Unification of Multisets with Multiple Labelled Multiset Variables *

Zan Naeem[1] and Giselle Reis[1]

Carnegie Mellon University, Doha, Qatar
znaeem@andrew.cmu.edu, giselle@cmu.edu

Abstract

We look into the problem of unifying multisets containing (first-order) terms and *multiple* multiset variables. The variables are labelled, meaning that a unifier that places a term in a multiset variable M_i is different from another that places a term in a multiset variable M_j, for $i \neq j$. We describe a sound, complete, and terminating algorithm for computing the set of all possible unifiers, and analyse its complexity. We also prove an input pre-processing step that avoids the computation of less general unifiers.

1 Introduction

Multiset is an important data-structure that is used to specify various object systems. Our motivation stems mostly from proof theory, where logical entailment is encoded as sequents $\Gamma \vdash \Delta$, where both Γ and Δ are typically considered as multisets. When reasoning about such objects, one might need to use an implementation of sets/multisets based on lists, since most reasoning tools (i.e., logical frameworks, proof assistants, and logic programming languages) do not have built-in support for these data structures [2, 3, 6, 7]. Adding this kind of support requires, among other things, a unification algorithm.

Multiset unification was studied in [1, 4], where the authors propose solutions for the problem of unifying multisets with at most one multiset variable. We extend those results for multisets with multiple multiset variables. In our setting, each multiset variable is *labelled*, meaning that assigning a term to either a multiset M_i or M_j, where $i \neq j$, should be considered different solutions. We describe a terminating algorithm and analyse its complexity. Moreover, we prove that a simple modification of our algorithm avoids the computation of less general unifiers.

The need for labelled multiset variables emerged when reasoning with multiplicative rules in sequent calculi, such as:

$$\frac{\Gamma_1 \vdash A \quad \Gamma_2, B \vdash C}{\Gamma_1, \Gamma_2, A \rightarrow B \vdash C} \rightarrow_l$$

To apply this rule to, e.g., the sequent $\Gamma, D, A \rightarrow B \vdash C$, where A, B, C, and D are formulas, we need to unify its antecedent $\Gamma, D, A \rightarrow B$ with $\Gamma_1, \Gamma_2, A \rightarrow B$. Assigning formula D to Γ_1 or Γ_2 should be considered two different solutions, since they result in two different applications of the rule.

1.1 Preliminaries

A *multiset $\mathcal{M}$ with multiple labelled multiset variables* is denoted by $\{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}$. Each t_i is a *term* ranging over a first-order language $\mathcal{L} = \langle \Sigma, \mathcal{V} \rangle$, where Σ is a set of constants

*This publication was made possible by the support of Qatar Foundation through Carnegie Mellon University in Qatar's Seed Research program. The statements made herein are solely the responsibility of the authors.

and function symbols, and $\mathcal{V}$ is a denumerable set of *term variables*. Each M_i is a *multiset variable* ranging over a denumerable set $\mathcal{V}_M$ of multiset variables. When n and k are not relevant, we abbreviate $t_1, ..., t_n$ as $\bar{t}$ and $M_1, ..., M_k$ as $\overline{M}$. When $k = 0$, we write the multiset as $\{\!\!\{t_1, ..., t_n\}\!\!\}$. Henceforth, we refer to multisets with multiple labelled multiset variables as *mmsets* for brevity.

Definition 1 (Substitution)**.** *An* mmset substitution σ *is a finite mapping of term variables to terms, and of multiset variables to mmsets. The application of a substitution σ to an mmset $\{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}$ is defined as:*

$$\{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}\sigma = \{\!\!\{t_1\sigma, ..., t_n\sigma\}\!\!\} \uplus M_1\sigma \uplus ... \uplus M_k\sigma$$

where each $t_i\sigma$ is the usual first-order substitution and $\uplus$ is left-associative and defined as:

$$\{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\} \uplus \{\!\!\{s_1, ..., s_m | N_1, ..., N_l\}\!\!\} = \{\!\!\{t_1, ..., t_n, s_1, ..., s_m | M_1, ..., M_k, N_1, ..., N_l\}\!\!\}$$

Definition 2 (Equality)**.** *Mmsets $\mathcal{M} = \{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}$ and $\mathcal{N} = \{\!\!\{s_1, ..., s_m | N_1, ..., N_l\}\!\!\}$ are considered* equal modulo a constraint theory $\mathcal{T}$, *written $\mathcal{M} =_\mathcal{T} \mathcal{N}$ iff: $n = m$ and $t_1, ..., t_n$ is a permutation of $s_1, ..., s_m$; and $\mathcal{T} \vdash M_1 \cup ... \cup M_k \equiv N_1 \cup ... \cup N_l$.*

2 Mmsets Unification

The *mmset unification problem* of mmsets $\mathcal{M}_1$ and $\mathcal{M}_2$ consists of finding a substitution σ and constraint theory $\mathcal{T}_\sigma$ such that $\mathcal{M}_1\sigma =_{\mathcal{T}_\sigma} \mathcal{M}_2\sigma$. The theory $\mathcal{T}_\sigma$ consists of an equality over unions of multiset variables, and it is computed *a posteriori* for each unifier σ.

