either one of Type 3, 4, 5 or 6, then t_y has at least two occurrences of $\vdash$. Thus t_0 must be of type 8, and written as $t_0 = \vdash(q'(u,v), q'(u,v))$ with $q' \in Q_F$. Since $t_y = \vdash(q'(u,v), q'(u,v))\sigma_y$ equals to $\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t_x)$, it follows that $q = q' \in Q_F$. The if part holds by taking $\alpha_F = \alpha$, $\beta_F = \beta$ and $q_F = q$ in this case.

Next, suppose that the if part holds for any case where the number of occurrences of $\vdash$ in t_y is k (> 1) or less, and consider that t_y has $k+1$ occurrences of $\vdash$. If t_0 is a term of Type 8, then the number of occurrences of $\vdash$ in $t_y = t_0\sigma_y = \vdash(q'(u,v), q'(u,v))\sigma_y$ is $1 + 2 \times$ (the number of $\vdash$ in $q'(u,v)\sigma_y$). Remark that $t_y = \vdash(q'(u,v), q'(u,v))\sigma_y = \vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t_x)$ implies $q'(u,v)\sigma_y = q(\overleftarrow{\alpha}, \overrightarrow{\beta})$, and hence $\vdash$ does not occur in $q'(u,v)\sigma_y$. It follows that t_y has only one occurrence of $\vdash$, which contradicts the assumption that the number of occurrences of $\vdash$ in t_y is $k+1 \geq 2$. Thus, t_y must be any of Type 3, 4, 5 or 6. Without loss of generality, let $\alpha = a_1 \cdots a_m$ and $\beta = b_1 \cdots b_n$.

If t_0 is of Type 3, then t_0 is written as $t_0 = \vdash(q_i(u, b(v)), \vdash(q_j(c(u), v), z))$, and σ_y is written as $\sigma_y = \{u \mapsto t_u, v \mapsto t_v, z \mapsto t_z\}$ for A-terms t_u, t_v and t_z. Since $t_y = t_0\sigma_y = \vdash(q_i(t_u, b(t_v)), \vdash(q_j(c(t_u), t_v), t_z))$ equals to the term $\vdash(q(\overleftarrow{\alpha}, \overrightarrow{\beta}), t_x)$, it follows that $q_i = q$, $t_u = \overleftarrow{\alpha}$, $b(t_v) = \overrightarrow{\beta}$ (and hence $b = b_1$), and

$$\vdash(q_j(c(t_u), t_v), t_z) = t_x. \tag{2}$$

Let $\alpha' = a_1 \cdots a_m c$ and $\beta' = b_2 \cdots b_n$, then $\overleftarrow{\alpha'} = c(\overleftarrow{\alpha})$, $\overrightarrow{\beta'} = t_v$ and (2) is written as $\vdash(q_j(\overleftarrow{\alpha'}, \overrightarrow{\beta'}), t_z) = t_x$. Since the number of occurrences of $\vdash$ in t_x is k, by the inductive hypothesis, there are $\alpha_F, \beta_F \in \Gamma^*$ and $q_F \in Q$ such that $\alpha' q_j \beta' \vdash^* \alpha_F q_F \beta_F$. On the other hand, $t_0 \in A$ means that δ has a function value $\delta(q_i, b) = \delta(q, b_1) = (q_j, c, R)$, and hence $\alpha q \beta \vdash \alpha' q_j \beta'$. Summarizing the above discussion, $\alpha q \beta \vdash \alpha' q_j \beta' \vdash^* \alpha_F q_F \beta_F$ and the if part holds in this case. Similar discussion holds for other cases where t_0 is of Type 4, 5 or 6, and the lemma holds. □

Since the accepting problem of Turing machines is undecidable, we have the following theorem.

Theorem 4. *SUPCS for the classes 3Aa, 3Ab, 3Ba and 3Bb is undecidable.*

5 Classes for which SUPCS is decidable

We introduce a procedure to solve SUPCS for the class 2Bb. Since *tree automata* play an essential role in the procedure, we first review a definition and basic properties of tree automata in Sect. 5.1. The procedure is given in Sect. 5.2. Sects. 5.3 and 5.4 are devoted to discuss computational complexity of SUPCS.

5.1 Tree Automata

A *tree automaton* is an automaton of which input is a term. Formally, a tree automaton is defined by a 5-tuple $M = (Q, F, \delta, q_0, Q_F)$ where Q is a finite set of states, F is a finite set of function symbols, δ is a transition function, $q_0 \in Q$ is an initial state and $Q_F \subset Q$ is a set of final states. The transition function δ maps $Q \times \{F \cup \{\varepsilon\}\}$ to $\bigcup_{n=1}^{amax} 2^{Q^n}$ where 2^{Q^n} is the power set of n-tuples of Q and $amax$ is the maximum arity of function symbols in F. For $q \in Q$ and $f \in F$, δ is defined as in the following forms:

$$\begin{array}{ll} (q_1, \ldots, q_n) \in \delta(q, f) & (n = \max(1, a(f)), q_1, \ldots, q_n \in Q) \\ (q_1) \in \delta(q, \varepsilon) & (q_1 \in Q). \end{array}$$

This is essentially the same formalism as a *nondeterministic root-to-frontier tree recognizer* in [5], though notations and terminologies are modified to make discussion succinctly.

