Online appendix for the paper

Concolic Testing in Logic Programming

published in Theory and Practice of Logic Programming

FRED MESNARD, ÉTIENNE PAYET
LIM - Université de la Réunion, France
(*e-mail:* {fred,epayet}@univ-reunion.fr)

GERMÁN VIDAL
MiST, DSIC, Universitat Politècnica de València
(*e-mail:* gvidal@dsic.upv.es)

submitted 29 April 2015; revised 3rd July 2015; accepted 14 July 2015

In this appendix we report, for the sake of completeness, some auxiliary contents that, for space limitations, we could not include in the paper.

Appendix A Towards Extending Concolic Testing to Full Prolog

In this section, we show a summary of our preliminary research on extending concolic execution to deal with full Prolog. First, we consider the extension of the concrete semantics. Here, we mostly follow the linear semantics of (Ströder et al. 2011), being the main differences that we consider built-ins explicitly, we excluded dynamic predicates for simplicity —but could be added along the lines of (Ströder et al. 2011)— and that, analogously to what we did in Section 2, only the first answer for the initial goal is considered.

In the following, we let the Boolean function defined return true when its argument is an atom rooted by a defined predicate symbol, and false otherwise (i.e., a built-in). Moreover, for evaluating relational and arithmetic expressions, we assume a function eval such that, given an expression e, $\mathsf{eval}(e)$ either returns the evaluation of e (typically a number or a Boolean value) or the special constant error when the expression is not instantiated enough to be evaluated. E.g., $\mathsf{eval}(2 + 2) = 4$, $\mathsf{eval}(3 > 1) = \mathit{true}$, but $\mathsf{eval}(X > 0) = \mathsf{error}$.

The transitions rules are shown in Figure A 1. In the following, we briefly explain the novelties w.r.t. the rules of Section 2:

- In rule choice we use the notation $c[!/!^m]$ to denote a copy of clause c where the occurrences of (possibly labeled) cuts ! at predicate positions (e.g., not inside a call), if any, are replaced by a *labeled* cut $!^m$, where m is a fresh label. Also, in the derived state, we add a *scope delimiter* $?^m$.
- Rule cut removes some alternatives from the current state, while rule cut_fail applies when a goal reaches the scope delimiter without success.
- The rules for call and negation should be clear. Let us only mention that the

notation $A[\mathcal{V}/\mathsf{call}(\mathcal{V}), !/!^m]$ denotes the atom A in which all variables X on predicate positions are replaced by $\mathsf{call}(X)$ and all (possibly labeled) cuts on predicate positions are replaced by $!^m$.

- Calls to the built-in predicate is are dealt with rules is and is_error by means of the auxiliary function eval. Rules rel and rel_error proceed analogously with relational operators like $>$, $<$, $==$, etc.

Regarding the concolic execution semantics, we follow a similar approach to that of Section 3. The labeled transition rules can be seen in Figure A 2. Now, we consider six kinds of labels for $\rightsquigarrow$:

- The labels $\diamond$ and $c(\mathcal{L}_1, \mathcal{L}_2)$ with the same meaning as in the concolic semantics of Section 3.
- The label $u(t_1, t_2)$, which is used to denote a unification step, i.e., the step implies that t_1 and t_2 should unify.
- In contrast, the label $d(t_1, t_2)$ denotes a disunification, i.e., the step implies that t_1 and t_2 should not unify.
- The label $is(X, t)$ denotes a step where is is evaluated (see below).

$$\text{(success)}\ \frac{}{\langle \mathsf{true}_\delta \,|\, S\rangle \rightarrow \langle \textsc{success}_\delta\rangle}$$

$$\text{(failure)}\ \frac{}{\langle (\mathsf{fail}, \mathcal{B})_\delta\rangle \rightarrow \langle \textsc{fail}_\delta\rangle} \qquad \text{(backtrack)}\ \frac{S \neq \epsilon}{\langle (\mathsf{fail}, \mathcal{B}) \,|\, S\rangle \rightarrow \langle S\rangle}$$

$$\text{(choice)}\ \frac{\mathsf{defined}(A) \wedge \mathsf{clauses}(A, \mathcal{P}) = (c_1, \ldots, c_n) \wedge n > 0 \wedge m \text{ is fresh}}{\langle (A, \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle (A, \mathcal{B})_\delta^{c_1[!/!^m]} \,|\, \ldots \,|\, (A, \mathcal{B})_\delta^{c_n[!/!^m]} \mid\ ?_\delta^m \,|\, S\rangle}$$

$$\text{(choice\_fail)}\ \frac{\mathsf{defined}(A, \mathcal{P}) \wedge \mathsf{clauses}(A, \mathcal{P}) = \{\}}{\langle (A, \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle (\mathsf{fail}, \mathcal{B})_\delta \,|\, S\rangle}$$

$$\text{(unfold)}\ \frac{\mathsf{mgu}(A, H_1) = \sigma}{\langle (A, \mathcal{B})_\delta^{H_1 \leftarrow \mathcal{B}_1} \,|\, S\rangle \rightarrow \langle (\mathcal{B}_1\sigma, \mathcal{B}\sigma)_{\delta\sigma} \,|\, S\rangle}$$

$$\text{(cut)}\ \frac{}{\langle (!^m, \mathcal{B})_\delta \,|\, S' \mid\ ?_{\delta'}^m \,|\, S\rangle \rightarrow \langle \mathcal{B}_\delta \mid\ ?_{\delta'}^m \,|\, S\rangle} \qquad \text{(cut\_fail)}\ \frac{}{\langle ?_\delta^m \,|\, S\rangle \rightarrow \langle \mathsf{fail}_\delta \,|\, S\rangle}$$

$$\text{(call)}\ \frac{A \notin \mathcal{V} \wedge m \text{ is fresh}}{\langle (call(A), \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle (A[\mathcal{V}/\mathsf{call}(\mathcal{V}), !/!^m], \mathcal{B})_\delta \mid\ ?_\delta^m \,|\, S\rangle}$$

$$\text{(call\_error)}\ \frac{A \in \mathcal{V}}{\langle (call(A), \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle \textsc{error}_\delta\rangle}$$