2.1 Algorithm

In what follows, we use σ to denote a single substitution, Σs to denote sets of substitutions, $\times$ for the Cartesian product of two sets (or lists), and $\setminus$ for multiset difference. The pseudo code for all algorithms are listed in Appendix A, and an implementation in SML can be found at https://github.com/meta-logic/mmset-unif.

The main function for mmset unification is implemented by Algorithm 1. In the most general case (lines 11 to 17), the unifiers of $\{\!\!\{\bar{t}|\overline{M}\}\!\!\}$ and $\{\!\!\{\bar{s}|\overline{N}\}\!\!\}$ are computed by choosing a subset of terms (of the same size) from $\bar{t}$ and $\bar{s}$ to be unified, and distributing the rest among the multiset variables $\overline{M}$ and $\overline{N}$. The number of terms chosen to be unified can vary from 0 to the minimum length of $\bar{t}$ and $\bar{s}$. Two other cases are considered separately for efficiency purposes. The first one is when there are no multiset variables (line 1). Here a unification is only possible if $|\bar{t}| = |\bar{s}|$. The second case is when one of the mmsets does not have multiset variables (lines 4 and 7). If $\overline{M}$ is empty, then all terms in $\bar{s}$ must be unified with a term from $\bar{t}$. The remaining terms in $\bar{t}$ can be allocated in $\overline{N}$.

Function $\mathsf{unify\_c}$ (Algorithm 2) chooses c terms from the multisets $\bar{t}$ and $\bar{s}$ to be unified, and distributes the rest of the terms among the multiset variables. The function $\mathsf{choose}(F, c)$ returns a set of tuples (F_c, F_r), where F_c is the multiset with the chosen c elements (thus $|F_c| = c$), and $F_r = F \setminus F_c$. Unifiers for the chosen terms are computed by $\mathsf{unify\_terms}$ and stored in Σ_t. Substitutions for each context variable, containing the remaining terms, are computed by $\mathsf{unify\_distribute}$ and stored in Σ_M. The final set of unifiers consists of the composition $\sigma_M\sigma_t$ for each $(\sigma_M, \sigma_t) \in \Sigma_M \times \Sigma_t$. Note that, since the image of σ_M may contain terms, the composition needs to be in this order.

Function unify_terms (Algorithm 3) finds all unifiers of two multisets of terms (without multiset variables) of equal length. This is done by testing all possible pairings of the two multisets, obtained by pairing some order of the first multiset with all possible permutations of the second one. For each pairing, the function unify_lists computes the most general unifier.

Function unify_distribute (Algorithm 4) computes unifiers for the mmsets $\{\!\!\{\bar{t}|\overline{M}\}\!\!\}$ and $\{\!\!\{\bar{s}|\overline{N}\}\!\!\}$ considering that all $\bar{t}$ occurs in $\overline{N}$ and all $\bar{s}$ occurs in $\overline{M}$. Let Σ_N denote the substitutions that distribute $\bar{t}$ into $\overline{N}$, and Σ_M the substitutions that distribute $\bar{s}$ into $\overline{M}$. The resulting substitutions are $\sigma_M \cup \sigma_N$ for each $(\sigma_M, \sigma_N) \in \Sigma_M \times \Sigma_N$. A simple union can be used in the case, as the image of σ_M is disjoint from the domain of σ_N (see below).

