On the Completeness of Selective Unification in Concolic Testing of Logic Programs

Frédéric Mesnard, Etienne Payet, Germán Vidal

On the Completeness of Selective Unification in Concolic Testing of Logic Programs*

Fred Mesnard[1], Étienne Payet[1], and Germán Vidal[2]

[1] LIM - Université de la Réunion, France
{frederic.mesnard,etienne.payet}@univ-reunion.fr
[2] MiST, DSIC, Universitat Politècnica de València, Spain
gvidal@dsic.upv.es

Abstract. Concolic testing is a popular dynamic validation technique that can be used for both model checking and automatic test case generation. We have recently introduced concolic testing in the context of logic programming. In contrast to previous approaches, the key ingredient in this setting is a technique to generate appropriate run-time goals by considering all possible ways an atom can unify with the heads of some program clauses. This is called "selective" unification. In this paper, we show that the existing algorithm is not complete and explore different alternatives in order to have a sound and complete algorithm for selective unification.

1 Introduction

A popular approach to software validation is based on so called *concolic execution* [4, 11], which combines both *conc*olic and symb*olic* execution [6, 3, 1]. Concolic *testing* [4] is a technique based on concolic execution for finding run time errors and automatically generating test cases. In this approach, both concrete and symbolic executions are performed in parallel, so that concrete executions may help to spot (run time) errors—thus avoiding false positives—and symbolic executions are used to generate alternative input data—new test cases—so that a good coverage is obtained.

In concolic testing of imperative programs, one should augment the states with a so called *path condition* that stores the constraints on the variables of the symbolic execution. Then, after a (possibly incomplete) concolic execution, these constraints are used for producing alternative input data (e.g., by negating one of the constraints). Furthermore, and this is one of the main advantages of concolic testing over the original approach based solely on symbolic execution, if the constraints in the path condition become too complex, one can still take some values from the concrete execution to simplify them. This is sound (but typically incomplete) and often allows one to explore a larger execution space than just

* This work has been partially supported by the EU (FEDER) and the Spanish *Ministerio de Economía y Competitividad* under grant TIN2013-44742-C4-1-R and by the *Generalitat Valenciana* under grant PROMETEO-II/2015/013 (SmartLogic).

giving up (as in the original approach based only on symbolic execution). Some successful tools that are based on concolic execution are, e.g., CUTE [11], SAGE [5], and Java Pathfinder [10].

We have recently introduced concolic testing in the context of logic programming [7]. There, a concolic state has the form $\langle S \,[\![\, S' \rangle$, where S and S' are sequences of concrete and symbolic goals,[3] respectively. In logic programming, the notion of *symbolic* execution is very natural. Indeed, the structure of both S and S' is the same—the sequences of atoms have the same predicates and in the same order—and the only difference is that some atoms might be less instantiated in S' than in S.

A key ingredient of concolic testing in logic programming is the search for new concrete goals so that alternative paths can be explored, thus improving the coverage achieved so far. Let us illustrate it with an example. Consider the following (labelled) logic program:

$$\begin{array}{lll} (\ell_1)\ p(s(a)). & (\ell_4)\ q(a). & (\ell_6)\ r(a). \\ (\ell_2)\ p(s(W)) \leftarrow q(W). & (\ell_5)\ q(b). & (\ell_7)\ r(c). \\ (\ell_3)\ p(f(X)) \leftarrow r(X). & & \end{array}$$

Given the initial goal $p(f(a))$, a concolic execution would combine a concrete execution of the form

$$p(f(a)) \rightarrow_{id} r(a) \rightarrow_{id} true$$

where id denotes the empty substitution, with another one for the more general goal $p(N)$:

$$p(N) \rightarrow_{\{N/f(Y)\}} r(Y) \rightarrow_{\{Y/a\}} true$$

that only mimicks the steps of the former derivation despite being more general. The technique in [7] would basically produce the following concolic execution:

$$\begin{array}{l} \langle p(f(a))_{id} \,[\![\, p(N)_{id} \rangle \rightsquigarrow_{c(\{\ell_3\},\{\ell_1,\ell_2,\ell_3\})} \langle r(a)_{id} \,[\![\, r(Y)_{\{N/f(Y)\}} \rangle \\ \qquad\qquad \rightsquigarrow_{c(\{\ell_6\},\{\ell_6,\ell_7\})} \langle \mathsf{true}_{id} \,[\![\, \mathsf{true}_{\{N/f(a)\}} \rangle \end{array}$$

where the goals are annotated with the answer computed so far. Roughly speaking, the above concolic execution is comprising the two standard SLD derivations for $p(f(a))$ and $p(N)$ above. Moreover, it also includes some further information: the labels of the clauses that unified with each concrete and symbolic goals.

For instance, the first step in the concolic execution above is labelled with $c(\{\ell_3\}, \{\ell_1, \ell_2, \ell_3\})$. This means that the concrete goal only unified with clause ℓ_3, but the symbolic goal unified with clauses ℓ_1, ℓ_2 and ℓ_3. Therefore, when looking for new run time goals that explore alternative paths, one should look for goals that unify with ℓ_1 but not with ℓ_2 and ℓ_3, that unify with ℓ_1 and ℓ_2 but not with ℓ_3, and so forth. In general, we should look for atoms that unify with

[3] Following the linear semantics of [12], we consider sequences of goals to represent the leaves of the SLD tree built so far.

all (and only) the feasible—i.e., those for which a solution exists—sets of clauses in $\{\{\}, \{\ell_1\}, \{\ell_1, \ell_2\}, \{\ell_1, \ell_2, \ell_3\}, \{\ell_2\}, \{\ell_2, \ell_3\}\}$. Also, some additional constraints on the groundness of some arguments are often required (e.g., to ensure that the generated goals are valid *run time* goals and, thus, will be terminating). A prototype implementation of the concolic testing scheme for pure Prolog, called `contest`, is publicly available from `http://kaz.dsic.upv.es/contest.html`.

In this paper, we focus on the so called *selective unification* problem that must be solved in order to produce the alternative goals during concolic testing. To be more precise, a selective unification problem is determined by a tuple $\langle A, \mathcal{H}^+, \mathcal{H}^-, G\rangle$ where

- A is the selected atom in a symbolic goal, e.g., $p(N)$,
- $\mathcal{H}^+$ are the atoms in the heads of the clauses we want A to unify with, e.g., for $\{\ell_1, \ell_2\}$ in the example above, we have $\mathcal{H}^+ = \{p(s(a)), p(s(W))\}$,
- $\mathcal{H}^-$ are the atoms in the heads of the clauses we do not want A to unify with, e.g., for $\{\ell_1, \ell_2\}$ in the example above, we have $\mathcal{H}^- = \{p(f(X))\}$,
- G is a set with the variables we want to be ground, e.g., $\{N\}$.

