Appendix to “Termination of Simply-Moded Well-Typed Logic Programs under a Tabled Execution Mechanism”: Proofs

Sofie Verbaeten

Danny De Schreye

Abstract

This is an appendix to the paper “Termination of Simply-Moded Well-Typed Logic Programs under a Tabled Execution Mechanism”, published in the journal *Applicable Algebra in Engineering, Communication and Computing*. It contains the proofs of the propositions and theorems in that paper.

A Proof of Theorem 4.1

The following concepts will be useful in the proof of Theorem 4.1.

Definition A.1 (direct descendant) Let P be a program and $\tilde{A} \in B_P^E$. Let $D : \leftarrow A = G_0, G_1, \dots$ be an LD-derivation of $\leftarrow A$ in P , with selected atoms $A = A_0, A_1, A_2, \dots$, applied, renamed clauses $H^i \leftarrow B_1^i, \dots, B_{n_i}^i$ ($i = 1, 2, \dots$), and applied mgu's $\theta_1, \theta_2, \dots$. We say that $\tilde{A}_k$ is a direct descendant of $\tilde{A}_i$, if $k > i$ and A_k is the atom $B_j^{i+1} \theta_{i+1} \dots \theta_k$ ($1 \leq j \leq n_{i+1}$).

Definition A.2 (directed subsequence of an LD-derivation) Let P be a program and $\tilde{A} \in B_P^E$. Let $D : \leftarrow A = G_0, G_1, \dots$ be an LD-derivation of $\leftarrow A$ in P . A subsequence $G_{i_0}, G_{i_1}, \dots$, with $G_{i_j} = \leftarrow A_{i_j}, B_{i_j}$, is called a directed subsequence of D iff for all $j \geq 0$, $\tilde{A}_{i_{j+1}}$ is a direct descendant of $\tilde{A}_{i_j}$ in the given LD-derivation.

Definition A.3 (call graph associated to S) Let P be a program and $S \subseteq B_P^E$. The call graph $Call-Gr(P, S)$ associated to P and S is a graph such that:

- its set of nodes is $Call(P, S)$,
- there exists a directed arc from $\tilde{A}$ to $\tilde{B}$ iff $\tilde{B}$ is a direct descendant of $\tilde{A}$.

In [2, Proposition A.1], it was noted that the notion of a call graph (Definition A.3) has a particularly interesting property, which is useful in the study of termination. We recall this proposition together with its proof.

Proposition A.1 Let P be a program, $S \subseteq B_P^E$, $Call(P, S)$ and $Call-Gr(P, S)$ be the call set and call graph associated to S . Let p be any directed path in $Call-Gr(P, S)$. Then there exists an LG-derivation for some element of $Call(P, S)$, such that all the nodes in p occur as selected atoms in the derivation.

Proof By Definition A.3 of $Call-Gr(P, S)$, for every arc from $\tilde{A}$ to $\tilde{B}$ in $Call-Gr(P, S)$, there exists a sequence of LG-resolution steps, starting from $\leftarrow A$ and having a variant of B as its selected atom at the end. Since (a variant of) B is selected at the end-point, any two such derivation-step sequences, corresponding to consecutive arcs in $Call-Gr(P, S)$, can be composed to form a new sequence of LG-derivation steps. In this sequence, all 3 nodes of the consecutive arcs remain selected atoms in the new sequence of derivation steps. Repeatedly applying the above argument yields the result. $\square$

Note that by definition of $Call-Gr(P, S)$ and $Call(P, S)$, the above proposition also implies that, for any directed path p in $Call-Gr(P, S)$, there is a $\tilde{A} \in S$ and a sequence of derivation steps starting from $P \cup \{\leftarrow A\}$, such that all nodes in the given path p are selected atoms in the derivation sequence.

In the proof of Theorem 4.1, we also need the following lemma.

Lemma A.1 Let P be a program and $\tilde{A} \in B_P^E$. Let $\mathcal{F}$ be the LG-forest of $P \cup \{\leftarrow A\}$. Suppose that there is an LG-tree in $\mathcal{F}$ which has an infinitely branching node. Then, there is an LG-tree in $\mathcal{F}$ which is infinitely branching in a non-root node, and this non-root node is a goal with leftmost atom $p(t_1, \dots, t_n)$, with

- $p \in Rec_P$, and
- $p(t_1, \dots, t_n)$ is directly descending from the atom $q(s_1, \dots, s_m)$ in the root via a recursive clause $C = q(u_1, \dots, u_m) \leftarrow \dots, p(v_1, \dots, v_n), \dots$, with $p \simeq q$.