Function distribute (Algorithm 5) is used by unify_distribute and that is where the aforementioned Σ_N and Σ_M are computed. It computes substitutions for the multiset variables $N_1, ..., N_l$ such that all terms $t_1, ..., t_n$ occur in one of the multisets. This is done by calculating all ordered l-partitions of the multiset $\{\!\!\{t_1, ..., t_n\}\!\!\}$. For example, the ordered 2-partitions of the multiset $\{\!\!\{a, b, c\}\!\!\}$ are:

$$\begin{array}{llll} [\{\!\!\{a,b,c\}\!\!\},\{\!\!\{\}\!\!\}] & [\{\!\!\{a,b\}\!\!\},\{\!\!\{c\}\!\!\}] & [\{\!\!\{a\}\!\!\},\{\!\!\{b,c\}\!\!\}] & [\{\!\!\{a,c\}\!\!\},\{\!\!\{b\}\!\!\}] \\ [\{\!\!\{\}\!\!\},\{\!\!\{a,b,c\}\!\!\}] & [\{\!\!\{c\}\!\!\},\{\!\!\{a,b\}\!\!\}] & [\{\!\!\{b,c\}\!\!\},\{\!\!\{a\}\!\!\}] & [\{\!\!\{b\}\!\!\},\{\!\!\{a,c\}\!\!\}] \end{array}$$

Each computed substitution corresponds to an l-ordered partition. If there are no terms ($n = 0$), then there is only the trivial partition of l empty multisets. In this case, the algorithm returns a list with one substitution, which maps every multiset variable N_i to the mmset with no terms. If there are no multiset variables ($l = 0$), then there are no partitions, and thus no possible substitutions. The exceptional case is when there are no terms nor multiset variables ($l = n = 0$). In this case, the solution is the set containing only the empty substitution ($\{\}$).

The last parameter of distribute indicates whether the multiset variables should contain *exactly* the terms $t_1, ..., t_n$. If set to true, then there is no more space for other terms, and N_i is mapped to an mmset with the appropriate terms and no multiset variables. Otherwise it is mapped to an mmset with a set of terms and a *fresh* multiset variable. If there are no terms to place in the multiset variable, it is mapped to itself (to avoid unecessary renamings). This is needed to compute the constraint theory, after which such identity substitutions can be eliminated.

Constraint theory For a unifier σ, the constraint theory $\mathcal{T}_\sigma$ is defined as:

$$\bigcup\{M'_i \mid M_i \mapsto \{\!\!\{ts|M'_i\}\!\!\} \in \sigma\} \equiv \bigcup\{N'_i \mid N_i \mapsto \{\!\!\{ts|N'_i\}\!\!\} \in \sigma\}$$

Soundness and completeness of the algorithm are straightforward, since it exhaustively checks all possibilities for unifying multisets.

Theorem 1. *Soundness If* unify$(\mathcal{M}, \mathcal{N}) \mapsto \{\sigma_1, ..., \sigma_n\}$ *then* $\forall \sigma_i . \mathcal{M}\sigma_i =_{\mathcal{T}_{\sigma_i}} \mathcal{N}\sigma_i$

Theorem 2. *Completeness If* $\exists \sigma . \mathcal{M}\sigma =_{\mathcal{T}_\sigma} \mathcal{N}\sigma$ *then* unify$(\mathcal{M}, \mathcal{N}) \mapsto \{\sigma_1, ..., \sigma_n\}$ *and* $\exists \sigma_i$ *such that* $\sigma = \sigma_i \sigma'$.

Note that the use of a substitution σ' is needed even if the set of computed unifiers is not the minimal one.

2.2 Complexity

The most expensive part of the unification algorithm is the one between lines 11 and 17 in Algorithm 1, so we concentrate our complexity analysis to that case. For each i from 0 to the minimum number of terms, $\mathsf{unify\_c}$ is called. This function has two nested loops over the sets T_c and S_c (Alg. 2, lines 4, 5), which contain all possible ways of choosing i elements from n and m, respectively. Thus $|T_c| = \binom{n}{i}$ and $|S_c| = \binom{m}{i}$. In the inner part of the loops, $\mathsf{unify\_terms}$ is called, which finds all possible unifiers for two multisets of size i. Since all possible pairings of elements must be tried, and for each order $\mathsf{unify\_lists}$ runs in i^2, the function (Alg. 3) has complexity $i^2 i!$. The function $\mathsf{unify\_distribute}$ (Alg. 4) computes all possible ways of partitioning $n - i$ elements into l parts, and $m - i$ elements into k parts, and returns the Cartesian product of these sets. Therefore, its complexity is $l^{n-i}\ k^{m-i}$.

Putting those together, we get to the cost for the unification of mmsets $\{\!|t_1, ..., t_n|M_1, ..., M_k|\!\}$ and $\{\!|s_1, ..., s_m|N_1, ..., N_l|\!\}$:

$$\sum_{i=0}^{\mathsf{min}(n,m)} \binom{n}{i}\binom{m}{i}\ i^2\ i!\ l^{n-i}\ k^{m-i}$$

After some arithmetic manipulation, we can conclude that, on the worst case, unify runs in $O(n!\ m!\ l^n\ k^m)$. For the special case where the multisets have only one multiset variable: $l = k = 1$, and the unification algorithm runs in $O(n!\ m!)$.