In this case, the problem is satisfiable and a solution is $\{N/s(a)\}$ since then $p(s(a))$ will unify with both atoms, $p(s(a))$ and $p(s(W))$, but it will not unify with $p(f(X))$ and, moreover, the variable N is ground.

In contrast, the case $\{\ell_1\}$ is not feasible, since there is no ground instance of $p(N)$ such that it unifies with $p(s(a))$ but not with $p(s(W))$.

In [7], we introduced a first algorithm for selective unification. Unfortunately, this algorithm was incomplete. In this paper, we further analyze this problem, identifying the potential sources of incompleteness, proving a number of properties, and introducing refined algorithms which are sound and complete under some circumstances.

This paper is organized as follows. After some preliminaries in Section 2, Section 3 recalls and then extends some of the developments in [7]. Then, Section 4 introduces refined versions of the algorithm for which we can obtain stronger results. Finally, Section 5 concludes and points out several possibilities for future work.

2 Preliminaries

We assume some familiarity with the standard definitions and notations for logic programs as introduced in [2]. Nevertheless, in order to make the paper as self-contained as possible, we present in this section the main concepts which are needed to understand our development.

We denote by $|S|$ the cardinality of the set S. In this work, we consider a first-order language with a fixed vocabulary of predicate symbols, function symbols, and variables denoted by Π, Σ and $\mathcal{V}$, respectively. We let $\mathcal{T}(\Sigma, \mathcal{V})$ denote the set of *terms* constructed using symbols from Σ and variables from $\mathcal{V}$. Positions are used to address the nodes of a term viewed as a tree. A *position* p in a term t, in symbols $p \in \mathcal{P}os(t)$, is represented by a finite sequence of natural

numbers, where ϵ denotes the root position. We let $t|_p$ denote the *subterm* of t at position p and $t[s]_p$ the result of *replacing the subterm* $t|_p$ by the term s. The depth $\mathit{depth}(t)$ of a term t is defined as: $\mathit{depth}(t) = 0$ if t is a variable and $\mathit{depth}(f(t_1, \ldots, t_n)) = 1 + \mathsf{max}(\mathit{depth}(t_1), \ldots, \mathit{depth}(t_n))$, otherwise. We say that $t|_p$ is a subterm of t at depth k if there are k nested function symbols from the root of t to the root of $t|_p$. An *atom* has the form $p(t_1, \ldots, t_n)$ with $p/n \in \Pi$ and $t_i \in \mathcal{T}(\Sigma, \mathcal{V})$ for $i = 1, \ldots, n$. The notion of position is extended to atoms in the natural way. A *goal* is a finite sequence of atoms $A_1, \ldots, A_n$, where the *empty goal* is denoted by *true*. A *clause* has the form $H \leftarrow \mathcal{B}$ where H is an atom and $\mathcal{B}$ is a goal (note that we only consider *definite* programs). A logic *program* is a finite sequence of clauses. $\mathcal{V}ar(s)$ denotes the set of variables in the syntactic object s (i.e., s can be a term, an atom, a query, or a clause). A syntactic object s is *ground* if $\mathcal{V}ar(s) = \emptyset$. In this work, we only consider *finite* ground terms.

Substitutions and their operations are defined as usual. In particular, the set $\mathcal{D}om(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ is called the *domain* of a substitution σ. We let id denote the empty substitution. The application of a substitution θ to a syntactic object s is usually denoted by juxtaposition, i.e., we write $s\theta$ rather than $\theta(s)$. The *restriction* $\theta\restriction_V$ of a substitution θ to a set of variables V is defined as follows: $x\theta\restriction_V = x\theta$ if $x \in V$ and $x\theta\restriction_V = x$ otherwise. We say that $\theta = \sigma\ [V]$ if $\theta\restriction_V = \sigma\restriction_V$. A syntactic object s_1 is *more general* than a syntactic object s_2, denoted $s_1 \leqslant s_2$, if there exists a substitution θ such that $s_2 = s_1\theta$. A *variable renaming* is a substitution that is a bijection on $\mathcal{V}$. Two syntactic objects t_1 and t_2 are *variants* (or equal up to variable renaming), denoted $t_1 \sim t_2$, if $t_1 = t_2\rho$ for some variable renaming ρ. A substitution θ is a unifier of two syntactic objects t_1 and t_2 iff $t_1\theta = t_2\theta$; furthermore, θ is the *most general unifier* of t_1 and t_2, denoted by $\mathsf{mgu}(t_1, t_2)$ if, for every other unifier σ of t_1 and t_2, we have that $\theta \leqslant \sigma$. We write $t_1 \approx t_2$ to denote that t_1 and t_2 unify for some substitution, which is not relevant here. By abuse of notation, we also use mgu to denote the most general unifier of a conjunction of equations of the form $s_1 = t_1 \wedge \ldots \wedge s_n = t_n$, i.e., $\mathsf{mgu}(s_1 = t_1 \wedge \ldots \wedge s_n = t_n) = \theta$ if $s_i\theta = t_i\theta$ for all $i = 1, \ldots, n$ and for every other unifier σ of s_i and t_i, $i = 1, \ldots, n$, we have that $\theta \leqslant \sigma$.

We say that a syntactic object o is *linear* if it does not contain multiple occurrences of the same variable. A substitution $\{X_1/t_1, \ldots, X_n/t_n\}$ is *linear* if $t_1, \ldots, t_n$ are linear and, moreover, they do not share variables.

3 The Selective Unification Problem

In this section, we first recall the unification problem from [7]. There, an algorithm for "selective unification" was proposed, and it was conjectured to be complete. Here, we prove that it is indeed incomplete and we identify two sources of incompleteness.

Definition 1 (selective unification problem). *Let A be an atom with $G \subseteq \mathcal{V}ar(A)$ a set of variables, and let $\mathcal{H}^+$ and $\mathcal{H}^-$ be finite sets of atoms such that all*

atoms are pairwise variable disjoint and $A \approx B$ *for all* $B \in \mathcal{H}^+ \cup \mathcal{H}^-$*. Then, the* selective unification problem *for* A *w.r.t.* $\mathcal{H}^+$*,* $\mathcal{H}^-$ *and* G *is defined as follows:*

$$\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) = \left\{ \sigma\restriction_{\mathcal{V}ar(A)} \;\middle|\; \begin{array}{l} \forall H \in \mathcal{H}^+ : A\sigma \approx H \\ \wedge\ \forall H \in \mathcal{H}^- : \neg(A\sigma \approx H) \\ \wedge\ G\sigma \text{ is ground} \end{array} \right\}$$

When the considered signature is finite, the following algorithm is sound and complete for solving the selective unification problem: first, bind the variables of A with terms of depth 0. If the condition above does not hold, then we try with terms of depth 1, and check it again. We keep increasing the considered term depth until a solution is found. Moreover, there exists a finite number n such that, if a solution has not been found when considering terms of depth n, then the problem is not satisfiable.