$$\text{(not)}\ \frac{m \text{ is fresh}}{\langle (\backslash\!+(A), \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle (\mathsf{call}(A), !^m, \mathsf{fail})_\delta \,|\, \mathcal{B}_\delta \mid\ ?_\delta^m \,|\, S\rangle}$$

$$\text{(unify)}\ \frac{\mathsf{mgu}(t_1, t_2) = \sigma \neq \mathsf{fail}}{\langle (t_1 = t_2, \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle \mathcal{B}\sigma_{\delta\sigma} \,|\, S\rangle} \qquad \text{(unify\_fail)}\ \frac{\mathsf{mgu}(t_1, t_2) = \mathsf{fail}}{\langle (t_1 = t_2, \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle \mathsf{fail}_\delta \,|\, S\rangle}$$

$$\text{(is)}\ \frac{\mathsf{eval}(e_2) = t_2 \neq \mathsf{error}}{\langle (t_1\ \mathsf{is}\ e_2, \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle (t_1 = t_2, \mathcal{B})_\delta \mid\ S\rangle} \qquad \text{(is\_error)}\ \frac{\mathsf{eval}(e_2) = \mathsf{error}}{\langle (t_1\ \mathsf{is}\ e_2, \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle \textsc{error}_\delta\rangle}$$

$$\text{(rel)}\ \frac{\mathsf{eval}(t_1 \oplus t_2) = A \in \{\mathsf{true}, \mathsf{fail}\}}{\langle (t_1 \oplus t_2, \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle (A, \mathcal{B})_\delta \mid\ S\rangle} \qquad \text{(rel\_error)}\ \frac{\mathsf{eval}(t_1 \oplus t_2) = \mathsf{error}}{\langle (t_1 \oplus t_2, \mathcal{B})_\delta \,|\, S\rangle \rightarrow \langle \textsc{error}_\delta\rangle}$$

Fig. A 1. Extended concrete semantics

- Finally, the label $r(A', A)$ denotes that the relational expression A' should be equal to $A \in \{\mathsf{true}, \mathsf{fail}\}$.

In particular, in rules unify and unify_fail, the labels store the unification that must hold in the step. Note that the fact that $\mathsf{mgu}(t_1, t_2) = \mathsf{fail}$ does not imply $\mathsf{mgu}(t_1', t_2') = \mathsf{fail}$ since t_1' and t_2' might be less instantiated than t_1 and t_2.

In rule is, we label the step with $is(X, t_2')$ which means that the fresh variable X

$$\text{(success)}\ \frac{}{\langle \mathsf{true}_\delta \mid S \parallel \mathsf{true}_\theta \mid S' \rangle \rightsquigarrow_\diamond \langle \textsc{success}_\delta \parallel \textsc{success}_\theta \rangle}$$

$$\text{(failure)}\ \frac{}{\langle (\mathsf{fail}, \mathcal{B})_\delta \parallel (\mathsf{fail}, \mathcal{B}')_\theta \rangle \rightsquigarrow_\diamond \langle \mathsf{fail}_\delta \parallel \mathsf{fail}_\theta \rangle}$$

$$\text{(backtrack)}\ \frac{S \neq \epsilon}{\langle (\mathsf{fail}, \mathcal{B}) \mid S \parallel (\mathsf{fail}, \mathcal{B}') \mid S' \rangle \rightsquigarrow_\diamond \langle S \parallel S' \rangle}$$

$$\text{(choice)}\ \frac{\mathsf{defined}(A) \wedge \mathsf{clauses}(A, \mathcal{P}) = \overline{c_n} \wedge n > 0 \wedge m \text{ is fresh} \wedge \mathsf{clauses}(A', \mathcal{P}) = \overline{d_k}}{\begin{array}{l}\langle (A, \mathcal{B})_\delta \mid S \parallel (A', \mathcal{B}')_\theta \mid S' \rangle \\ \quad \rightsquigarrow_{c(\ell(\overline{c_n}), \ell(\overline{d_k}))} \langle (A, \mathcal{B})_\delta^{c_1[!/!^m]} \mid \ldots \mid (A, \mathcal{B})_\delta^{c_n[!/!^m]} \mid\ ?_\delta^m \mid S \\ \qquad \parallel (A', \mathcal{B}')_\theta^{c_1[!/!^m]} \mid \ldots \mid (A', \mathcal{B}')_\theta^{c_n[!/!^m]} \mid\ ?_\theta^m \mid S' \rangle\end{array}}$$

$$\text{(choice\_fail)}\ \frac{\mathsf{defined}(A, \mathcal{P}) \wedge \mathsf{clauses}(A, \mathcal{P}) = \{\} \wedge \mathsf{clauses}(A', \mathcal{P}) = \overline{c_k}}{\langle (A, \mathcal{B})_\delta \mid S \parallel (A', \mathcal{B}')_\theta \mid S' \rangle \rightsquigarrow_{c(\{\}, \ell(\overline{c_k}))} \langle (\mathsf{fail}, \mathcal{B})_\delta \mid S \parallel (\mathsf{fail}, \mathcal{B}')_\theta \mid S' \rangle}$$

$$\text{(unfold)}\ \frac{\mathsf{mgu}(A, H_1) = \sigma \wedge \mathsf{mgu}(A', H_1) = \sigma'}{\langle (A, \mathcal{B})_\delta^{H_1 \leftarrow \mathcal{B}_1} \mid S \parallel (A', \mathcal{B}')_\theta^{H_1 \leftarrow \mathcal{B}_1} \mid S' \rangle \rightsquigarrow_\diamond \langle (\mathcal{B}_1\sigma, \mathcal{B}\sigma)_{\delta\sigma} \mid S \parallel (\mathcal{B}_1\sigma', \mathcal{B}'\sigma')_{\theta\sigma'} \mid S' \rangle}$$

$$\text{(cut)}\ \frac{}{\langle (!^m, \mathcal{B})_\delta \mid S_1 \mid\ ?_{\delta'}^m \mid S \parallel (!^m, \mathcal{B}')_\theta \mid S_1' \mid\ ?_{\theta'}^m \mid S' \rangle \rightsquigarrow_\diamond \langle \mathcal{B}_\delta \mid\ ?_{\delta'}^m \mid S \parallel \mathcal{B}_\theta' \mid\ ?_{\theta'}^m \mid S' \rangle}$$