2.3 Removing Less General Unifiers

The algorithm described in Section 2.1 does not compute the set of minimal unifiers. For example, given multiset $\{\!|a, a|M|\!\}$ and $\{\!|a|N|\!\}$, unify computes three unifiers with constraint: $\{N \mapsto \{\!|a|N'|\!\}\}$, where $M \equiv N'$, twice (once for each occurrence of a), and $\{M \mapsto \{\!|a|M'|\!\}\ ;\ N \mapsto \{\!|a, a|N'|\!\}\}$, where $M' \equiv N'$. These are the same unifiers obtained by the non-deterministic algorithm from [5].

In order to reduce the number of less general unifiers, we can remove every pair of equal terms t_i and s_j from the mmsets (i.e., t_i and s_j unify with the empty substitution). The rationale behind this is that, every other unifier that is obtained by unifying these terms with something else, or placing them in a multiset variable, can be recovered from the set of unifiers obtained when this pair is not in the mmset.

We start by showing that it is safe to eliminate pairs of equal terms from the problem of unifying multisets without multiset variables.

Theorem 3. *Let $\bar{t}$ and $\bar{s}$ be two multisets of terms such that $t_a \in \bar{t}$ and $s_b \in \bar{s}$ are equal, for some a and b. Let $\Sigma_{all} = \mathsf{unify\_terms}(\bar{t}, \bar{s})$, and $\Sigma = \mathsf{unify\_terms}(\bar{t} \backslash \{\!|t_a|\!\}, \bar{s} \backslash \{\!|s_b|\!\})$. Then for every $\sigma \in \Sigma_{all}$, there exists $\mu \in \Sigma$ s.t. $\sigma = \mu\sigma'$ for some substitution σ'.*

The proof for this theorem can be found in Appendix B. The overall idea is as follows. σ was obtained by some pairing of terms in $\bar{t}$ and $\bar{s}$. We choose μ as the unifier that used a pairing that is as close as possible as the one used for σ. Those pairings differ only for the terms involving t_a and s_b. Suppose t_a is paired with s_x and s_b is paired with t_y. Using the most general unifiers of t_y and s_x, we can conclude the existence of σ' such that $\sigma = \mu\sigma'$.

Theorem 4. *Let $\{\!|\bar{t}|\overline{M}|\!\}$ and $\{\!|\bar{s}|\overline{N}|\!\}$ be two mmsets such that $t_a \in \bar{t}$ is equal to $s_b \in \bar{s}$ for some a and b. Moreover, let $\Sigma_{all} = \mathsf{unify}(\{\!|\bar{t}|\overline{M}|\!\}, \{\!|\bar{s}|\overline{N}|\!\})$, and $\Sigma = \mathsf{unify}(\{\!|\bar{t} \backslash \{\!|t_a|\!\}|\overline{M}|\!\}, \{\!|\bar{s} \backslash \{\!|s_b|\!\}|\overline{N}|\!\})$. Then for every $\sigma \in \Sigma_{all}$ there exists $\mu \in \Sigma$ such that $\sigma = \mu\sigma'$ for some σ'.*

The proof for this theorem can be found in Appendix B. We proceed by a case analysis on whether t_a and s_b were chosen to be unified, or to be placed in a multiset variable. There are four cases. The case in which both are chosen to be unified is solved using Theorem 3. For the case in which both are placed in a multiset variable, we can construct σ'. The other two (dual) cases are the more involved ones. They use a combination of the two strategies of the first cases.

This modification is implemented in the algorithm available online, and extensive testing has shown that all less general unifiers are eliminated. In particular, only the unifier $\{N \mapsto \{\!|a|N'|\!\}\}$, where $M \equiv N'$ is computed for mmsets $\{\!|a, a|M|\!\}$ and $\{\!|a|N|\!\}$.

3 Conclusion

We have developed a sound and complete algorithm for finding unifiers of multisets with multiple multiset variables. The algorithm is deterministic and terminating. It is implemented in SML, and we also provide the pseudo code for reproducibility. The same algorithm can be used for the particular case where there is only one multiset variable.

The complexity of the unification procedure is analysed, and its cost is high. This is inherent to the problem, since it is of combinatorial nature. It may be possible to improve this result by using the right data-structures and heuristics, but for our purposes, since the numbers are quite small, it runs fast enough.