Theorem 1. *Let* A *be a linear atom with* $G \subseteq \mathcal{V}ar(A)$*,* $\mathcal{H}^+$ *be a finite set of linear atoms and* $\mathcal{H}^-$ *be a finite set of atoms such that all atoms are pairwise variable disjoint and* $A \approx B$ *for all* $B \in \mathcal{H}^+ \cup \mathcal{H}^-$*. Then, checking that* $\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) \neq \emptyset$ *is decidable.*

Proof. Here, we assume the naive algorithm sketched above. Let us first consider that all atoms in $\{A\} \cup \mathcal{H}^+ \cup \mathcal{H}^-$ are linear. Let k be the maximum depth of the atoms in $\{A\} \cup \mathcal{H}^+ \cup \mathcal{H}^-$. Consider the set

$$\Theta' = \{\theta \mid \mathcal{D}om(\theta) \subseteq \mathcal{V}ar(A),\ \mathit{depth}(A\theta) \leqslant k+1\}$$

On Θ', we define the binary relation $\theta_1 \simeq \theta_2$ iff $A\theta_1 \sim A\theta_2$. The relation $\simeq$ is an equivalence relation. Let $\Theta = \Theta'/\simeq$. The set Θ is usually large but finite. Now, we proceed by contradiction and assume that the problem is satisfiable but there is no solution in Θ.

Let $\sigma \in \mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ be one of such solutions with $\sigma \notin \Theta$. Let $k' \leqslant k$ be the maximum depth of the atoms in $\mathcal{H}^+$. Let $s_1, \ldots, s_n$ be the non-variable terms at depth $k'+1$ or higher in $A\sigma$, which occur at positions $p_1, \ldots, p_n$. Trivially, all atoms in $\mathcal{H}^+$ should have a variable at depth k' or lesser in order to still unify with $A\sigma$. Therefore, replacing these terms by any term would not change the fact that it unifies with all atoms in $\mathcal{H}^+$. Formally, $(\ldots(A\sigma[t_1]_{p_1})\ldots)[t_n]_{p_n} \approx H$ for all $H \in \mathcal{H}^+$ and for all terms $t_1, \ldots, t_n$.

Now, let us consider the negative atoms $\mathcal{H}^-$. Let us focus in the worst case, where the maximum depth of the atoms in $\mathcal{H}^-$ is $k \geq k'$. Since $\neg(A\sigma \approx H)$ for all $H \in \mathcal{H}^-$ and $(\ldots(A\sigma[t_1]_{p_1})\ldots)[t_n]_{p_n} \approx H$ for all $H \in \mathcal{H}^+$ and for all terms $t_1, \ldots, t_n$, let us choose terms $t'_1, \ldots, t'_n$ such that $\neg((\ldots(A\sigma[t'_1]_{p_1})\ldots)[t'_n]_{p_n} \approx H)$ for all $H \in \mathcal{H}^-$ and $(\ldots(A\sigma[t'_1]_{p_1})\ldots)[t'_n]_{p_n}$ has depth $k+1$. Note that this is always possible since, in the worst case, for each term in the atoms of $\mathcal{H}^-$ at depth k, we might need a term at depth $k+1$ (when the term in the atom of $\mathcal{H}^-$ is the only constant of the signature, so we need to introduce a function symbol and another constant if the argument should be ground). Let $\sigma' \subseteq \mathcal{D}om(A)$ be a subtitution such that $A\sigma' = (\ldots(A\sigma[t'_1]_{p_1})\ldots)[t'_n]_{p_n}$. Then, $\sigma' \in \mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ with $\sigma' \in \Theta$ and, thus, we get a contradiction.

Extending the proof to non-linear atoms is not difficult but it is tedious since we have to consider a higher depth that may depend on the multiple occurrences of the same variables. □

We conjecture that the above naive algorithm would also be complete for infinite signatures (e.g., integers) since the number of symbols in the considered atoms is finite. Nonetheless, such algorithms may be so inefficient that they are impractical in the context of concolic testing.

We note that the set $\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ is usually infinite. Moreover, even when considering only the *most general* solutions in this set, there may still exist more than one:

Example 1. Consider $A = p(X, Y)$, $\mathcal{H}^+ = \{p(Z, Z), p(a, b)\}$, $\mathcal{H}^- = \{p(c, c)\}$ and $G = \emptyset$. Then, both substitutions $\{X/a, Y/U\}$ and $\{X/U, Y/b\}$ are most general solutions in $\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$. In principle, any of them is equally good in our context.

In [7], we have introduced a stepwise method that, intuitively speaking, proceeds as follows:

- First, we produce some "maximal" substitutions θ for A such that $A\theta$ still unifies with the atoms in $\mathcal{H}^+$. Here, we use a special set $\mathcal{U}$ of fresh variables with $\mathcal{V}ar(\{A\} \cup \mathcal{H}^+ \cup \mathcal{H}^-) \cap \mathcal{U} = \emptyset$. The elements of $\mathcal{U}$ are denoted by U, U', $U_1 \ldots$ Then, in θ, the variables from $\mathcal{U}$ (if any) denote positions where further binding *might* prevent $A\theta$ from unifying with some atom in $\mathcal{H}^+$.
- In a second stage, we look for another substitution η such that $\theta\eta$ is a solution of the selective unification problem, i.e., $\theta\eta \in \mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$. Here, we basically follow a generate and test algorithm (as in the naive algorithm above), but it is now more restricted thanks to the bindings in θ and the fact that binding variables from $\mathcal{U}$ is not allowed.

In the first stage, we use the variables from the special set $\mathcal{U}$ to replace *disagreement pairs* (see [2] p. 27). Roughly speaking, given terms s and t, a subterm s' of s and a subterm t' of t form a disagreement pair if the root symbols of s' and t' are different, but the symbols from s' up to the root of s and from t' up to the root of t are the same. For instance, $X, g(a)$ and $b, h(Y)$ are disagreement pairs of the terms $f(X, g(b))$ and $f(g(a), g(h(Y)))$. A disagreement pair t, t' is called *simple* if one of the terms is a variable that does not occur in the other term and no variable of $\mathcal{U}$ occurs in t, t'. We say that the substitution $\{X/s\}$ is determined by t, t' if $\{X, s\} = \{t, t'\}$.

Definition 2 (algorithm for positive unification).