$$\text{(cut\_fail)}\ \frac{}{\langle ?_\delta^m \mid S \parallel ?_\theta^m \mid S' \rangle \rightsquigarrow_\diamond \langle \mathsf{fail}_\delta \mid S \parallel \mathsf{fail}_\theta \mid S' \rangle}$$

$$\text{(call)}\ \frac{A \notin \mathcal{V} \wedge m \text{ is fresh}}{\begin{array}{l}\langle (call(A), \mathcal{B})_\delta \mid S \parallel (call(A'), \mathcal{B}')_\theta \mid S' \rangle \\ \quad \rightsquigarrow_\diamond \langle (A[\mathcal{V}/\mathsf{call}(\mathcal{V}), !/!^m], \mathcal{B})_\delta \mid\ ?_\delta^m \mid S \parallel (A'[\mathcal{V}/\mathsf{call}(\mathcal{V}), !/!^m], \mathcal{B}')_\theta \mid\ ?_\theta^m \mid S' \rangle\end{array}}$$

$$\text{(call\_error)}\ \frac{A \in \mathcal{V}}{\langle (call(A), \mathcal{B})_\delta \mid S \parallel (call(A'), \mathcal{B}')_\theta \mid S' \rangle \rightsquigarrow_\diamond \langle \mathsf{error}_\delta \parallel \mathsf{error}_\theta \rangle}$$

$$\text{(not)}\ \frac{m \text{ is fresh}}{\begin{array}{l}\langle (\backslash+(A), \mathcal{B})_\delta \mid S \parallel (\backslash+(A'), \mathcal{B}')_\theta \mid S' \rangle \\ \quad \rightsquigarrow_\diamond \langle (\mathsf{call}(A), !^m, \mathsf{fail})_\delta \mid \mathcal{B}_\delta \mid\ ?_\delta^m \mid S \parallel (\mathsf{call}(A'), !^m, \mathsf{fail})_\theta \mid \mathcal{B}_\theta' \mid\ ?_\theta^m \mid S' \rangle\end{array}}$$

$$\text{(unify)}\ \frac{\mathsf{mgu}(t_1, t_2) = \sigma \wedge \mathsf{mgu}(t_1', t_2') = \sigma'}{\langle (t_1 = t_2, \mathcal{B})_\delta \mid S \parallel (t_1' = t_2', \mathcal{B}')_\theta \mid S' \rangle \rightsquigarrow_{u(t_1', t_2')} \langle \mathcal{B}\sigma_{\delta\sigma} \mid S \parallel \mathcal{B}'\sigma'_{\delta\sigma'} \mid S' \rangle}$$

$$\text{(unify\_fail)}\ \frac{\mathsf{mgu}(t_1, t_2) = \mathsf{fail}}{\langle (t_1 = t_2, \mathcal{B})_\delta \mid S \parallel (t_1' = t_2', \mathcal{B}')_\theta \mid S' \rangle \rightsquigarrow_{d(t_1', t_2')} \langle \mathsf{fail}_\delta \mid S \parallel \mathsf{fail}_\theta \mid S' \rangle}$$

$$\text{(is)}\ \frac{\mathsf{eval}(e_2) = t_2 \neq \mathsf{error} \wedge \mathsf{sym\_eval}(e_2') = t_2' \wedge X \text{ is fresh}}{\langle (t_1 \text{ is } e_2, \mathcal{B})_\delta \mid S \parallel (t_1' \text{ is } e_2', \mathcal{B}')_\theta \mid S' \rangle \rightsquigarrow_{is(X, t_2')} \langle (t_1 = t_2, \mathcal{B})_\delta \mid\ S \parallel (t_1' = X, \mathcal{B}')_\theta \mid\ S' \rangle}$$

$$\text{(is\_error)}\ \frac{\mathsf{eval}(e_2) = \mathsf{error}}{\langle (t_1 \text{ is } e_2, \mathcal{B})_\delta \mid S \parallel (t_1' \text{ is } e_2', \mathcal{B}')_\theta \mid S' \rangle \rightsquigarrow_\diamond \langle \mathsf{error}_\delta \parallel \mathsf{error}_\theta \rangle}$$

$$\text{(rel)}\ \frac{\mathsf{eval}(t_1 \oplus t_2) = A \in \{\mathsf{true}, \mathsf{fail}\} \wedge \mathsf{sym\_eval}(t_1' \oplus t_2') = A'}{\langle (t_1 \oplus t_2, \mathcal{B})_\delta \mid S \parallel (t_1' \oplus t_2', \mathcal{B}')_\theta \mid S' \rangle \rightsquigarrow_{r(A', A)} \langle (A, \mathcal{B})_\delta \mid\ S \parallel (A', \mathcal{B}')_\theta \mid\ S' \rangle}$$

$$\text{(rel\_error)}\ \frac{\mathsf{eval}(t_1 \oplus t_2) = \mathsf{error}}{\langle (t_1 \oplus t_2, \mathcal{B})_\delta \mid S \parallel (t_1' \oplus t_2', \mathcal{B}')_\theta \mid S' \rangle \rightsquigarrow_\diamond \langle \mathsf{error}_\delta \parallel \mathsf{error}_\theta \rangle}$$

Fig. A 2. Extended concolic execution semantics

should be bound to the evaluation of t_2' after grounding it. Note that introducing such a fresh variable is required to avoid a failure in the subsequent step with rule unify because of, e.g., a non-ground arithmetic expression that could not be evaluated yet to a value using function sym_eval. Note that rule is_error does not include any label since we assume that an error in the concrete computation just aborts the execution and also the test case generation process.

Finally, in rule rel we label the step with $r(A', A)$ where A is the value true/fail of the relational expression in the concrete goal, and A' is a (possibly nonground) corresponding expression in the symbolic goal. Here, we use the auxiliary function sym_eval to simplify the relational expression as much as possible. E.g., $\mathsf{sym\_eval}(3 > 0) = \mathsf{true}$ but $\mathsf{sym\_eval}(3 + 2 > X) = 5 > X$.