An input of a tree automaton M is a ground term in $\mathcal{T}_F$. At the initial moment, a tree automaton has only one head which is scanning the root of an input term. Furthermore, the initial state q_0 is associated with the head. If a head of M scans a symbol f with a state q, and if $(q_1, \ldots, q_n) \in \delta(q, f)$ where $n = a(f)$, then M "splits" the head into n pieces, and attaches each head to the root of the i-th child of f with the new state q_i. We also allow a tree automaton M to have ε-moves such that if a head of M is associated with a state q, and if $(q_1) \in \delta(q, \varepsilon)$, then M changes the state that is associated with the head from q to q_1, but does not move the head. A ground term t is accepted by M if every leaf of t is visited by one of heads of M, and the head falls off the input term by changing its state to a final state, like as a head of string automaton falls off an input tape.

Formally, an *ID* of a tree automaton $M = (Q, F, \delta, q_0, Q_F)$ for a term is defined as follows. Let $QF = \{qf \mid q \in Q, f \in F\}$ and $F_0Q = \{cq \mid c \in F, a(c) = 0, q \in Q\}$. Intuitively, the symbol $qf \in QF$ represents that a head scans a symbol $f \in F$ with the state $q \in Q$. The symbol $cq \in F_0Q$ is a symbol to represent that a head has fallen off the input term by reading the constant c, and the fallen head is associated with the state q. An arity of a symbol $qf \in QF$ is defined to be $a(f)$, and an arity of a symbol in F_0Q is defined to be 0. A term $I \in \mathcal{T}_{F \cup QF \cup F_0Q}$ is said to be an *ID* of M for a term $t \in \mathcal{T}_F$ if the following conditions hold:

1. If one replaces symbols $qf \in QF$ and $cq \in F_0Q$ in I by f and c, respectively, then the resulting term is t.
2. For every leaf of I, the path from the leaf to the root of I contains exactly one symbol in $QF \cup F_0Q$.

For the readability of the following discussion, an ID $qf(t_1, \ldots, t_n)$ (the root is a symbol $qf \in QF$) is written as $q\lceil f(t_1, \ldots, t_n)\rceil$. Intuitively, this ID represents a situation such that a head which is associated with a state q scans the root of a term $f(t_1, \ldots, t_n)$.

A *move* of M is represented by a binary relation "$\vdash$" on IDs of M, which is defined as follows:

1. If $(q_1) \in \delta(q, \varepsilon)$, then $q\lceil t \rceil \vdash q_1\lceil t \rceil$ for any term t.
2. If $(q_1, \ldots, q_n) \in \delta(q, f)$ where $n = a(f) > 0$, then $q\lceil f(t_1, \ldots, t_n) \rceil \vdash f(q_1\lceil t_1 \rceil, \ldots, q_n\lceil t_n \rceil)$.
3. If $(q_1) \in \delta(q, c)$ where $a(c) = 0$, then $q\lceil c \rceil \vdash cq_1$.
4. Let I be an ID and let o be an occurrence of I. If $I/o = I_1$ is an ID such that $I_1 \vdash I_2$, then $I \vdash I[o \leftarrow I_2]$.

Let $\vdash^*$ be the reflexive and transitive closure of $\vdash$. An ID I of M for t is called an *accepting ID* if every leaf of I is a symbol in the form $cq \in F_0Q$ with $q \in Q_F$. Intuitively, an accepting ID corresponds to a situation such that each head having scanned constant leaf has fallen off the input term with a final state. The tree automaton M *accepts* a ground term t if and only if there is an accepting ID I of M for t such that $q_0\lceil t \rceil \vdash^* I$.

As for moves of tree automata, the following lemma holds. Remark that these properties are counterparts of well-known properties of string automata, and hence proofs are omitted.