Input: *an atom A and a set of atoms $\mathcal{H}^+$ such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+$.*
Output: *a substitution θ.*

1. *Let $\mathcal{B} := \{A\} \cup \mathcal{H}^+$.*

2. *While simple disagreement pairs occur in* $\mathcal{B}$ *do*
   (a) *nondeterministically choose a simple disagreement pair* X, t *(respectively,* t, X*) in* $\mathcal{B}$*;*
   (b) *set* $\mathcal{B}$ *to* $\mathcal{B}\eta$ *where* $\eta = \{X/t\}$*.*
3. *While* $|\mathcal{B}| \neq 1$ *do*
   (a) *nondeterministically choose a disagreement pair* t, t' *in* $\mathcal{B}$*;*
   (b) *replace* t, t' *with a fresh variable from* $\mathcal{U}$*.*
4. *Return* $\theta\gamma$*, where* $\mathcal{B} = \{B\}$*,* $A\theta = B$*,* $\mathcal{D}om(\theta) \subseteq \mathcal{V}ar(A)$*, and* γ *is a variable renaming for the variables of* $\mathcal{V}ar(A\theta)\backslash\mathcal{U}$ *with fresh variables from* $\mathcal{V}\backslash\mathcal{U}$*.*

We denote by $\mathcal{SU}^+(A, \mathcal{H}^+)$ *the set of non-deterministic substitutions computed by the above algorithm.*

Observe that the step (2a) involves two types of non-determinism:

- *Don't care* nondeterminism, when there are several disagreement pairs X, t (or t, X) for *different* variables. In this case, we can select any of them and continue with the next step. The final solution would be the same no matter the selection. This is also true for step (3a), since the order in which the non-simple disagreement pairs are selected will not affect the final result.
- *Don't know* nondeterminism, when there are several disagreement pairs X, t (or t, X) for the same variable X. In this case, we should consider all possibilities since they may give rise to different solutions.

Example 2. Let $A = p(X, Y)$ and $\mathcal{H}^+ = \{p(a, b), p(Z, Z)\}$. Therefore, we start with $\mathcal{B} := \{p(X, Y), p(a, b), p(Z, Z)\}$. The algorithm then considers the simple disagreement pairs in $\mathcal{B}$. From X, a, we get $\eta_1 := \{X/a\}$ and the action (2b) sets $\mathcal{B}$ to $\mathcal{B}\eta_1 = \{p(a, Y), p(a, b), p(Z, Z)\}$. The substitution $\eta_2 := \{Y/b\}$ is determined by Y, b and the action (2b) sets $\mathcal{B}$ to $\mathcal{B}\eta_2 = \{p(a, b), p(Z, Z)\}$. Now, we have two don't know nondeterministic possibilities:

- If we consider the disagreement pair a, Z, we have a substitution $\eta_3 := \{Z/a\}$ and action (2b) then sets $\mathcal{B}$ to $\mathcal{B}\eta_3 = \{p(a, b), p(a, a)\}$. Now, no simple disagreement pair occurs in $\mathcal{B}$, hence the algorithm jumps to the loop at line 3. Action (3b) replaces the disagreement pair b, a with a fresh variable $U \in \mathcal{U}$, hence $\mathcal{B}$ is set to $\{p(a, U)\}$. As $|\mathcal{B}| = 1$ the loop at line 3 stops and the algorithm returns the substitution $\{X/a, Y/U\}$.
- If we consider the disagreement pair b, Z instead, we have a substitution $\eta_3' := \{Z/b\}$, and action (2b) sets $\mathcal{B}$ to $\mathcal{B}\eta_3' = \{p(a, b), p(b, b)\}$. Now, by proceeding as in the previous case, the algorithm returns $\{X/U', Y/b\}$.

Therefore, $\mathcal{SU}^+(A, \mathcal{H}^+) = \{\{X/a, Y/U\}, \{X/U', Y/b\}\}$.

The soundness of the algorithm in Definition 2 can then be proved as follows (termination is straightforward, see [8]). Note that this result was incomplete in [7] since the condition on $\mathcal{R}an(\eta)$ was missing.

Theorem 2. *Let A be an atom and $\mathcal{H}^+$ be a set of atoms such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+$. Then, for all $\theta \in \mathcal{SU}^+(A, \mathcal{H}^+)$, we have that $A\theta\eta \approx H$ for all $H \in \mathcal{H}^+$ and for any idempotent substitution η with $\mathcal{D}om(\eta) \subseteq \mathcal{V}ar(A\theta)\backslash\mathcal{U}$ and $\mathcal{R}an(\eta) \cap (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U}) = \emptyset$.*

In order to prove this theorem, we first need the following results, which can be found in [8, Appendix B.2]:

Lemma 1. *Suppose that $A\theta = B\theta$ for some atoms A and B and some substitution θ. Then we have $A\theta\eta = B\eta\theta\eta$ for any substitution η with $[\mathcal{D}om(\eta) \cap \mathcal{V}ar(B)] \cap \mathcal{D}om(\theta) = \emptyset$ and $Ran(\eta) \cap \mathcal{D}om(\theta\eta) = \emptyset$.*

Proposition 1. *The loop at line 3 always terminates and the following statement is an invariant of this loop.*

(inv′) *For each $A' \in \{A\} \cup \mathcal{H}^+$ there exists $B \in \mathcal{B}$ and a substitution θ such that $A'\theta = B\theta$, $\mathcal{D}om(\theta) \subseteq (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U})$ and $\mathcal{V}ar(\mathcal{B}) \cap \mathcal{D}om(\theta) \subseteq \mathcal{U}$.*

The proof of Theorem 2 can now proceed as follows:

Proof. Upon termination of the loop at line 3 we have $|\mathcal{B}| = 1$. Let B be the element of $\mathcal{B}$ with $A\theta = B$, and let $\theta' \in \mathcal{SU}^+(A, \mathcal{H}^+)$ be a renaming of θ for the variables of $A\theta\backslash\mathcal{U}$. By Proposition 1, we have that, for all $H \in \mathcal{H}^+$, there exists a substitution μ such that $A\theta\mu = H\mu$ and the following conditions hold:

- $\mathcal{D}om(\mu) \subseteq (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U})$ and
- $\mathcal{V}ar(A\theta) \cap \mathcal{D}om(\mu) \subseteq \mathcal{U}$.

Trivially, there exists a unifier μ' for $A\theta'$ and H too, and the same conditions hold: $\mathcal{D}om(\mu') \subseteq (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U})$ and $\mathcal{V}ar(A\theta') \cap \mathcal{D}om(\mu') \subseteq \mathcal{U}$.

Now, in order to apply Lemma 1, we need to prove the following conditions:

- $[\mathcal{D}om(\eta) \cap \mathcal{V}ar(A\theta')] \cap \mathcal{D}om(\mu') = \emptyset$. This is trivially implied by the fact that $\mathcal{D}om(\eta) \subseteq \mathcal{V}ar(A\theta')\backslash\mathcal{U}$ and $\mathcal{V}ar(A\theta') \cap \mathcal{D}om(\mu') \subseteq \mathcal{U}$.
- $\mathcal{R}an(\eta) \cap \mathcal{D}om(\mu'\eta) = \emptyset$. First, since $\mathcal{D}om(\mu'\eta) \subseteq \mathcal{D}om(\mu') \cup \mathcal{D}om(\eta)$, we prove the stronger claim: $\mathcal{R}an(\eta) \cap \mathcal{D}om(\mu') = \emptyset$ and $\mathcal{R}an(\eta) \cap \mathcal{D}om(\eta) = \emptyset$. The second condition is triviallly implied by the idempotency of η. Regarding the first condition, it is implied by $\mathcal{R}an(\eta) \cap (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U}) = \emptyset$ since $\mathcal{D}om(\mu') \subseteq (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U})$, which is true.