These labels can be used for extending the concolic testing algorithm of Section 4. For instance, given a concolic execution step labeled with $r(X > 0, \mathsf{true})$, we have that solving $\neg(X > 0)$ will produce a binding for X (e.g., $\{X/0\}$) that will follow an alternative path. Here, the concolic testing procedure will integrate a constraint solver for producing solutions to negated constraints. We find this extension of the concolic testing procedure an interesting topic for future work.

Appendix B Proofs of Technical Results

B.1 Concolic Execution Semantics

Proof of Theorem 1
Since the base case $i = 0$ trivially holds, in the following we only consider the inductive case $i > 0$. Let $C_i = \langle \mathcal{B}_\delta^c \mid S \,[\![\, \mathcal{D}_\theta^{c'} \mid S' \rangle$. By the inductive hypothesis, we have $|S| = |S'|$, $\mathcal{D} \leqslant \mathcal{B}$, $c = c'$ (if any), and $p(\overline{X_n})\theta \leqslant p(\overline{t_n})\delta$. Now, we consider the step $C_i \rightsquigarrow C_{i+1}$ and distinguish the following cases, depending on the applied rule:

- If the rule applied is success, failure, backtrack or choice_fail, the claim follows trivially by induction.
- If the rule applied is choice, let us assume that we have $\mathcal{B} = (A, \mathcal{B}')$, $\mathcal{D} = (A', \mathcal{D}')$ and $\mathsf{clauses}(A, \mathcal{P}) = \overline{c_j}$, $j > 0$. Therefore, we have $C_{i+1} = \langle \mathcal{B}_\delta^{c_1} \mid \ldots \mid \mathcal{B}_\delta^{c_j} \mid S \,[\![\, \mathcal{D}_\theta^{c_1} \mid \ldots \mid \mathcal{D}_\theta^{c_j} \mid S' \rangle$, and the claim follows straightforwardly by the induction hypothesis.
- Finally, if the applied rule is unfold, then we have that $\mathcal{B}_\delta^c = (A, \mathcal{B}')_\delta^c$, $\mathcal{D}_\theta^c = (A', \mathcal{D}')_\theta^c$ for some clause $c = H_1 \leftarrow \mathcal{B}_1$. Therefore, we have $C_{i+1} = \langle (\mathcal{B}_1\sigma, \mathcal{B}'\sigma)_{\delta\sigma} \mid S \,[\![\, (\mathcal{B}_1\sigma', \mathcal{D}'\sigma')_{\theta\sigma'} \mid S' \rangle$, where $\mathsf{mgu}(A, H_1) = \sigma$ and $\mathsf{mgu}(A', H_1) = \sigma'$. First, $c = c'$ holds by vacuity since the goals are not labeled with a clause. Also, the number of concrete and symbolic goals is trivially the same since $|S| = |S'|$ by the inductive hypothesis. Now, by the inductive hypothesis, we have $\mathcal{D} \leqslant \mathcal{B}$ and thus $A' \leqslant A$ and $\mathcal{D}' \leqslant \mathcal{B}'$. Then, since $\sigma = \mathsf{mgu}(A, H_1)$, $\sigma' = \mathsf{mgu}(A', H_1)$, $\mathcal{V}ar(H_1 \leftarrow \mathcal{B}_1) \cap \mathcal{V}ar(A) = \{\}$, and $\mathcal{V}ar(H_1 \leftarrow \mathcal{B}_1) \cap \mathcal{V}ar(A') = \{\}$, it is easy to see that $A'\sigma' \leqslant A\sigma$ (and thus $\mathcal{D}'\sigma' \leqslant \mathcal{B}'\sigma$) and $\sigma' \leqslant \sigma$ when restricted to the variables of H_1 (and thus $\mathcal{B}_1\sigma' \leqslant \mathcal{B}_1\sigma$). Therefore, we can conclude $(\mathcal{B}_1\sigma', \mathcal{D}'\sigma') \leqslant (\mathcal{B}_1\sigma, \mathcal{B}'\sigma)$. Finally, using a similar argument, we have $p(\overline{X_n})\theta\sigma' \leqslant p(\overline{t_n})\delta\sigma$.

□

B.2 Solving Unifiability Problems

First, we prove the following invariant which justifies that the algorithm in Definition 6 is well defined.

Proposition 1
The following statement is an invariant of the loops at lines 2 and 3 of the algorithm in Definition 6:

(invariant) (a) $A \approx B$ for all $B \in \mathcal{B}$ and (b) $A \leq B'$ for some $B' \in \mathcal{B}$.

Proof
Let us first consider the loop at line 2. Clearly, the invariant holds upon initialization. Therefore, let us assume that it holds for some arbitrary set $\mathcal{B}$ and we prove it also holds for $\mathcal{B}' = \mathcal{B}\eta$ with $\eta = \{X/t\}$ for some simple disagreement pair X, t (or t, X). Let us consider part (a). Since $A \approx B$ for all $B \in \mathcal{B}$, there exist a substitution θ such that $A\theta = B\theta$ for all $B \in \mathcal{B}$. Consider such an arbitrary $B \in \mathcal{B}$. If $X \notin \mathcal{V}ar(B)$, then part (a) of the invariant holds trivially in $\mathcal{B}'$. Otherwise, $\theta\{X/t\}$ is clearly a unifier A and B, and it also holds. Consider now part (b). Since $A \leq B'$ for some $B' \in \mathcal{B}$, there exists a substitution σ such that $A\sigma = B'$. Using a similar argument as before, either $A\sigma = B'$ with $B' \in \mathcal{B}'$ or $A\sigma\{X/t\} = B'\{X/t\}$ with $B'\{X/t\} \in \mathcal{B}$, and part (b) of the invariant also holds in $\mathcal{B}'$.