Lemma 5. *Let I_1, I_2 and I_3 be IDs of M for a term t such that $I_1 \vdash^* I_2 \vdash^* I_3$. Furthermore let $o_1, \ldots, o_n$ be occurrences of t such that $I_2/o_i = q_i\lceil t/o_i \rceil$ $(q_i \in Q)$ for $1 \leq i \leq n$ (i.e. heads of M visit subterms of t at occurrences $o_1, \ldots, o_n$).*

1. *For $1 \leq i \leq n$, $q_i\lceil t/o_i \rceil \vdash^* I_3/o_i$ holds. If I_3 is an accepting ID of M for t, then I_3/o_i is an accepting ID of M for t/o_3.*
2. *For arbitrary terms $s_1, \ldots, s_n$, $I_1[o_1 \leftarrow s_1, \ldots, o_n \leftarrow s_n] \vdash^* I_2[o_1 \leftarrow q_1\lceil s_1 \rceil, \ldots, o_n \leftarrow q_n\lceil s_n \rceil]$.*

Let $L(M)$ be a set of ground terms that is accepted by M. It is known that, for arbitrary tree automata M_1 and M_2, we can construct a tree automaton M_3 such that $L(M_3) = L(M_1) \cap L(M_2)$ in $O(|M_1| \times |M_2|)$-time[5] where $|M|$ stands for the size of M. Furthermore, given a tree automaton M, it is decidable in $O(|M|^2)$-time whether $L(M)$ is empty or not[5].

5.2 Decidability Result for the Class 2Bb

In this section, we introduce a procedure to solve SUPCS for the class 2Bb. Let $\langle g_1, g_2, A \rangle$ be a given instance of SUPCS which belongs to the class 2Bb, and let $\mathcal{V}(g_1) \cup \mathcal{V}(g_2) = \{x_1, \ldots, x_n\}$. An overview of the procedure is as follows: The procedure first computes the mgu $\sigma_{\mathbf{mgu}} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ of g_1 and g_2. Then the procedure checks whether there is a substitution σ such that all of $t_i\sigma$ with $1 \leq i \leq n$ are A-terms. Remark that g_1 and g_2 are A-unifiable if and only if there is such a substitution σ ($\sigma_{\mathbf{mgu}}\sigma$ is the A-substitution that unifies g_1 and g_2). The essential point of this procedure is how to check the existence of such a substitution σ. We use tree automata constructed as follows to help this check.

Let $A_0 = \{t \mid t \text{ is a subterm of a term in } A\}$. For every term $t \in A_0$, we construct a tree automaton $M[t]$ such that $L(M[t])$ is the set of *Ω-terms* of A-instances of t. An Ω-term of a term s, denoted by $\underline{s}$, is a term that is obtained from s by replacing all variables in s with a new symbol Ω. For example, $\underline{f(x_1, x_2, h(x_3))} = f(\Omega, \Omega, h(\Omega))$.

Construction of tree automata: Components of tree automata are almost the same for all $t \in A_0$. The set F' of function symbols of the tree automata is defined to be $F' = F \cup \{\Omega\}$ (remark that, by convention, F is the set of function symbols of SUPCS). The set Q of states are defined to be $Q = \{[\underline{s}] \mid s \in A_0\} \cup \{[\Omega], q_f\}$ where q_f is a newly introduced symbol that does not belong to F. The function δ is defined as follows:

- For $[f(t_1, \ldots, t_n)] \in Q$ such that $f \in F$ and $a(f) = n > 0$, define $\delta([f(t_1, \ldots, t_n)], f) = \{([t_1], \ldots, [t_n])\}$.
- For $[c] \in Q$ such that $c \in F$ and $a(c) = 0$, define $\delta([c], c) = \{(q_F)\}$.
- Define $\delta([\Omega], \Omega) = \{(q_f)\}$.
- Define $\delta([\Omega], \varepsilon) = \{([\underline{s}]) \mid s \in A\}$.

For $t \in A_0 \cup \{\Omega\}$, let $M[t]$ be a tree automaton $M[t] = (F', Q, \delta, [\underline{t}], \{q_f\})$. This construction of tree automata is essentially the same as that of [8], and the following lemma holds.

Lemma 6. *For $t \in A_0$, $L(M[t]) = \{\underline{t\sigma} \mid \sigma$ is an A-constrained substitution$\}$, and especially $L(M[\Omega])$ equals to the set of A-terms.*

Let us brief the following Procedure 1. For simplicity, we extend the concept of IDs of tree automata to non-ground terms in a natural way. That is, an ID is a term in$\mathcal{T}_{F \cup QF \cup F_0 Q}[X \cup QX]$ where $QX = \{qx \mid q \in Q, \in X\}$. An ID I of M for t is called a *partial accepting ID of M for t* if every leaf of I is a symbol in the form $cq \in F_0 Q$ with $q \in Q_F$, or a symbol in QX. A partial accepting ID corresponds to a situation such that each head having scanned a constant leaf has fallen off the input term with a final state and only those heads scanning variables remain on the term.