We have also tried to eliminate all sources of redundancy, so that the set of computed unifiers is as close as possible to the minimal one. In particular, we have shown that a simple pre-processing of the input problem will produce fewer unifiers, and all those that are no longer produced can be recovered. We conjecture that this optimization leads to the computation of the *minimal* set of unifiers, but we leave this investigation as future work.

References

[1] I. Cervesato. Solution Count for Multiset Unification with Trailing Multiset Variables. In C. Ringeissen, C. Tinelli, F. Trinen, and R. Verma, editors, *Sixteenth International Workshop on Unification — UNIF'02*, pages 64–68, 2002.

[2] K. Chaudhuri, L. Lima, and G. Reis. Formalized Meta-Theory of Sequent Calculi for Substructural Logics. *Electronic Notes in Theoretical Computer Science*, 332:57 – 73, 2017. LSFA 2016 - 11th Workshop on Logical and Semantic Frameworks with Applications (LSFA).

[3] J. E. Dawson and R. Goré. Generic Methods for Formalising Sequent Calculi Applied to Provability Logic. In *Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, 2010. Proceedings*, pages 263–277, 2010.

[4] A. Dovier, A. Policriti, and G. Rossi. Integrating Lists, Multisets, and Sets in a Logic Programming Framework. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems: First International Workshop, Munich, March 1996*, pages 303–319. Springer Netherlands, 1996.

[5] A. Dovier, A. Policriti, and G. Rossi. A uniform axiomatic view of lists, multisets, and sets, and the relevant unification algorithms. *Fundam. Inf.*, 36(2-3):201–234, 1998.

[6] H. Tews. Formalizing Cut Elimination of Coalgebraic Logics in Coq. In *Automated Reasoning with Analytic Tableaux and Related Methods: 22nd International Conference, TABLEAUX 2013, Proceedings*, pages 257–272. Springer, 2013.

[7] B. Xavier, C. Olarte, G. Reis, and V. Nigam. Mechanizing Linear Logic in Coq. In *Proceedings of the 12th Workshop on Logical and Semantics Frameworks with Applications (LSFA)*, pages 60–77, 2017.

A Algorithms

Algorithm 1 $\mathsf{unify}(\{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}, \{\!\!\{s_1, ..., s_m | N_1, ..., N_l\}\!\!\})$

if $k = 0 \wedge l = 0$ **then**
    if $n = m$ **then return** $\mathsf{unify\_terms}([t_1, ..., t_n], [s_1, ..., s_m])$
    else return $[]$
else if $k = 0$ **then**
    if $m \leq n$ **then return** $\mathsf{unify\_c}(m, \{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}, \{\!\!\{s_1, ..., s_m | N_1, ..., N_l\}\!\!\})$
    else return $[]$
else if $l = 0$ **then**
    if $n \leq m$ **then return** $\mathsf{unify\_c}(n, \{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}, \{\!\!\{s_1, ..., s_m | N_1, ..., N_l\}\!\!\})$
    else return $[]$
else
    $c \leftarrow \mathsf{min}(n, m)$
    $\Sigma \leftarrow []$
    for $i = 0$ **to** c **do**
        $\Sigma' \leftarrow \mathsf{unify\_c}(i, \{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}, \{\!\!\{s_1, ..., s_m | N_1, ..., N_l\}\!\!\})$
        $\Sigma \leftarrow \Sigma \cup \Sigma'$
    end for
    return Σ
end if

Algorithm 2 $\mathsf{unify\_c}(c, \{\!\!\{t_1, ..., t_n | M_1, ..., M_k\}\!\!\}, \{\!\!\{s_1, ..., s_m | N_1, ..., N_l\}\!\!\})$

$\Sigma \leftarrow []$
$T_c \leftarrow \mathsf{choose}(\{\!\!\{t_1, ..., t_n\}\!\!\}, c)$
$S_c \leftarrow \mathsf{choose}(\{\!\!\{s_1, ..., s_m\}\!\!\}, c)$
for $(\{\!\!\{t'_1, ..., t'_c\}\!\!\}, \{\!\!\{t'_{c+1}, ..., t'_n\}\!\!\}) \in T_c$ **do**
    for $(\{\!\!\{s'_1, ..., s'_c\}\!\!\}, \{\!\!\{s'_{c+1}, ..., s'_m\}\!\!\}) \in S_c$ **do**
        $\Sigma_t \leftarrow \mathsf{unify\_terms}([t'_1, ..., t'_c], [s'_1, ..., s'_c])$
        if $\Sigma_t \neq []$ **then**
            $\Sigma_M \leftarrow \mathsf{unify\_distribute}((\{\!\!\{t'_{c+1}, ..., t'_n\}\!\!\}, \{M_1, ..., M_k\}), (\{\!\!\{s'_{c+1}, ..., s'_m\}\!\!\}, \{N_1, ..., N_l\}))$
            $\Sigma' \leftarrow \mathsf{map}(\lambda(\sigma_t, \sigma_M).\sigma_M \sigma_t)(\Sigma_t \times \Sigma_M)$
            $\Sigma \leftarrow \Sigma \cup \Sigma'$
        end if
    end for
end for
return Σ