Let us now consider the loop at line 3. Clearly, the invariant holds when the previous loop terminates. Let t, t' be the selected disagreement pair. Then t, t' is replaced in $\mathcal{B}$ by a fresh variable $U \in \mathcal{U}$, thus obtainining a new set $\mathcal{B}'$. Let $\eta_1 := \{U/t\}$ and $\eta_2 := \{U/t'\}$. Both η_1 and η_2 are idempotent substitutions because $U \notin \mathcal{V}ar(t)$ and $U \notin \mathcal{V}ar(t')$ since U is fresh. Let B_1, B_2 be the atoms of $\mathcal{B}$ where t, t' come from and C_1, C_2 be the atoms obtained by replacing t, t' in B_1, B_2 by U. Then $B_1 = C_1\eta_1$ and $B_2 = C_2\eta_2$. Now, we want to prove that the invariant also holds in $\mathcal{B}' = \mathcal{B} \setminus \{B_1, B_2\} \cup \{C_1, C_2\}$. Part (a) is trivial, since we only generalize some atoms: if A unify with B_1 and B_2, it will also unify with C_1 and C_2. Regarding part (b), we have that $A \leq B'$ for some $B' \in \mathcal{B}$. Clearly, part (b) also holds in $\mathcal{B}'$ if B' is different from B_1 and B_2. Otherwise, w.l.o.g., assume that $B' = B_1$ and $A \leq B_1$. Since $A \approx B_1$ and $A \approx B_2$, and t, t' is a disagreement pair for B_1, B_2, we have that the subterm of A that corresponds to the position of t, t' should be more general than t, t' (otherwise, it would not unify with both terms). Therefore, replacing t by a fresh variable U will not change that, and we have $A \leq C_1$ for some $C_1 \in \mathcal{B}$. □

The following auxiliary results are useful to prove the correctness of the algorithms in Definitions 6 and 7.

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 $[Dom(\eta) \cap \mathcal{V}ar(B)] \cap Dom(\theta) = \{\}$ and $Ran(\eta) \cap Dom(\theta\eta) = \{\}$.

Proof
For any $X \in \mathcal{V}ar(B)$,

- either $X \notin Dom(\eta)$ and then $X\eta\theta\eta = X\theta\eta$
- or $X \in Dom(\eta)$ and then $X\eta\theta\eta = (X\eta)\theta\eta = X\eta$ because $Ran(\eta) \cap Dom(\theta\eta) = \{\}$. Moreover, $X \notin Dom(\theta)$ because $[Dom(\eta) \cap \mathcal{V}ar(B)] \cap Dom(\theta) = \{\}$, so $X\theta\eta = X\eta$. Finally, $X\eta\theta\eta = X\theta\eta$.

Consequently, $B\eta\theta\eta = B\theta\eta$. As $A\theta = B\theta$, we have $A\theta\eta = B\theta\eta$ i.e. $A\theta\eta = B\eta\theta\eta$. □

Proposition 2
The loop at line 2 always terminates and the following statement is an invariant of this loop.

(inv) For each $A' \in \{A\} \cup \mathcal{H}_{pos}$ there exists $B \in \mathcal{B}$ and a substitution θ such that $A'\theta = B\theta$ and $\mathcal{V}ar(\mathcal{B}) \cap Dom(\theta) = \{\}$.

Proof
Action (2b) reduces the number of simple disagreement pairs in $\mathcal{B}$ which implies termination of the loop at line 2.

Let us prove that (inv) is an invariant. First, (inv) clearly holds upon initialization of $\mathcal{B}$. Suppose it holds prior to an execution of action (2b). Therefore, for each $A' \in \{A\} \cup \mathcal{H}_{pos}$ there exists $B \in \mathcal{B}$ and a substitution θ such that $A'\theta = B\theta$ and $\mathcal{V}ar(\mathcal{B}) \cap Dom(\theta) = \{\}$. Let t, t' be the selected simple disagreement pair. Then, we consider a substitution η determined by t, t'. For any $X \in Ran(\eta)$, we have $X \in \mathcal{V}ar(\mathcal{B})$. Thus $X \notin Dom(\theta)$ by (inv). Hence $Ran(\eta) \cap Dom(\theta) = \{\}$. Moreover, as t, t' is a simple pair we have $Ran(\eta) \cap Dom(\eta) = \{\}$. Hence,

$$Ran(\eta) \cap Dom(\theta\eta) = \{\} \ . \tag{B1}$$

Since $B \in \mathcal{B}$, we have $[Dom(\eta) \cap \mathcal{V}ar(B)] \cap Dom(\theta) = \{\}$. Consequently, by (B1) and Lemma 1 we have

$$A'\theta\eta = B\eta\theta\eta \ .$$

Now, we want to prove that (inv) holds for $\mathcal{B}\eta$, i.e., that for each $A' \in \{A\} \cup \mathcal{H}_{pos}$ there exists $B\eta \in \mathcal{B}\eta$ and a substitution θ' such that $A'\theta' = B\eta\theta'$ and $\mathcal{V}ar(\mathcal{B}\eta) \cap Dom(\theta') = \{\}$. We let $\theta' = \theta\eta$, so $A'\theta\eta = B\eta\theta\eta$ holds. Now, suppose by contradiction that $\mathcal{V}ar(\mathcal{B}\eta) \cap Dom(\theta\eta) \neq \{\}$, and let X be one of its elements. We have $X \notin Dom(\eta)$ because $Ran(\eta) \cap Dom(\eta) = \{\}$, so $X \in Dom(\theta)$. Moreover, $X \notin Ran(\eta)$ by (B1) so $X \in \mathcal{V}ar(\mathcal{B})$. Therefore, $X \in \mathcal{V}ar(\mathcal{B}) \cap Dom(\theta)$ which by (inv) gives a contradiction. Consequently,

$$\mathcal{V}ar(\mathcal{B}\eta) \cap Dom(\theta\eta) = \{\}$$

and the claim follows. □

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

(inv′) For each $A' \in \{A\} \cup \mathcal{H}_{pos}$ there exists $B \in \mathcal{B}$ and a substitution θ such that $A'\theta = B\theta$ and $\mathcal{V}ar(\mathcal{B}) \cap Dom(\theta) \subseteq \mathcal{U}$.

Proof
Action (3b) reduces the number of disagreement pairs in $\mathcal{B}$ which implies termination of the loop at line 3.