Assume that for the mgu $\sigma_{\mathbf{mgu}} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ of g_1 and g_2, there is a substitution σ such that all of $t_i\sigma$ $(1 \leq i \leq n)$ are A-terms. By Lemma 6, $M[\Omega]$ accepts $\underline{t_i\sigma}$ for $1 \leq i \leq n$. In this case, if t_i is given as an input of $M[\Omega]$, then $M[\Omega]$ can reach a partial accepting ID for t_i. Conversely, if $M[\Omega]$ cannot reach a partial accepting ID, then there cannot exist a substitution that we are looking for since no instance of t_i will not be accepted by $M[\Omega]$. The following procedure is based on this idea. The step 3 of the procedure is a means to deal with variables that have multiple occurrences in $t_1, \ldots, t_n$.

Procedure 1.

Step 1. Compute the mgu $\sigma'_{\mathbf{mgu}}$ of g_1 and g_2. If there is no mgu of g_1 and g_2, then output "not A-unifiable" and halt. If $\sigma'_{\mathbf{mgu}}$ is found, then rename variables in $\sigma'_{\mathbf{mgu}}$ and obtain $\sigma_{\mathbf{mgu}} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ such that $\mathcal{V}(t_1) \cup \cdots \mathcal{V}(t_n) = \{v_1, \ldots, v_m\}$ and $\{v_1, \ldots, v_m\} \cap \{x_1, \ldots, x_n\} = \emptyset$. For the following discussion, let $p_i^1, \ldots, p_i^{d_i} \in \mathcal{O}(t_i)$ $(1 \leq i \leq n, d_i \geq 0)$ be variable occurrences of t_i, e_h $(1 \leq h \leq m)$ be the total number of occurrences of the variable v_h in $t_1, \ldots, t_n$, and assume that all occurrences of v_h are uniquely number from 1 to e_h. To make discussion succinctly, we introduce two functions α and β such that if the k-th occurrence of the variable v_h $(1 \leq h \leq m, 1 \leq k \leq e_h)$ is $p_i^j \in \mathcal{O}(t_i)$, then $\alpha(h,k) = i$ and $\beta(h,k) = j$.

Step 2. For each i with $1 \leq i \leq n$, nondeterministically choose a partial accepting ID I_i of $M[\Omega]$ for t_i such that $[\Omega]\lceil t_i \rceil \vdash^* I_i$. Since t_i has variable occurrences $p_i^1, \ldots, p_i^{d_i}$, there are states $[l_i^1], \ldots, [l_i^{d_i}]$ such that $I_i/p_i^j = [l_i^j]\lceil t_i/p_i^j \rceil$ for $1 \leq j \leq d_i$. If there is no such ID for some i $(1 \leq i \leq n)$, then output "not A-unifiable" and halt.

Step 3. If $\bigcap_{k=1}^{e_h} L(M[l_{\alpha(h,k)}^{\beta(h,k)}]) \neq \emptyset$ for every h $(1 \leq h \leq m)$, then output "A-unifiable" and halt. Otherwise, output "not A-unifiable" and halt.

Lemma 7. *Goal terms g_1 and g_2 are A-unifiable if and only if Proc. 1 outputs A-unifiable.*

*Proof (Only if part).*Let $\rho = \{x_1 \mapsto r_1, \ldots, x_n \mapsto r_n \mid r_i$ is an A-term for $1 \leq i \leq n\}$ be an A-substitution that unifies g_1 and g_2. Since g_1 and g_2 are unifiable, there are the mgu $\sigma_{\mathrm{mgu}} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ which is computed at Step 1, and a substitution $\sigma = \{v_1 \mapsto s_1, \ldots, v_m \mapsto s_m\}$ such that $r_i = t_i\sigma$ for $1 \leq i \leq n$. Since r_i $(1 \leq i \leq n)$ is an A-term, $M[\Omega]$ accepts $\underline{r_i}$ by Lemma 6. Hence, for each i $(1 \leq i \leq n)$, there are an ID I_i of $M[\Omega]$ for $\underline{r_i}$ such that

$$[\Omega]\lceil \underline{r_i} \rceil \vdash^* I_i \tag{3}$$

and states $[l_i^j]$ $(1 \leq j \leq d_i)$ such that $I_i/p_i^j = [l_i^j]\lceil \underline{r_i/p_i^j} \rceil$ for $1 \leq j \leq d_i$. Furthermore, there is an accepting ID I_i' for $\underline{r_i}$ such that

$$I_i \vdash^* I_i'. \tag{4}$$

Remark that t_i is a term that is obtained from r_i by replacing subterms of r_i at occurrences p_i^j with variables for $1 \leq j \leq d_i$. By applying Lemma 5(2) at occurrences p_i^j $(1 \leq j \leq d_i)$ in (3),

$$[\Omega]\lceil t_i \rceil \vdash^* I_i''$$

where I_i'' is a partial accepting ID for t_i such that $I_i''/p_i^j = [l_i^j]\lceil t_i/p_i^j \rceil$, which can be chosen at Step 2 of Proc. 1. On the other hand, by applying Lemma 5(1) at occurrences p_i^j $(1 \leq j \leq d_i)$ in (4), we have $[l_i^j]\lceil \underline{r_i/p_i^j} \rceil \vdash^* I_i'/p_i^j$ for $1 \leq j \leq d_i$. Since I_i'/p_i^j is an accepting ID for $\underline{r_i/p_i^j}$,

$$\underline{r_i/p_i^j} \in L(M[l_i^j]) \qquad (1 \leq i \leq n, 1 \leq j \leq d_i). \tag{5}$$