Therefore, by Lemma 1, we have that $A\theta'\eta\mu'\eta = H\mu'\eta$ and, thus, $A\theta'\eta$ unifies with H. Hence, we have proved that $A\theta'\eta$ unifies with every atom in $\mathcal{H}^+$. □

Now we deal with the negative atoms and the groundness constraints by means of the following algorithm:

Definition 3 (algorithm for selective unification).

Input: *an atom A with $G \subseteq \mathcal{V}ar(A)$ a set of variables, and two finite sets $\mathcal{H}^+$ and $\mathcal{H}^-$ such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+ \cup \mathcal{H}^-$.*

Output: fail *or a substitution* $\theta\eta$ *(restricted to the variables of* A*).*

1. *Generate—using a fair algorithm—pairs* (θ, η) *with* $\theta \in \mathcal{SU}^+(A, \mathcal{H}^+)$ *and* η *an idempotent substitution such that* $G\theta\eta$ *is ground,* $\mathcal{D}om(\eta) \subseteq \mathcal{V}ar(A\theta)\backslash\mathcal{U}$ *and* $\mathcal{R}an(\eta) \cap (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U}) = \emptyset$*, otherwise return* fail*.*
2. *Check that for each* $H^- \in \mathcal{H}^-$*,* $\neg(A\theta\eta \approx H^-)$*, otherwise return* fail*.*
3. *Return* $\theta\eta\gamma$ *(restricted to the variables of* A*), where* γ *is a variable renaming for* $A\theta\eta$ *with fresh variables from* $\mathcal{V}\backslash\mathcal{U}$*.*

We denote by $\mathcal{SU}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ *the set of non-deterministic (non-failing) substitutions computed by the above algorithm.*

Note that step (1) above is don't know nondeterministic and, thus, all substitutions in $\mathcal{SU}^+(A, \mathcal{H}^+)$ should in principle be considered. On the other hand, computing the first solution of the above algorithm is enough for concolic testing.

The soundness of the selective unification algorithm is a straightforward consequence of Theorem 2 and the fact that the algorithm in Definition 3 is basically a fair generate-and-test procedure.

Unfortunately, the selective unification algorithm is not complete in general, as Examples 3 and 4 below illustrate. Example 3 shows that the algorithm cannot always compute all the solutions while Example 4 shows that it may even find no solution at all for a satisfiable instance of the problem.

Example 3. Consider the atom $A = p(X_1, X_2)$ with $G = \{X_1\}$, and the sets $\mathcal{H}^+ = \{p(X, g(X)), p(Z, Z)\}$ and $\mathcal{H}^- = \{p(g(b), W)\}$. Here, we have

$$\mathcal{SU}^+(A, \mathcal{H}^+) = \{\underbrace{\{X_1/X', X_2/U\}}_{\theta_1}, \underbrace{\{X_1/U, X_2/g(X')\}}_{\theta_2}\}$$

The algorithm is able to compute the solution $\{X_1/g(a), X_2/U\}$ from θ_1, $\eta = \{X'/g(a)\}$ and $\gamma = id$. However, it cannot compute $\{X_1/g(a), X_2/g(X')\} \in \mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$.

The algorithm fails here because the instantiation of variables from $\mathcal{U}$ is not allowed. In [7], it was incorrectly assumed that *any* binding of a variable from $\mathcal{U}$ will result in a substitution θ' such that $A\theta'$ does not unify will all the atoms in $\mathcal{H}^+$ anymore. However, the universal quantification was not right. For each variable from $\mathcal{U}$, we can only ensure that *there exists some* term t such that binding this variable to t will result in a substitution that prevents A from unifying with some atom in $\mathcal{H}^+$. Therefore, since the algorithm of Definition 3 forbids the bindings of the variables in $\mathcal{U}$, completeness is lost. We will propose a solution to this problem in the next section

Example 4. Consider $A = p(X_1, X_2)$, $\mathcal{H}^+ = \{p(X, a), p(b, Y)\}$, $\mathcal{H}^- = \{p(b, a)\}$, and $G = \emptyset$. Here, we have $\mathcal{SU}^+(A, \mathcal{H}^+) = \{\{X_1/b, X_2/a\}\}$ and, thus, the algorithm in Definition 3 fails. However, the following substitution $\{X_1/Z, X_2/Z\}$ is a solution, i.e., $\{X_1/Z, X_2/Z\} \in \mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$.

Unfortunately, we do not know how to generate such non-linear solutions except with the naive semi-algorithm mentioned at the beginning of this section, which is not generally useful in practice. Therefore, in the next section we will rule out these solutions.

4 Recovering Completeness for Linear Selective Unification

In this section, we introduce different alternatives to recover the completeness of the selective unification algorithm.

In the following, we only consider a subset of the solutions to the selective unification problem, namely those which are *linear*:

$$\mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G) = \{\sigma \in \mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) \mid \sigma \text{ is linear}\}$$

i.e., we rule out solutions like those in Example 4 since we do not know how such solutions can be produced using a constructive algorithm. We refer to $\mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ as the *linear selective unification problem*.

4.1 A Naive Extension

One of the sources of incompleteness of the algorithm in Definition 3 comes from the fact that the variables from $\mathcal{U}$ cannot be bound. Therefore, one can consider a naive extension of this algorithm as follows:

Definition 4 (extended algorithm for selective unification).

Input: *an atom A with $G \subseteq \mathcal{V}ar(A)$ a set of variables, and two finite sets $\mathcal{H}^+$ and $\mathcal{H}^-$ such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+ \cup \mathcal{H}^-$.*

Output: fail *or a substitution $\theta\eta$ (restricted to the variables of A).*

1. *Generate—using a fair algorithm—pairs (θ, η) with $\theta \in \mathcal{SU}^+(A, \mathcal{H}^+)$ and η an idempotent substitution such that $G\theta\eta$ is ground, $\mathcal{D}om(\eta) \subseteq \mathcal{V}ar(A\theta)$ and $\mathcal{R}an(\eta) \cap (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U}) = \emptyset$, otherwise return* fail.
2. *Check that for each $H^- \in \mathcal{H}^-$, $\neg(A\theta\eta \approx H^-)$, otherwise return* fail.
3. *Return $\theta\eta\gamma$ (restricted to the variables of A), where γ is a variable renaming for $A\theta\eta$ with fresh variables from $\mathcal{V}\backslash\mathcal{U}$.*

We denote by $\mathcal{SU}^*(A, \mathcal{H}^+, \mathcal{H}^-, G)$ the set of non-deterministic (non-failing) substitutions computed by the above algorithm.