Let us prove that (inv′) is an invariant. By Proposition 2, (inv) holds upon termination of the loop at line 2, hence (inv′) holds just before execution of the loop at line 3. Suppose it holds prior to an execution of action (3b), so we have that, for each $A' \in \{A\} \cup \mathcal{H}_{pos}$ there exists $B \in \mathcal{B}$ and a substitution θ such that $A'\theta = B\theta$ and $\mathcal{V}ar(\mathcal{B}) \cap Dom(\theta) \subseteq \mathcal{U}$. Let t, t' be the selected disagreement pair. Then t, t' is replaced in $\mathcal{B}$ by a fresh variable $U \in \mathcal{U}$, thus obtainining a new set $\mathcal{B}'$. Let $\eta_1 := \{U/t\}$ and $\eta_2 := \{U/t'\}$. Both η_1 and η_2 are idempotent substitutions because $U \notin \mathcal{V}ar(t)$ and $U \notin \mathcal{V}ar(t')$ since U is fresh. Let B_1, B_2 be the atoms of $\mathcal{B}$ where t, t' come from and C_1, C_2 be the atoms obtained by replacing t, t' in B_1, B_2 by U. Then $B_1 = C_1\eta_1$ and $B_2 = C_2\eta_2$. Now, we want to prove that (inv′) holds in $\mathcal{B}' = \mathcal{B} \setminus \{B_1, B_2\} \cup \{C_1, C_2\}$, i.e., that for each $A' \in \{A\} \cup \mathcal{H}_{pos}$ there exists $B \in \mathcal{B}'$ and a substitution θ such that $A'\theta = B\theta$ and $\mathcal{V}ar(\mathcal{B}') \cap Dom(\theta) \subseteq \mathcal{U}$.

Since (inv′) holds in $\mathcal{B}$, we have $A'\theta = B\theta$. Moreover, $A' = A'\eta_1 = A'\eta_2$ because U does not occur in A'. So if $B = B_1$ then $A'\eta_1\theta = C_1\eta_1\theta$ and if $B = B_2$ then $A'\eta_2\theta = C_2\eta_2\theta$. Consequently, let us set

- $\theta' := \theta$ and $B' := B$ if $B \notin \{B_1, B_2\}$
- $\theta' := \eta_1\theta$ and $B' := C_1$ if $B = B_1$
- $\theta' := \eta_2\theta$ and $B' := C_2$ if $B = B_2$.

Then we have

$$A'\theta' = B'\theta' \ . \tag{B2}$$

Moreover, $Dom(\theta') \subseteq Dom(\theta) \cup Dom(\eta_1) \cup Dom(\eta_2)$ i.e.

$$Dom(\theta') \subseteq Dom(\theta) \cup \{U\} \ . \tag{B3}$$

As $\mathcal{V}ar(C_1, C_2) \subseteq \mathcal{V}ar(B_1, B_2) \cup \{U\}$ then

$$\mathcal{V}ar(C_2, C_2) \cap Dom(\theta') \subseteq \mathcal{U}$$

because $\mathcal{V}ar(B_1, B_2) \cap Dom(\theta) \subseteq \mathcal{U}$ by (inv′) and $\mathcal{V}ar(B_1, B_2) \cap \{U\} = \{U\} \cap Dom(\theta) = \{\}$ and $\{U\} \cap \{U\} \subseteq \mathcal{U}$. Moreover, by (inv′) we have $\mathcal{V}ar(\mathcal{B}) \cap (Dom(\theta) \cup \{U\}) \subseteq \mathcal{U}$ so by (B3)

$$\mathcal{V}ar(\mathcal{B}) \cap Dom(\theta') \subseteq \mathcal{U} \ .$$

Hence, $\mathcal{V}ar(\mathcal{B} \setminus \{B_1, B_2\} \cup \{C_1, C_2\}) \cap Dom(\theta') \subseteq \mathcal{U}$. With (B2) this implies that upon termination of action (3b) the invariant (inv′) holds because B_1 is set to C_1 and B_2 to C_2. □

The correctness of the algorithm in Definition 6 is then stated as follows.

Theorem 1
Let A be an atom and $\mathcal{H}_{pos}$ be a set of atoms such that $\mathcal{V}ar(\{A\} \cup \mathcal{H}_{pos}) \cap \mathcal{U} = \{\}$ and $A \approx B$ for all $B \in \mathcal{H}_{pos}$. The algorithm in Definition 6 with input A and $\mathcal{H}_{pos}$ always terminates and returns a substitution θ such that $A\theta\eta$ unifies with all the atoms of $\mathcal{H}_{pos}$ for any idempotent substitution η with $Dom(\eta) \subseteq \mathcal{V}ar(A\theta)$ and $\mathcal{V}ar(\eta) \cap \mathcal{U} = \{\}$.

Proof
Proposition 2 and Proposition 3 imply termination of the algorithm. Upon termination of the loop at line 3 we have $|\mathcal{B}| = 1$. Let B be the element of $\mathcal{B}$ with $A\theta = B$. Now, we want to prove that $A\theta\eta$ unifies with all the atoms in $\mathcal{H}_{pos}$ for any idempotent substitution η (i.e., $Dom(\eta) \cap Ran(\eta) = \{\}$) such that $Dom(\eta) \subseteq \mathcal{V}ar(A\theta) = \mathcal{V}ar(B)$ and $\mathcal{V}ar(\eta) \cap \mathcal{U} = \{\}$. By Proposition 3, we have that, for all $B' \in \mathcal{H}_{pos}$, there exists a substitution θ' such that $B\theta' = B'\theta'$ and $\mathcal{V}ar(B) \cap Dom(\theta') \subseteq \mathcal{U}$. From all the previous conditions, it follows that $[Dom(\eta) \cap \mathcal{V}ar(B)] \cap Dom(\theta') = \{\}$ and $Ran(\eta) \cap Dom(\theta'\eta) = \{\}$. Therefore, by Lemma 1, we have $B\eta\theta'\eta = B'\theta'\eta$. Finally, since $A\theta = B$, we have $A\theta\eta\theta'\eta = B'\theta'\eta$ and, thus, $A\theta\eta$ unifies with B'. □