By the definition of α and β, $1 \leq \alpha(h,k) \leq n$ and $1 \leq \beta(h,k) \leq d_{\alpha(h,k)}$ hold for $1 \leq h \leq m$ and $1 \leq k \leq e_h$. Thus (5) is written as

$$\underline{r_{\alpha(h,k)}/p_{\alpha(h,k)}^{\beta(h,k)}} \in L(M[l_{\alpha(h,k)}^{\beta(h,k)}]) \qquad (1 \leq h \leq m, 1 \leq k \leq e_h). \tag{6}$$

Furthermore, by the definition of α and β, the subterm of $t_{\alpha(h,k)}$ at the occurrence $p_{\alpha(h,k)}^{\beta(h,k)}$ is the variable v_h, and thus, the subterm of $r_{\alpha(h,k)}$ at the occurrence $p_{\alpha(h,k)}^{\beta(h,k)}$ must be s_h. Therefore (6) is written as

$$\underline{s_h} \in L(M[l_{\alpha(h,k)}^{\beta(h,k)}]) \qquad (1 \leq h \leq m, 1 \leq k \leq e_h).$$

This implies that $\underline{s_h}$ belongs to $\bigcap_{k=1}^{e_h} L(M[l_{\alpha(h,k)}^{\beta(h,k)}])$ for $1 \leq h \leq m$, and Proc. 1 outputs "A-unifiable" and halts.

*(If part).*If Proc. 1 outputs "A-unifiable", then (i) there is the mgu $\sigma_{\mathbf{mgu}} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ of g_1 and g_2; (ii) for each i $(1 \leq i \leq n)$, there is a partial accepting ID I_i for t_i such that $I_i/p_i^j = [l_i^j]\lceil t_i/p_i^j \rceil$ $([l_i^j] \in Q)$ for $1 \leq j \leq d_i$ and

$$[\Omega]\lceil t_i \rceil \vdash^* I_i \qquad (1 \leq i \leq n); \tag{7}$$

and (iii) $\bigcap_{k=1}^{e_h} L(M[l_{\alpha(h,k)}^{\beta(h,k)}]) \neq \emptyset$ for $1 \leq h \leq m$. By the last condition (iii), there are terms s_h $(1 \leq h \leq m)$ such that

$$\underline{s_h} \in L(M[l_{\alpha(h,k)}^{\beta(h,k)}]) \qquad (1 \leq h \leq m, 1 \leq k \leq e_h). \tag{8}$$

Let $\sigma = \{v_1 \mapsto s_1, \ldots, v_m \mapsto s_m\}$ and $r_i = t_i\sigma$ for $1 \leq i \leq n$. Since $\sigma_{\mathbf{mgu}}$ is the mgu of g_1 and g_2, it is obvious that $\sigma_{\mathbf{mgu}}\sigma = \{x_1 \mapsto r_1, \ldots, x_n \mapsto r_n\}$ also unifies g_1 and g_2. In the following, we show that each r_i $(1 \leq i \leq n)$ is an A-term. By applying Lemma 5(2) at occurrences p_i^j $(1 \leq j \leq d_i)$ in (7),

$$[\Omega]\lceil \underline{r_i} \rceil \vdash^* I_i' \tag{9}$$

where I_i' is an ID for $\underline{r_i}$ such that

$$I_i'/p_i^j = [l_i^j]\lceil \underline{r_i/p_i^j} \rceil \qquad (1 \leq j \leq d_i). \tag{10}$$

Since p_i^j $(1 \leq i \leq n, 1 \leq j \leq d_i)$ are variable occurrences of t_i, there are integers b_{ij} $(1 \leq b_{ij} \leq m)$ and c_{ij} $(1 \leq c_{ij} \leq e_{b_{ij}})$ such that $\alpha(b_{ij}, c_{ij}) = i$ and $\beta(b_{ij}, c_{ij}) = j$. That is, t_i/p_i^j is the c_{ij}-th occurrence of the variable $v_{b_{ij}}$, and thus r_i/p_i^j must be $s_{b_{ij}}$. By using (8),

$$\underline{r_i/p_i^j} = \underline{s_{b_{ij}}} \in L(M[l_{\alpha(b_{ij},c_{ij})}^{\beta(b_{ij},c_{ij})}]) = L(M[l_i^j]) \qquad (1 \leq i \leq n, 1 \leq j \leq d_i). \tag{11}$$