B Proofs

Proof for Theorem 3. We know that Σ_{all} contains at most $n!$ unifiers, one for each way of pairing elements of $\bar{t}$ with elements of $\bar{s}$. Analogously, Σ contains at most $(n-1)!$ unifiers. Let: $\Sigma_{\text{all}} = \{\sigma_1, ..., \sigma_{n!}\}$ and $\Sigma = \{\mu_1, ..., \mu_{(n-1)!}\}$. Then each σ_i can be obtained from some μ_j.

If σ_i is a unifier resulting from pairing t_a with s_b, then there exists $\mu_j = \sigma_i$ and we are done.

Let σ_i be a unifier resulting from pairing t_a with some s_x, s_b with some t_y, and some permutation P_t of $\bar{t} \backslash \{\!\!\{t_a, t_y\}\!\!\}$ with some permutation P_s of $\bar{s} \backslash \{\!\!\{s_b, s_x\}\!\!\}$. There exists a unifier $\mu_j \in \Sigma$ that is the result of unifying the same permutations P_t and P_s, and t_y with s_x. We show how σ_i can be reconstructed from μ_j. Since the order in which terms are unified does not matter, we assume that σ_i and μ_j are obtained as follows:

Algorithm 3 unify_terms$([t_1, ..., t_n], [s_1, ..., s_n])$

$\Sigma \leftarrow []$
$P_s \leftarrow$ permutations$([s_1, ..., s_n])$
$P \leftarrow [[t_1, ..., t_n]] \times P_s$
for $(T, S) \in P$ **do**
  $\sigma \leftarrow$ unify_lists(T, S)
  if $\sigma \neq$ None **then** $\Sigma \leftarrow \{\sigma\} \cup \Sigma$
end for

Algorithm 4 unify_distribute$((\{\!\!\{t_1, ..., t_n\}\!\!\}, \{M_1, ..., M_k\}), (\{\!\!\{s_1, ..., s_m\}\!\!\}, \{N_1, ..., N_l\}))$

$\Sigma_N \leftarrow$ distribute$(\{\!\!\{t_1, ..., t_n\}\!\!\}, \{N_1, ..., N_l\}, k = 0)$ {List of substitutions for N_i}
$\Sigma_M \leftarrow$ distribute$(\{\!\!\{s_1, ..., s_m\}\!\!\}, \{M_1, ..., M_k\}, l = 0)$ {List of substitutions for M_i}
$\Sigma \leftarrow$ map$(\lambda(\sigma_M, \sigma_N).\sigma_M \cup \sigma_N)(\Sigma_M \times \Sigma_N)$
return Σ

Algorithm 5 distribute$(\{\!\!\{t_1, ..., t_n\}\!\!\}, \{M_1, ..., M_k\}, \text{exact})$

if $n = 0 \wedge k = 0$ **then**
    return $[\{\}]$
end if
$\Sigma \leftarrow []$
$P_t \leftarrow$ ordered_partitions$(\{\!\!\{t_1, ..., t_n\}\!\!\}, k)$
for $p \in P_t$ **do**
    $\sigma \leftarrow \{\}$
    for $i = 1$ **to** k **do**
        $ts \leftarrow p[i]$
        if exact **then** $\sigma \leftarrow \sigma\{M_i \mapsto \{\!\!\{ts | \cdot\}\!\!\}\}$
        if $\neg$exact $\wedge\, ts = \emptyset$ **then** $\sigma \leftarrow \sigma\{M_i \mapsto \{\!\!\{\cdot | M_i\}\!\!\}\}$
        if $\neg$exact $\wedge\, ts \neq \emptyset$ **then** $\sigma \leftarrow \sigma\{M_i \mapsto \{\!\!\{ts | M_i'\}\!\!\}\}$
    end for
    $\Sigma \leftarrow \{\sigma\} \cup \Sigma$
end for
return Σ