Proof of Theorem 2
Each step of the algorithm terminates, hence the algorithm terminates. Assume that the algorithm returns a substitution σ. The set $G\sigma$ is ground by construction. By Theorem 1, we have that $A\sigma = A\theta\eta$ unifies with all the atoms in $\mathcal{H}_{pos}$ as long as η is idempotent, $Dom(\eta) \subseteq \mathcal{V}ar(A\theta)$ and $\mathcal{V}ar(\eta) \cap \mathcal{U} = \{\}$. Finally, the last check ensures that $A\sigma$ does not unify with any atom of $\mathcal{H}_{neg}$. □

B.2.1 Completeness

For simplicity, we ignore the groundness constraint in this section. Therefore, we now focus on the completeness of the following unification problem: Let A be an atom and $\mathcal{H}_{pos}, \mathcal{H}_{neg}$ be sets of atoms such that $A \approx B$ for all $B \in \mathcal{H}_{pos} \cup \mathcal{H}_{neg}$. Then, we want to find a substitution σ such that

$$A\sigma \approx B \text{ for all } B \in \mathcal{H}_{pos} \text{ but } \neg(A\sigma \approx B') \text{ for all } B' \in \mathcal{H}_{neg} (**)$$

We further assume that all atoms are renamed apart.

Let us first formalize the notion of unifying substitution:

Definition 1 (*unifying substitution*)
Let A be an atom and let $\mathcal{B}$ be a set of atoms such that $\mathcal{V}ar(A, \mathcal{B}) \cap \mathcal{U} = \{\}$ and $A \approx B$ for all $B \in \mathcal{B}$. We say that σ is a unifying substitution for A w.r.t. $\mathcal{B}$ if $A\sigma \approx B$ for all $B \in \mathcal{B}$.

In particular, we are interested in *maximal* unifying substitutions computed by the algorithm in Definition 6. The relevance of maximal unifying substitutions is that variables from $\mathcal{U}$ identify where further instantiation would result in a substitution which is not a unifying substitution anymore. For the remaining positions, we basically return their most general unifier.

Now, we prove that binding an atom A with a maximal unifying substitution for A w.r.t. $\mathcal{H}_{pos}$ does not affect to the existence of a solution to our unification problem (**) above. Here, for simplicity, we assume that only *most specific* solutions are considered, where a solution σ is called a *most specific* solution for A and $\mathcal{H}_{pos}, \mathcal{H}_{neg}$ if there exists no other solution which is strictly less general than σ. Furthermore, we also assume that the atom A has the form $p(X_1, \ldots, X_n)$.

Lemma 2
Let A be an atom and $\mathcal{H}_{pos}, \mathcal{H}_{neg}$ be sets of atoms such that $A \approx B$ for all $B \in \mathcal{H}_{pos} \cup \mathcal{H}_{neg}$. If there exists a substitution σ such that $A\sigma \approx B$ for all $B \in \mathcal{H}_{pos}$ and $\neg(A\sigma \approx B)$ for all $B \in \mathcal{H}_{neg}$, then there exists a maximal unifying substitution θ and a substitution σ' such that $A\theta\sigma' \approx B$ for all $B \in \mathcal{H}_{pos}$ and $\neg(A\theta\sigma' \approx B)$ for all $B \in \mathcal{H}_{neg}$.

Proof
(*sketch*) Let us consider the stages of the algorithm in Definition 6 with input $\mathcal{H}_{pos}$ (atom A is not needed since it has the form $p(X_1, \ldots, X_n)$ and, thus, imposes no constraint). The first stage just propagates simple disagreement pairs of the form X, t or t, X. When X only occurs once, it is easy to see that σ is also a (most specific) unifying substitution for A w.r.t. $\mathcal{H}_{pos}\{X/t\}$. Consider, e.g., that σ contains a binding of the form $X_i/C[t']$ for some $i \in \{1, \ldots, n\}$ and context $C[\,]$ and such that t' corresponds to the same position of X and t in $\mathcal{H}_{pos}$. Depending on the terms in the corresponding position of the remaining atoms, we might have $t' \leq t$ or $t \leq t'$. Either case, replacing X by t will not change the fact that σ is still a most specific unifying substitution for $\mathcal{H}_{pos}\{X/t\}$.

The step is more subtle when there are several simple disagreement pairs for a given variable, e.g., X, t_1 and X, t_2 (we could generalize it to an arbitrary number of pairs, but two are enough to illustrate how to proceed). In this case, if $t_1 \leq t_2$, we choose X, t_2 and the reasoning is analogous to the previous case. However, when neither $t_1 \leq t_2$ not $t_2 \leq t_1$, the algorithm in Definition 6 is non-deterministic and allows us to choose any of them. As before, let us consider that σ contains bindings of the form $X_i/C[t'_1]$ and $X_j/C'[t'_2]$ for some $i, j \in \{1, \ldots, n\}$ and contexts $C[\,], C'[\,]$ and such that t'_1 and t'_2 correspond to the same positions of t_1 and t_2 in $\mathcal{H}_{pos}$, respectively. Here, assuming there are no further constraints from the remaining atoms, a most specific unifying substitution might either bind X_i to $C[t_1]$ and leave X_j unconstrained (e.g., bound to a fresh variable) or the other way around: bind X_i to $C[t_2]$ and leave X_j unconstrained. Here, we choose the same alternative as in the considered solution σ, say X_i is bound to $C[t_1]$. Therefore, σ is still a unifying substitution for A w.r.t. $\mathcal{H}_{pos}\{X/t_1\}$. Note that the new (non-simple) disagreement pair t_1, t_2 introduced in $\mathcal{H}_{pos}\{X/t_1\}$ will be generalized away in the next stage (and replaced by a fresh variable from $\mathcal{U}$).

Therefore, when the first stage is completed (i.e., step 2 in Definition 6), we have propagated some terms from one atom to the remaining ones –as in the computation of a most general unifier– thus producing a new set $\mathcal{H}'_{pos}$ such that σ is still a (most specific) unifying substitution for A w.r.t. $\mathcal{H}'_{pos}$.