By (9), (10) and (11), it follows that the ID $[\Omega]\lceil \underline{r_i} \rceil$ can reach an accepting ID for $\underline{r_i}$. This means that $M[\Omega]$ accepts $\underline{r_i}$ for $1 \leq i \leq n$, and r_i is an A-term by Lemma 6. Summarizing the above discussion, g_1 and g_2 are A-unifiable by the A-substitution $\sigma_{\mathbf{mgu}}\sigma$. □

Theorem 8. *SUPCS for the classes 2Bb, 2Ba, 2Ab, 2Aa, 1Bb, 1Ba, 1Ab and 1Aa is decidable.*

5.3 Intractability Result for the Class 1Ab

3SAT problem is polynomial time reducible to SUPCS for the class 1Ab. Let E be a given 3-conjunctive normal form boolean formula. Without loss of generality, we assume that E contains m variables $v_1, \ldots v_m$, each of which has $p_1, \ldots, p_m$ occurrences in E, respectively. The formula E is written as $E = C_1 \wedge \cdots \wedge C_n$ where C_i $(1 \leq i \leq n)$ is a clause that is written as $C_i = (u_i^1 \vee u_i^2 \vee u_i^3)$ where u_i^j $(1 \leq j \leq 3)$ is either v_k or $\neg v_k$ with $1 \leq k \leq m$.

Define a set F of function symbols to be $F = \{\mathrm{ROOT}, \mathrm{OR}, N, T\} \cup \{V_k \mid 1 \leq k \leq m\}$ where $a(\mathrm{ROOT}) = m + n$, $a(\mathrm{OR}) = 3$, $a(N) = 1$, $a(T) = 0$, and $a(V_k) = p_k$ for $1 \leq k \leq m$. We use variables x_k^l $(1 \leq k \leq m, 1 \leq l \leq p_k)$, y_i $(1 \leq i \leq n)$ and z_k $(1 \leq k \leq m)$ to define goal terms. For each $1 \leq i \leq n$ and $1 \leq j \leq 3$, define terms $\langle u_i^j \rangle$ and $\langle C_i \rangle$ as follows.

$$\langle u_i^j \rangle = \begin{cases} x_k^l & \text{if } u_i^j = v_k, \text{ and it is the } l\text{-th occurrence} \\ & \qquad \text{of } v_k \text{ in } E \text{ from the left,} \\ N(x_k^l) & \text{if } u_i^j = \neg v_k, \text{ and it is the } l\text{-th occurrence} \\ & \qquad \text{of } v_k \text{ in } E \text{ from the left,} \end{cases}$$

$$\langle C_i \rangle = \mathrm{OR}(\langle u_i^1 \rangle, \langle u_i^2 \rangle, \langle u_i^3 \rangle).$$

Goal terms g_1 and g_2 are constructed as follows.

$$g_1 = \mathrm{ROOT}(\langle C_1 \rangle, \ldots, \langle C_n \rangle, z_1, \ldots, z_m),$$
$$g_2 = \mathrm{ROOT}(y_1, \ldots, y_n, V_1(x_1^1, \ldots, x_1^{p_1}), \ldots, V_m(x_m^1, \ldots, x_m^{p_m})).$$

For example, if $E = (v_1 \vee v_2 \vee \neg v_3) \wedge (\neg v_3 \vee \neg v_1 \vee v_3)$, then

$$g_1 = \mathrm{ROOT}(\mathrm{OR}(x_1^1, x_2^1, N(x_3^1)), \mathrm{OR}(N(x_3^2), N(x_1^2), x_3^3), z_1, z_2, z_3),$$
$$g_2 = \mathrm{ROOT}(y_1, y_2, V_1(x_1^1, x_1^2), V_2(x_2^1), V_3(x_3^1, x_3^2, x_3^3)).$$

The set A of terms is defined as follows.

$$\begin{aligned} A = &\{T, N(T)\} \cup \{V_k(T, \ldots, T), V_k(N(T), \ldots, N(T)) \mid 1 \leq k \leq m\} \\ &\cup \{\mathrm{OR}(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in \{T, N(T), N(N(T))\} \\ &\setminus \{\mathrm{OR}(N(T), N(T), N(T))\} \end{aligned}$$

Remark that the above construction of g_1, g_2 and A is possible in linear time to the size of E, and that the triple $\langle g_1, g_2, A \rangle$ belongs to the class 1Ab.