In general, though, the above algorithm can be very inefficient since all variables in $A\theta$ can now be bound, even those in $\mathcal{U}$. Nevertheless, one can easily define a fair procedure for generating pairs (θ, η) in step (1) above which gives priority to binding the variables in $\mathcal{V}ar(A\theta)\backslash\mathcal{U}$, so that the variables from $\mathcal{U}$ are only bound when no solution can be found otherwise.

4.2 The Positive Unification Problem

Now, we introduce a more efficient instance of the algorithm for linear selective unification which is sound and complete when the atoms in A and $\mathcal{H}^+$ are linear. Formally, we are concerned with the following unification problem:

Definition 5 (positive linear unification problem). *Let A be a linear atom and let $\mathcal{H}^+$ be a finite set of linear atoms such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+$. Then, the* positive linear unification problem *for A w.r.t. $\mathcal{H}^+$ is defined as follows:*

$$\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+) = \{\sigma\restriction_{\mathcal{V}ar(A)} |\ (\forall H \in \mathcal{H}^+ : A\sigma \approx H) \text{ and } \sigma \text{ is linear}\}$$

Note that we do not want to find a unifier between A and *all* the atoms in $\mathcal{H}^+$, but a substitution θ such that $A\theta$ still unifies with *each* atom in $\mathcal{H}^+$ independently. So this problem is different from the usual unification problems found in the literature.

Clearly, $|\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)| \geq 1$ since the identity substitution is always a solution to the positive linear unification problem. In general, though, the set $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$ is infinite.

Example 5. Let us consider $A = p(X)$ and $\mathcal{H}^+ = \{p(f(Y)), p(f(g(Z)))\}$. Then, we have $\{id, \{X/f(X')\}, \{X/f(g(X'))\}, \{X/f(g(a))\}, \{X/f(g(f(X')))\}, \ldots\} \subseteq \mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$, which is clearly infinite.

Therefore, in the following, we restrict our interest to so called *maximal* solutions:

Definition 6 (maximal solution). *Let A be a linear atom and $\mathcal{H}^+$ be a finite set of linear atoms such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+$. We say that a substitution $\theta \in \mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$ is* maximal *when the following conditions hold:*

1. *for any idempotent substitution γ with $\mathcal{D}om(\gamma) \subseteq \mathcal{V}ar(A\theta) \setminus \mathcal{U}$ and $\mathcal{R}an(\gamma) \cap (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U}) = \emptyset$, $(\theta\gamma)\restriction_{\mathcal{V}ar(A)}$ is still an element of $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$,*
2. *for any variable $U \in \mathcal{V}ar(A\theta) \cap \mathcal{U}$, we have that $(\theta\{U/t\})\restriction_{\mathcal{V}ar(A)}$ is not an element of $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$ anymore for all non-variable term t, and*
3. *for any $X/t \in \theta$ and for all non-variable term $t|_p$, replacing it by a non-variable term rooted by a different symbol will result in a substitution which is not an element of $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$ anymore.*

We let $max(A, \mathcal{H}^+)$ denote the set of maximal solutions in $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$.

Intuitively speaking, given a maximal solution θ, the first condition implies that $(\theta\gamma)\restriction_{\mathcal{V}ar(A)}$ is still a solution of the positive linear unification problem as long as no variables from $\mathcal{U}$ are bound. The second and third conditions mean that the rest of the symbols in θ cannot be changed, i.e., binding a variable from $\mathcal{U}$ with a non-variable term or changing any constant or function symbol by a different one, will always result in a substitution which is not a solution of the positive linear unification problem anymore.

Example 6. Consider, e.g., $A = p(X_1, X_2)$ and $\mathcal{H}^+ = \{p(f(Y), a), p(f(g(Z)), b)\}$. Here, we have $\{X_1/X', X_2/X''\} \in \mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$ but it is not a maximal solution, i.e., $\{X_1/X', X_2/X''\} \notin \mathit{max}(A, \mathcal{H}^+)$ since binding X'' to, e.g., a, will result in a substitution which is not in $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$. In contrast, $\{X_1/f(g(Z')), X_2/U\}$ is a maximal solution. However, any substitution of the form $\{X_1/f(g(t)), X_2/U\}$ for any non-variable term t is not a maximal solution since the third condition will not hold anymore (one can change the symbols introduced by t and still get a solution in $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$). The substitution $\{X_1/f(Y'), X_2/U\}$ is not a maximal solution as well since binding Y' to, e.g., a, will result in a substitution which is not in $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$, hence the first condition does not hold. And the same applies to $\{X_1/f(U'), X_2/U\}$, which is not a maximal solution either since we can bind U' to $g(X')$ and still get a substitution in $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$.

In contrast to $\mathcal{P}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$, the set $\mathit{max}(A, \mathcal{H}^+)$ is finite, since it is bounded by the depth of the terms in $\mathcal{H}^+$. Actually, for linear atoms in $\{A\} \cup \mathcal{H}^+$, there is only *one* maximal solution.

Proposition 2. *Let A be a linear atom and $\mathcal{H}^+$ be a finite set of linear atoms such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+$. Then, the set $\mathit{max}(A, \mathcal{H}^+)$ is a singleton (up to variable renaming).*

Proof. We proceed by contradiction. Let us assume that there are two maximal solutions $\sigma, \theta \in \mathit{max}(A, \mathcal{H}^+)$, where $X/s \in \sigma$ and $X/t \in \theta$ for some variable $X \in \mathcal{V}ar(A)$. Let us consider that s and t differ at position p such that $s|_p$ and $t|_p$ are rooted by a different symbol. Now, we distinguish the following cases:

- If $s|_p$ and $t|_p$ are rooted by different constant or function symbols, we get a contradiction by condition (3) of maximal solution.
- If $s|_p$ is rooted by a constant or function symbol, while $t|_p$ is rooted by a variable from $\mathcal{U}$ (or viceversa), we get a contradiction by condition (2) of maximal solution.
- If $s|_p$ is rooted by a constant or function symbol, while $t|_p$ is rooted by a variable from $\mathcal{V}\backslash\mathcal{U}$ (or viceversa), we get a contradiction either by condition (1) or (3) of maximal solution.
- Finally, if $s|_p$ is rooted by a variable from $\mathcal{U}$, while $t|_p$ is rooted by a variable from $\mathcal{V}\backslash\mathcal{U}$ (or viceversa), we get a contradiction either by condition (1) or (2) of maximal solution.

Therefore, the set $\mathit{max}(A, \mathcal{H}^+)$ is necessarily a singleton. □

Moreover, the following key property holds: a maximal solution can always be *completed* in order to get a solution to the linear unification problem when it is satisfiable. In order to prove this result, we need to recall the definition of *parallel composition* of substitutions, denoted by $\Uparrow$ in [9].