1. Computation of σ_i:
   (a) mgu σ_{ax} of t_a and s_x
   (b) mgu σ_{by} of $t_y\sigma_{ax}$ and $s_b\sigma_{ax}$
   (c) mgu σ_{P_i} of $P_t\sigma_{ax}\sigma_{by}$ and $P_t\sigma_{ax}\sigma_{by}$

2. Computation of μ_j:
   (a) mgu σ_{xy} of t_y and s_x
   (b) mgu σ_{P_j} of $P_t\sigma_{xy}$ and $P_s\sigma_{xy}$

Therefore $\sigma_i = \sigma_{ax}\sigma_{by}\sigma_{P_i}$ and $\mu_j = \sigma_{xy}\sigma_{P_j}$. We know that:

$$t_a\sigma_{ax} = s_x\sigma_{ax} \quad \text{from 1a} \tag{1}$$

$$t_y\sigma_{ax}\sigma_{by} = s_b\sigma_{ax}\sigma_{by} \quad \text{from 1b} \tag{2}$$

$$t_y\sigma_{ax}\sigma_{by} = t_a\sigma_{ax}\sigma_{by} \quad \text{because } t_a = s_b \tag{3}$$

$$t_y\sigma_{ax}\sigma_{by} = s_x\sigma_{ax}\sigma_{by} \quad \text{from 3 and 1} \tag{4}$$

Thus, $\sigma_{ax}\sigma_{by}$ is a unifier of t_y and s_x. But from 2a we have that σ_{xy} is the most general unifiers of these terms, which means that there exists a σ' such that

$$\sigma_{ax}\sigma_{by} = \sigma_{xy}\sigma' \tag{5}$$

From 5 and 1c, we know that $P_t\sigma_{xy}\sigma'\sigma_{P_i} = P_s\sigma_{xy}\sigma'\sigma_{P_i}$, meaning that $\sigma'\sigma_{P_i}$ is a unifier of $P_t\sigma_{xy}$ and $P_s\sigma_{xy}$. But from 2b, σ_{P_j} is the most general unifier of these two lists, therefore, there exists σ'' such that

$$\sigma'\sigma_{P_i} = \sigma_{P_j}\sigma'' \tag{6}$$

Using 5, 6, and associativity of substitution composition:

$$\sigma_i = \sigma_{ax}\sigma_{by}\sigma_{P_i} = \sigma_{xy}\sigma_{P_j}\sigma'' = \mu_j\sigma''$$

□

Proof for Theorem 4. We case on how t_a and s_b were used to compute σ. Let $\overrightarrow{t_c}$ and $\overrightarrow{s_c}$ denote the terms *and order* chosen from $\bar{t}$ and $\bar{s}$, respectively, to be unified. Let $\overline{t_r}$ and $\overline{s_r}$ denote the rest of the terms in $\bar{t}$ and $\bar{s}$ that will be distributed to $\overline{N}$ and $\overline{M}$, respectively.

We know that $\sigma = \sigma_M\sigma_t$ (Alg. 2, line 9), where σ_t is the unifier of $\overrightarrow{t_c}$ and $\overrightarrow{s_c}$, and σ_M is obtained from partitions denoted as $\pi_t(\overline{t_r})$ and $\pi_s(\overline{s_r})$.

1. $\boxed{t_a \in \overrightarrow{t_c} \text{ and } s_b \in \overrightarrow{s_c}}$

   Take $\mu = \mu_M\mu_t$ such that $\mu_M = \sigma_M$ is obtained from the same partitions $\pi_t(\overline{t_r})$ and $\pi_s(\overline{s_r})$, and μ_t is such that $\sigma_t = \mu_t\sigma'$ for some σ'. The existence of such μ_t is guaranteed by Theorem 3. Therefore, $\sigma = \sigma_M\sigma_t = \mu_M\mu_t\sigma' = \mu\sigma'$.

2. $\boxed{t_a \in \overline{t_r} \text{ and } s_b \in \overline{s_r}}$

   Let $\overline{t_{p_a}}, t_a$ be the part from $\pi_t(\overline{t_r})$ with t_a, and $\overline{s_{p_b}}, s_b$ the part from $\pi_s(\overline{s_r})$ with s_b.