Lemma 9. *The formula E is satisfiable if and only if g_1 and g_2 are A-unifiable.*

Proof (Only if part). Let $\rho = \{v_1 \mapsto b_1, \ldots, v_m \mapsto b_m \mid b_i \ (1 \leq i \leq m)$ is either true or false$\}$ be an assignment to variables in E which makes E true. For this ρ, define an A-substitution $\sigma_x = \{x_k^l \mapsto t_k \mid 1 \leq k \leq m, 1 \leq l \leq p_k\}$ where t_k $(1 \leq k \leq m)$ is the A-term T if b_k is true, and $N(T)$ if b_k is false. Also define an A-substitution $\sigma_z = \{z_k \mapsto V_k(t_k, \ldots, t_k) \mid 1 \leq k \leq m\}$ and a substitution $\sigma_y = \{y_i \mapsto \langle C_i \rangle \sigma_x \mid 1 \leq i \leq n\}$. It is easily verified that g_1 and g_2 are unifiable by the substitution $\sigma = \sigma_x \cup \sigma_y \cup \sigma_z$. To say g_1 and g_2 are A-unifiable, it suffices to show that σ_y is an A-substitution, that is, $\langle C_i \rangle \sigma_x$ is an A-term for $1 \leq i \leq n$. By the definitions of $\langle C_i \rangle$ and σ_x, $\langle C_i \rangle \sigma_x$ is written as $\langle C_i \rangle \sigma_x = \mathrm{OR}(w_i^1, w_i^2, w_i^3)$ where $w_i^1, w_i^2, w_i^3 \in \{T, N(T), N(N(T))\}$. By the definition of A, $\langle C_i \rangle \sigma_x$ is not an A-term if and only if $w_i^1 = w_i^2 = w_i^3 = N(T)$. Observe that $w_i^1 = N(T)$ if either one of the followings holds.

- $\langle u_i^1 \rangle = x_k^l$ $(1 \leq k \leq m, 1 \leq l \leq p_k)$ and $t_k = N(T)$; this implies that the literal u_i^1 of E is v_k and false is assigned to v_k under ρ.

– $\langle u_i^1 \rangle = N(x_k^l)$ $(1 \leq k \leq m, 1 \leq l \leq p_k)$ and $t_k = T$; this implies that the literal u_i^1 of E is $\neg v_k$ and true is assigned to v_k under ρ.

In either case, the literal u_i^1 in E becomes false under the assignment ρ. Similar discussions hold for u_i^2 and u_i^3, and therefore $w_i^1 = w_i^2 = w_i^3 = N(T)$ implies that all of literals u_i^1, u_i^2 and u_i^3 become false under the assignment ρ. This contradicts the assumption that ρ makes E true. Thus, $\langle C_i \rangle \sigma_x$ is an A-term for $1 \leq i \leq n$.

(If part). Let σ be an A-substitution such that $g_1\sigma = g_2\sigma$. Without loss of generality, we assume that σ is written as $\sigma = \{x_k^l \mapsto t_k^l \mid 1 \leq k \leq m, 1 \leq l \leq p_k\} \cup \{y_i \mapsto c_i \mid 1 \leq i \leq n\} \cup \{z_k \mapsto s_k \mid 1 \leq k \leq m\}$. Remark that s_k $(1 \leq k \leq m)$ must be an A-term of which root symbol is V_k, and hence $s_k = V_k(T, \ldots, T)$ or $s_k = V_k(N(T), \ldots, N(T))$. This implies that all of t_k^l $(1 \leq l \leq p_k)$ are T, or all of them are $N(T)$ (otherwise, subterms of $g_1\sigma$ and $g_2\sigma$ at occurrences $n+1, \ldots, n+m$ cannot be the same). Define a boolean assignment ρ to variables in E as $\rho = \{v_1 \mapsto b_1, \ldots, v_m \mapsto b_m\}$ where b_k $(1 \leq k \leq m)$ is defined to be true if all of t_k^l $(1 \leq l \leq p_k)$ are T, and false if all of them are $N(T)$. We show that this assignment ρ makes E true. The proof is by contradiction. Assume that E is false under the assignment ρ, then there is i $(1 \leq i \leq n)$ such that the clause C_i is false, that is, all of literals u_i^1, u_i^2 and u_i^3 are false under ρ. Remark that u_i^1 is false if one of the followings holds.

– $u_i^1 = v_k$ $(1 \leq k \leq m)$ and b_k is false; in this case, $\langle u_i^1 \rangle = x_k^l$ $(1 \leq l \leq p_k)$ and t_k^l is $N(T)$.
– $u_i^1 = \neg v_k$ $(1 \leq k \leq m)$ and b_k is true; in this case, $\langle u_i^1 \rangle = N(x_k^l)$ $(1 \leq l \leq p_k)$ and t_k^l is T.