Definition 7 (parallel composition [9]). *Let θ_1 and θ_2 be two idempotent substitutions. Then, we define $\Uparrow$ as follows:*

$$\theta_1 \Uparrow \theta_2 = \begin{cases} \mathsf{mgu}(\widehat{\theta_1} \wedge \widehat{\theta_2}) & \text{if } \widehat{\theta_1} \wedge \widehat{\theta_2} \text{ has a solution (a unifier)} \\ \mathit{fail} & \text{otherwise} \end{cases}$$

where $\widehat{\theta}$ *denotes the* equational representation *of a substitution* θ*, i.e., if* $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$ *then* $\widehat{\theta} = (X_1 = t_1 \wedge \cdots \wedge X_n = t_n)$.

Proposition 3. *Let* A *be a linear atom and* $\mathcal{H}^+$ *be a finite set of linear atoms such that all atoms are pairwise variable disjoint and* $A \approx B$ *for all* $B \in \mathcal{H}^+$*. Let* $\theta \in max(A, \mathcal{H}^+)$ *be the maximal solution for* A *and* $\mathcal{H}^+$*. Then, if* $\mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ *is satisfiable (the set contains at least one substitution), then there exists some substitution* γ *such that* $\theta\gamma \in \mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$.

Proof. For simplicity, we consider that $A = p(X)$, $\mathcal{H}^+ = \{p(t_1), \ldots, p(t_n)\}$ and $\mathcal{H}^- = \{p(s_1), \ldots, p(s_m)\}$. Since the atoms are linear, the claim would follow by a similar argument. Let $\theta = \{X/t\} \in max(A, \mathcal{H}^+)$ be the maximal solution. Hence, we have $t \approx t_i$ for all $i = 1, \ldots, n$. Let $\sigma \in \mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ be a solution to the selective unification problem. By definition of maximal solution, there may be other solutions to the positive unification problem, but every introduced symbol cannot be different if we want to still unify with all terms $t_1, \ldots, t_n$ by condition (3) in the definition of maximal solution. Therefore, both substitutions must be compatible, i.e., we have $\theta \Uparrow \sigma = \delta \neq \mathsf{fail}$. Furthermore, taking into account the negative atoms in $\mathcal{H}^-$ as well as the groundness constraints w.r.t. G, δ can only introduce further bindings, but would never require to generalize any term introduced by θ and, thus, δ can be decomposed as $\theta\gamma$, with $\theta\gamma \in \mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$. □

Therefore, computing the maximal solution suffices to check for satisfiability. Here, we use again the algorithm in Definition 2 for computing the maximal solution, with the following differences:

- First, both A and the atoms in $\mathcal{H}^+$ are now linear.
- Moreover, step (2a) is now don't care nondeterministic, so the algorithm will return a single solution, which we denote by $\mathcal{SU}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$.

Proposition 4. *Let* A *be a linear atom and* $\mathcal{H}^+$ *be a finite set of linear atoms such that all atoms are pairwise variable disjoint and* $A \approx B$ *for all* $B \in \mathcal{H}^+$*. Then,* $\mathcal{SU}^+_{\mathsf{lin}}(A, \mathcal{H}^+) = max(A, \mathcal{H}^+)$.

Proof. (sketch) The fact that $\mathcal{SU}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$ returns a singleton is trivial by definition, since the algorithm has no don't know nondeterminism and no step admits a failure.

Regarding the fact that θ is a maximal solution, let us prove that all three conditions in Definition 6 hold. The first condition of maximal solution follows by Theorem 2, which is proved for the more general case of arbitrary (possibly non-linear) atoms. The third condition holds from the fact that in step (2) of $\mathcal{SU}^+_{\mathsf{lin}}$ only symbols from the atoms A and $\mathcal{H}^+$ are introduced following a *mgu*-like algorithm; therefore they are possibly not necessary, but cannot be replaced by different symbols and still unify with all the atoms in $\mathcal{H}^+$. Finally, the second condition derives from step (3) of $\mathcal{SU}^+_{\mathsf{lin}}$ where non-simple disagreement pairs are replaced by fresh variables from $\mathcal{U}$ and, thus, any binding to a non-variable term would result in $A\theta$ not unifying with some atom of $\mathcal{H}^+$. □

4.3 Dealing with the Negative Atoms

The algorithm $\mathcal{SU}$ in Definition 3 is now redefined as follows:

Definition 8 (algorithm for linear selective unification).

Input: *a linear atom A with $G \subseteq \mathcal{V}ar(A)$ a set of variables, and two finite sets $\mathcal{H}^+$ and $\mathcal{H}^-$ such that the atoms in $\mathcal{H}^+$ are linear and all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+ \cup \mathcal{H}^-$.*
Output: fail *or a substitution $\theta\eta$ (restricted to the variables of A).*

1. *Let $\{\theta\} = \mathcal{SU}^+_{\mathsf{lin}}(A, \mathcal{H}^+)$. Then, generate—using a fair algorithm—linear idempotent substitutions η such that $G\theta\eta$ is ground, $\mathcal{D}om(\eta) \subseteq \mathcal{V}ar(A\theta)\backslash\mathcal{U}$ and $\mathcal{R}an(\eta) \cap (\mathcal{V}ar(\mathcal{H}^+ \cup \{A\}) \cup \mathcal{U}) = \emptyset$, otherwise return* fail.
2. *Check that for each $H^- \in \mathcal{H}^-$, $\neg(A\theta\eta \approx H^-)$, otherwise return* fail.
3. *Return $\theta\eta\gamma$ (restricted to the variables of A), where γ is a variable renaming for $A\theta\eta$ with fresh variables from $\mathcal{V}\backslash\mathcal{U}$.*

We denote by $\mathcal{SU}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ the set of non-deterministic (non-failing) substitutions computed by the above algorithm.

Example 7. Consider again $A = p(X_1, X_2)$ and $\mathcal{H}^+ = \{p(f(Y), a), p(f(g(Z)), b)\}$, together with $\mathcal{H}^- = \{p(f(g(a)), c)\}$ and $G = \{X_1\}$. The algorithm for linear positive unification returns the maximal substitution $\{X_1/f(g(Z')), X_2/U\}$. Therefore, the algorithm for linear selective unification would eventually produce a solution of the form $\theta = \{X_1/f(g(b)), X_2/X'\}$ since $A\theta = p(f(g(b), X')$ unifies with $p(f(Y), a)$ and $p(f(g(Z)), b)$ but not with $p(f(g(a)), c)$ and, moreover, X_1 is not ground. However, if we consider a non-maximal solution, the algorithm in Definition 3 may fail, even if there exists some solution to the linear selective unification problem. This is the case, e.g., if we consider the non-maximal solution $\{X_1/f(g(a)), X_2/U\}$.