By definition, after this stage, there are no simple disagreement pairs in $\mathcal{H}'_{pos}$. Then, in the second stage (step 3 in Definition 6), we replace every (non-simple) disagreement pair t_1, t_2 by a fresh variable U from $\mathcal{U}$. Since σ was a unifying substitution for $\mathcal{H}'_{pos}$, it should have a binding $X_i/C[W]$ for some $i \in \{1, \ldots, n\}$ and context $C[\ ]$ and such that W corresponds to the same position of t_1 and t_2 in $\mathcal{H}'_{pos}$, where W is a variable. Therefore, replacing t, t' by a fresh variable U will not change the fact that σ is still a unifying substitution for the resulting set (up to variable renaming).

Hence, when the second stage is finished, we have a new set $\mathcal{H}''_{pos}$ without any disagreement pair at all, i.e., $\mathcal{H}''_{pos} = \{B\}$ with $A\theta = B$. Moreover, since σ is a most specific uniyfing substitution for A w.r.t. $\mathcal{H}''_{pos}$, we have $\theta \leq \sigma\ [\mathcal{V}ar(A)]$. Therefore, there exists a substitution σ' such that $A\sigma = A\theta\sigma'$ such that σ' is a solution for $A\theta$ and $\mathcal{H}_{pos}, \mathcal{H}_{neg}$, which concludes the proof. □

Appendix C Some More Examples on Solving Unifiability Problems

Example 1 (*maximal unifying substitution*)
Let $A = p(X, Y)$ and $\mathcal{H}_{pos} = \{p(s(a), s(c)), p(s(b), s(c)), p(Z, Z)\}$. First the algorithm of Definition 6 sets $\mathcal{B} := \{p(X, Y), p(s(a), s(c)), p(s(b), s(c)), p(Z, Z)\}$, then it considers the simple disagreement pairs in $\mathcal{B}$. The substitution $\eta_1 := \{X/s(a)\}$ is determined by $X, s(a)$. Action (2b) sets $\mathcal{B}$ to $\mathcal{B}\eta_1$ i.e. to

$$\{p(s(a), Y), p(s(a), s(c)), p(s(b), s(c)), p(Z, Z)\}\ .$$

The substitution $\eta_2 := \{Y/s(c)\}$ is determined by $Y, s(c)$. Action (2b) sets $\mathcal{B}$ to $\mathcal{B}\eta_2 = \{p(s(a), s(c)), p(s(b), s(c)), p(Z, Z)\}$. The substitution $\eta_3 := \{Z/s(c)\}$ is determined by $Z, s(c)$. Action (2b) sets $\mathcal{B}$ to $\mathcal{B}\eta_3$ i.e. to

$$\{p(s(a), s(c)), p(s(b), s(c)), p(s(c), s(c))\}\ .$$

Now no simple disagreement pair occurs in $\mathcal{B}$ hence the algorithm skips to the loop at line 3.

- Action (3b) replaces the disagreement pair a, b with a fresh variable $U \in \mathcal{U}$, hence $\mathcal{B}$ is set to $\{p(s(U), s(c)), p(s(c), s(c))\}$.
- Action (3b) replaces the disagreement pair U, c with a fresh variable $U' \in \mathcal{U}$, hence $\mathcal{B}$ is set to $\{p(s(U'), s(c))\}$.

As $|\mathcal{B}| = 1$ the loop at line 3 stops and the algorithm returns the substitution $\{X/s(U'), Y/s(c)\}$.

Note that there are several non-deterministic possibilities for η_1, η_2 and η_3. For instance, if we consider $\eta_3 := \{Z/s(a)\}$, which is determined by $Z/s(a)$, then $\mathcal{B}$ is set to $\{p(s(a), s(c)), p(s(b), s(c)), p(s(a), s(a))\}$. The loop at line 3 finally sets $\mathcal{B}$ to $\{p(s(U), s(U'))\}$, so the algorithm returns the substitution $\{X/s(U), Y/s(U')\}$.

We note that the set $\mathcal{B}$ used by the algorithm of Definition 6 may contain several occurrences of a same, non-simple, disagreement pair.

Example 2 (*maximal unifying substitution*)
Let $A = p(X, Y)$ and $\mathcal{H}_{pos} = \{p(a, a), p(b, b)\}$. First the algorithm sets $\mathcal{B} := \{p(X, Y), p(a, a), p(b, b)\}$. Then the loop at line 2 considers the simple disagreement pairs in $\mathcal{B}$ and, for instance, it sets $\mathcal{B}$ to $\{p(a, a), p(b, b)\}$ (it may also set $\mathcal{B}$ to $\{p(a, b), p(a, a), p(b, b)\}$ or to $\{p(b, a), p(a, a), p(b, b)\}$). As no simple disagreement pair now occurs in $\mathcal{B}$, the algorithm jumps at line 3. The pair a, b occurs twice in $\mathcal{A}$. Action (3b) replaces each occurrence with the same variable $U \in \mathcal{U}$, so the loop at line 3 sets $\mathcal{B}$ to $\{p(U, U)\}$ and the algorithm returns $\{X/U, Y/U\}$.

Example 3 (*maximal unifying substitution*)
Let $A = p(X, Y)$ and $\mathcal{H}_{pos} = \{p(a, b), p(b, a)\}$. First the algorithm sets $\mathcal{B} := \{p(X, Y), p(a, b), p(b, a)\}$. Then the loop at line 2 considers the simple disagreement pairs in $\mathcal{B}$ and, for instance, it sets $\mathcal{B}$ to $\{p(a, b), p(b, a)\}$ (it may also set $\mathcal{B}$ to $\{p(a, a), p(a, b), p(b, a)\}$ or to $\{p(b, b), p(a, b), p(b, a)\}$). As no simple disagreement pair now occurs in $\mathcal{B}$, the algorithm jumps at line 3. The pairs a, b and b, a occur once in $\mathcal{A}$ and Action (3b) replaces them with two different variables $U, U' \in \mathcal{U}$. So the loop at line 3 sets $\mathcal{B}$ to $\{p(U, U')\}$ and the algorithm returns $\{X/U, Y/U'\}$.