In either case, $\langle u_i^1 \rangle \sigma = N(T)$. Similar discussions hold for u_i^2 and u_i^3. Thus, if all of literals u_i^1, u_i^2 and u_i^3 are false under ρ, then $\langle u_i^1 \rangle \sigma = \langle u_i^2 \rangle \sigma = \langle u_i^3 \rangle \sigma = N(T)$, and $\langle C_i \rangle \sigma = \mathrm{OR}(N(T), N(T), N(T))$. On the other hand, since σ is an A-substitution such that $g_1\sigma = g_2\sigma$, $\langle C_i \rangle \sigma = y_i\sigma = c_i$ for $1 \leq i \leq n$. If $c_i = \langle C_i \rangle \sigma = \mathrm{OR}(N(T), N(T), N(T))$, then c_i does not belong to A, a contradiction.□

Theorem 10. *SUPCS for 2Bb, 2Ba, 2Ab, 1Bb, 1Ba and 1Ab is $\mathcal{NP}$-hard.*

Since SUPCS obviously belongs to $\mathcal{NP}$ if all terms in A are ground, we have the following corollary.

Corollary 11. *SUPCS for 1Bb, 1Ba and 1Ab is $\mathcal{NP}$-complete.*

5.4 Polynomial-Time Procedure for the Class 2Aa

Proc. 1 introduced in Sect. 5.2 is somewhat complicated since it must deal with variables which have multiple occurrences in the co-domain of the mgu $\sigma_{\mathbf{mgu}} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. If an instance of SUPCS belongs to the class 2Aa or 1Aa, then no variable occurs multiple time in $t_1, \ldots, t_n$. In this case, SUPCS is solvable more efficiently by the following Procedure 2.

Procedure 2.

Step 1. Compute the mgu $\sigma_{\mathrm{mgu}} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ of g_1 and g_2.
Step 2. For each $1 \leq i \leq n$, construct a tree automaton M_i that accepts Ω-terms of (the usual) instances of t_i. This is possible since t_i is linear. The construction is similar to the construction of tree automata in Sect. 5.2.
Step 3. If $L(M_i) \cap L(M[\Omega]) \neq \emptyset$ holds for $1 \leq i \leq n$, then output "A-unifiable", otherwise output "not A-unifiable".

Steps 1, 2 and 3 of Proc. 2 are executed in $O(m)$, $O(m^2)$ and $O(m^8)$-time, respectively, where m is the size of an instance of SUPCS. If the procedure outputs "A-unifiable", then there are r_i $(1 \leq i \leq n)$ such that $\underline{r_i} \in L(M_i) \cap L(M[\Omega])$ which implies that r_i is an A-term and is an instance of t_i. Thus, g_1 and g_2 are A-unifiable by an A-substitution $\sigma = \{x_1 \mapsto r_1, \ldots, x_n \mapsto r_n\}$.

Theorem 12. *SUPCS for the classes 2Aa and 1Aa belongs to $\mathcal{P}$.*

5.5 Concluding Remarks

The decidability and computational complexity of SUPCS for various classes of inputs are discussed. The results are summarized in Fig. 1. These results will give an insight into decidability and computational complexity of E-unification problems under constrained substitutions.

References

1. Baader, F. and Siekmann, J.H.: "Unification Theory", *Handbook of Logic in Artificial Intelligence and Logic Programming*, **2**, Oxford Univ. Press, pp.41–125 (1994).
2. Dershowitz, N. and Sivakumar, G.: "Solving Goals in Equational Languages", Proc. First Workshop of CTRS, Orsay, France, , LNCS **308**, pp.45–55 (1987).
3. Dolev, D., Even, S. and Karp, R.M.: "On the Security of Ping-Pong Protocols", Information and Control, **55**, 1–3, pp.57–68 (1982).
4. Fujiwara, T., Taniguchi, K. and Kasami, T.: "Decision Problem of the Security for Cryptographic Protocols", Trans. of IEICE, Institute of Electronics, Information and Communication Engineers, Tokyo, **J69-D**, 6, pp.984–992 (1986, in Japanese).
5. Gécseq, F. and Steinby, M.: *Tree Automata*, Akadémiai Kiadó, Budapest (1984).
6. Hopcroft, J.E. and Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley (1979).
7. Hullot, J.M: "Canonical Forms and Unification", Proc. Fifth International Conference on Automated Deduction, Les Arcs, France, LNCS **87**, pp.318–334 (1980).
8. Kaji, Y., Fujiwara, T. and Kasami, T: "Solving a Unification Problem under Constrained Substitutions Using Tree Automata", Proc. Fourteenth Conference on FST & TCS, Madras, India, LNCS **880**, pp.276–287 (1994).
9. Paterson, M.S. and Wegman, M.N.: "Linear Unification", Journal of Computer and System Sciences, **16**, 2, pp.158–167 (1978).