Theorem 3 (soundness). *Let A be a linear atom with $G \subseteq \mathcal{V}ar(A)$, $\mathcal{H}^+$ be a finite set of linear atoms and $\mathcal{H}^-$ be a finite set of atoms such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+ \cup \mathcal{H}^-$. Then, we have $\mathcal{SU}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G) \subseteq \mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$.*

Proof. The claim follows from Proposition 4 by assuming that the don't know nondeterministic substitutions considered in step (1) of the algorithm of Definition 8 are obtained by a fair generate-and-test algorithm which produces substitutions systematically starting with terms of depth 0, then depth 1, etc., as in the naive algorithm described at the beginning of Section 3.

The following result states the completeness of our algorithm. In principle, we do not guarantee that all solutions are computed using our algorithms, even for the linear case. However, we can ensure that if the linear selective unification problem is satisfiable, our algorithm will find at least one solution.

Theorem 4 (completeness). *Let A be a linear atom with $G \subseteq \mathcal{V}ar(A)$, $\mathcal{H}^+$ be a finite set of linear atoms and $\mathcal{H}^-$ be a finite set of atoms such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+ \cup \mathcal{H}^-$. Then, if $\mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G) \neq \emptyset$ (i.e., it is satisfiable), then $\mathcal{SU}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G) \neq \emptyset$.*

Proof. By Proposition 3, if $\mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G) \neq \emptyset$ and θ is the computed maximal solution, then there exists a substitution γ such that $(\theta\gamma)\restriction_{\mathcal{V}ar(A)} \in \mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$. Moreover, such a substitution γ can be obtained by a fair generate-and-test algorithm such as that considered in Definition 8. Finally, the claim follows by Proposition 4 which ensures that the algorithm in Definition 2 will always produce the maximal solution for A and $\mathcal{H}^+$.

In general, though, we cannot ensure that all solutions are computed (which is not a drawback of the algorithm since we are only interested in finding one solution if it exists):

Example 8. Consider again $A = p(X_1, X_2)$ and $\mathcal{H}^+ = \{p(f(Y), a), p(f(g(Z)), b)\}$, together with $\mathcal{H}^- = \{p(g(W), c)\}$ and $G = \emptyset$. The algorithm for linear positive unification returns the maximal substitution $\{X_1/f(g(Z')), X_2/U\}$. Therefore, it is impossible that the algorithm in Definition 3 could produce a solution like $\{X_1/f(X'), X_2/X''\} \in \mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G)$.

5 Discussion

In this paper, we have studied the soundness and completeness of selective unification, a relevant operation in the context of concolic testing of logic programs. In contrast to [7], we have provided a refined correctness result (one condition was missing in [7]), and we have also identified the main sources of incompleteness for the algorithm in [7]. Then, we have introduced several refinements so that the procedure is now sound and complete w.r.t. linear solutions. We are not aware of any other work that deals with the kind of unification problems that we consider in this paper.

Clearly, the fact that we only consider linear solutions (i.e., the relation $\mathcal{P}_{\mathsf{lin}}$) means that our procedure can be incomplete in general. For instance, we consider the problem shown in Example 4 unsatisfiable, though a nonlinear solution exists. Nevertheless, we do not expect this restriction to have a significant impact in practice and, moreover, concolic testing algorithms are usually incomplete in order to avoid a state explosion. On the other hand, the refined algorithm in Sections 4.2 and 4.3 only considers linear atoms. This restriction may have a more significant impact since many programs have nonlinear atoms in the heads of the clauses and/or equalities in the bodies. In such cases, we can still resort to using the algorithm of Section 4.1, though it may be less efficient.

As for future work, we are considering to introduce a technique to "linearize" the atoms in $A \cup \mathcal{H}^+$ by introducing some constraints which could be solved later in the algorithm (e.g., replacing $p(X, X)$ by $p(X, Y)$ and the constraint $X = Y$).

Another interesting line of research involves improving the efficiency of the selective unification algorithm. For this purpose, we plan to investigate the conditions ensuring the following property:

$$\text{if } \mathcal{P}_{\mathsf{lin}}(A, \mathcal{H}^+, \mathcal{H}^-, G) = \emptyset, \text{ then } \mathcal{P}_{\mathsf{lin}}(A\theta, \mathcal{H}^+, \mathcal{H}^-, G) = \emptyset \text{ for all substitution } \theta$$

If this property indeed holds, then one could check *statically* the satisfiability of all possible selective unification problems in a program, e.g., for atoms of the form $p(X_1, \ldots, X_n)$. We can then use this information during concolic testing to rule out those problems which we know are unfeasible no matter the run time values (denoted by θ). From our preliminary experience with the tool `contest` (`http://kaz.dsic.upv.es/contest.html`), this might result in significant efficiency improvements.

Finally, we are also considering the definition of a possibly approximate formulation of selective unification which could be solved using an SMT solver. This might imply a loss of completeness, but will surely improve the efficiency of the process. Moreover, it will also allow a smoother integration with the constraint solving process which is required when extending our concolic testing technique to full Prolog programs.

References

1. Saswat Anand, Corina S. Pasareanu, and Willem Visser. Symbolic execution with abstraction. *STTT*, 11(1):53–67, 2009.
2. K.R. Apt. *From Logic Programming to Prolog*. Prentice Hall, 1997.
3. L.A. Clarke. A program testing system. In *Proceedings of the 1976 Annual Conference (ACM'76)*, pages 488–491, 1976.
4. P. Godefroid, N. Klarlund, and K. Sen. DART: directed automated random testing. In *Proc. of PLDI'05*, pages 213–223. ACM, 2005.
5. P. Godefroid, M.Y. Levin, and D.A. Molnar. Sage: whitebox fuzzing for security testing. *CACM*, 55(3):40–44, 2012.
6. James C. King. Symbolic execution and program testing. *CACM*, 19(7):385–394, 1976.
7. F. Mesnard, É. Payet, and G. Vidal. Concolic testing in logic programming. *TPLP*, 15(4-5):711–725, 2015.
8. F. Mesnard, É. Payet, and G. Vidal. Concolic testing in logic programming (extended version). *CoRR*, abs/1507.05454, 2015. Available from the following URL: `http://arxiv.org/abs/1507.05454`.
9. C. Palamidessi. Algebraic Properties of Idempotent Substitutions. In M.S. Paterson, editor, *Proc. of 17th Int'l Colloquium on Automata, Languages and Programming*, pages 386–399. Springer LNCS 443, 1990.
10. C.S. Pasareanu and N. Rungta. Symbolic PathFinder: symbolic execution of Java bytecode. In Charles Pecheur, Jamie Andrews, and Elisabetta Di Nitto, editors, *ASE*, pages 179–180. ACM, 2010.
11. K. Sen, D. Marinov, and G. Agha. CUTE: a concolic unit testing engine for C. In *Proc. of ESEC/SIGSOFT FSE 2005*, pages 263–272. ACM, 2005.
12. T. Ströder, F. Emmes, P. Schneider-Kamp, J. Giesl, and C. Fuhs. A Linear Operational Semantics for Termination and Complexity Analysis of ISO Prolog. In *LOPSTR'11*, pages 237–252. Springer LNCS 7225, 2011.