   Take $\mu = \mu_M\mu_t$ such that $\mu_t = \sigma_t$ is the unifier of $\overrightarrow{t_c}$ and $\overrightarrow{s_c}$, and μ_M is obtained from partitions $\pi_t(\overline{t_r})$ where $\overline{t_{p_a}}, t_a$ is replaced by $\overline{t_{p_a}}$ and analogously for $\pi_s(\overline{s_r})$. Thus the mappings in σ_M and μ_M are the same, except for two multiset variables N_a and M_b:

   $$\begin{array}{llll} \{M_b \mapsto \overline{s_{p_b}}, s_b, M'_b & ; & N_a \mapsto \overline{t_{p_a}}, t_a, N'_a\} & \subset \sigma_M \\ \{M_b \mapsto \overline{s_{p_b}}, M'_b & ; & N_a \mapsto \overline{t_{p_a}}, N'_a\} & \subset \mu_M \end{array}$$

   Since M'_b and N'_a are fresh multiset variable names:

   $$\sigma_M = \mu_M\{M'_b \mapsto s_b, M'_b\ ;\ N'_a \mapsto t_a, N'_a\}$$

   And since $\sigma_t = \mu_t$:

   $$\sigma_M\sigma_t = \mu_M\{M'_b \mapsto s_b, M'_b\ ;\ N'_a \mapsto t_a, N'_a\}\mu_t$$

   The image of μ_t does not contain M'_b nor N'_a, therefore:

   $$\sigma_M\sigma_t = \mu_M\mu_t\{M'_b \mapsto s_b, M'_b\ ;\ N'_a \mapsto t_a, N'_a\}\mu_t$$

   Thus:

   $$\sigma = \mu\{M'_b \mapsto s_b, M'_b\ ;\ N'_a \mapsto t_a, N'_a\}\mu_t$$

3. $\boxed{t_a \in \overrightarrow{t_c} \text{ and } s_b \in \overline{s_r}}$

Let s_k be the term from $\overrightarrow{s_c}$ that is paired with t_a and $\overline{s_{p_b}}, s_b$ be the part from $\pi_s(\overline{s_r})$ containing s_b. Assume that t_a is unified with s_k with mgu σ_{ak}, and that $(\overrightarrow{t_c} \backslash \{\!|t_a|\!\})\sigma_{ak}$ unifies with $(\overrightarrow{s_c} \backslash \{\!|s_k|\!\})\sigma_{ak}$ with mgu σ_c. Thus $\sigma_t = \sigma_{ak}\sigma_c$. Let M_b be the variable to which partition $\overline{s_{p_b}}, s_b$ is assigned. Thus:

$$\{M_b \mapsto \overline{s_{p_b}}, s_b, M'_b\} \subset \sigma_M \qquad \text{By definition} \tag{1}$$

$$\{M_b \mapsto \overline{s_{p_b}}\sigma_{ak}, s_b\sigma_{ak}, M'_b\} \subset \sigma_M\sigma_{ak} \qquad \text{Composition with } \sigma_{ak} \tag{2}$$

$$\{M_b \mapsto \overline{s_{p_b}}\sigma_{ak}, t_a\sigma_{ak}, M'_b\} \subset \sigma_M\sigma_{ak} \qquad t_a = s_b \tag{3}$$

$$\{M_b \mapsto \overline{s_{p_b}}\sigma_{ak}, s_k\sigma_{ak}, M'_b\} \subset \sigma_M\sigma_{ak} \qquad t_a\sigma_{ak} = s_k\sigma_{ak} \tag{4}$$

Take $\mu = \mu_M\mu_t$ such that the terms $\overrightarrow{t_c} \backslash \{\!|t_a|\!\}$ and $\overrightarrow{s_c} \backslash \{\!|s_k|\!\}$ are unified with mgu μ_t, and μ_M is computed using partition $\pi_t(\overline{t_r})$ and $\pi_s(\overline{s_r})$ where part $\overline{s_{p_b}}, s_b$ is replaced by $\overline{s_{p_b}}, s_k$. Therefore,

$$\{M_b \mapsto \overline{s_{p_b}}, s_k, M'_b\} \subset \mu_M \tag{5}$$

Because μ_t is the mgu of the two lists, we have that: $\sigma_t = \sigma_{ak}\sigma_c = \mu_t\sigma'_t$ for some σ'_t. And from 4 and 5 we can also conclude: $\sigma_M\sigma_{ak} = \mu_M\sigma_{ak}$. Using these equalities: $\sigma = \sigma_M\sigma_t = \sigma_M\sigma_{ak}\sigma_c = \mu_M\sigma_{ak}\sigma_c = \mu_M\mu_t\sigma'_t = \mu\sigma'_t$.

4. $\boxed{t_a \in \overline{t_r} \text{ and } s_b \in \overrightarrow{s_c}}$ Analogous to the previous case.

□