Proof Let $\mathcal{F}$ be the LG-forest of $P \cup \{\leftarrow A\}$, and suppose the LG-tree T_0 of $\mathcal{F}_1$ has an infinitely branching node N_1 . Let $p_1(t_1^1, \dots, t_{n_1}^1)$ be the leftmost atom of the goal in N_1 . If $p_1(t_1^1, \dots, t_{n_1}^1)$ already satisfies the two conditions in the lemma, then there is nothing to prove. Suppose this is not the case. Let T_1 be the LG-tree in $\mathcal{F}$ with a variant of $\leftarrow p_1(t_1^1, \dots, t_{n_1}^1)$ in the root. It follows that the tree T_1 has an infinitely branching node. Let N_2 be such an infinitely branching node in T_1 . Let $p_2(t_1^2, \dots, t_{n_2}^2)$ be the leftmost atom of the goal in N_2 . Either $p_1 \simeq p_2$ or $p_1 \sqsupset p_2$. In the first case, we have that $p_2 \in \text{Rec}_P$ and also that the clause used in the partial derivation from $\leftarrow p_1(t_1^1, \dots, t_{n_1}^1)$ to N_2 is recursive. Hence, the two conditions of the lemma are satisfied for the tree T_1 with infinitely branching node N_2 . In the other case, where $p_1 \sqsupset p_2$, we apply the same reasoning on $p_2(t_1^2, \dots, t_{n_2}^2)$. Since the set of predicate symbols Pred_P is finite, we eventually arrive in the first case. This proves the lemma. $\square$

We can now prove Theorem 4.1 of Section 4.

Theorem 4.1 *Let P be a program and $S \subseteq B_P^E$. P is LG-acceptable w.r.t. S iff P is LG-terminating w.r.t. S .*

Proof $\Rightarrow$: Suppose P is LG-acceptable w.r.t. S and a level mapping $|\cdot|$. Let A be an atom such that $\tilde{A} \in S$. Let $\mathcal{F}$ be the LG-forest of $P \cup \{\leftarrow A\}$. We prove that $\mathcal{F}$ consists of a finite number of finite LG-trees.

- The LG-trees in $\mathcal{F}$ are finitely branching.

Suppose this is not the case, i.e. there is an LG-tree in $\mathcal{F}$ which is infinitely branching. Then, by Lemma A.1, there is an LG-tree $\mathcal{T}$ in $\mathcal{F}$ infinitely branching in a non-root node, and this non-root node is a goal with leftmost atom $p(t_1, \dots, t_n)$, $p \in \text{Rec}_P$, which is directly descending from an atom $q(s_1, \dots, s_m)$, with $p \simeq q$, via a recursive clause $C = q(u_1, \dots, u_m) \leftarrow \dots, p(v_1, \dots, v_n), \dots$. Now, consider the LG-forest $\mathcal{F}^a$ of $P^a \cup \{\leftarrow A\}$. Let $\mathcal{T}^a$ be the LG-tree in $\mathcal{F}^a$ corresponding to $\mathcal{T}$. Note that the clause C^a instead of C is used in $\mathcal{T}^a$. Because of this, the atom on the right of $p(t_1, \dots, t_n)$ in the infinitely branching node is $p^a(t_1, \dots, t_n)$. Thus, $\mathcal{F}^a$ consists of a infinite number of LG-trees (there are an infinite number of LG-trees with predicate p^a in the root). But, all these p^a -atoms directly descend from the node $q(s_1, \dots, s_m)$ via the clause C^a in P^a and hence, their value under the level mapping $|\cdot|$ is smaller or equal to $|q(s_1, \dots, s_m)|$. Since $|\cdot|$ is finitely partitioning on $\text{Call}(P^a, S) \cap B_{\text{Rec}_P^a}^E$, this gives a contradiction.

- $\mathcal{F}$ consists of a finite number of LG-trees, i.e. $\# \text{Call}(P, \{A\}) < \infty$.

Suppose this is not the case. A first possible reason for an infinite number of LG-trees in $\mathcal{F}$ is an infinitely branching LG-tree in $\mathcal{F}$. But we already proved that this does not occur. The other possibility is that there exists an infinite LD-derivation of $\leftarrow A$ in P which contains an infinite directed subsequence, such that this infinite directed subsequence has a tail $G_n, G_{n+1}, \dots$ with $G_i = \leftarrow A_i, \mathbf{B}_i$, $i \geq n$, such that $\{A_i \mid i \geq n\}$ is an infinite set and $\text{Rel}(A_i) \simeq \text{Rel}(A_{i+1})$ for all $i \geq n$. Since $A_i \in \text{Call}(P, S) \cap B_{\text{Rec}_P}^E \subseteq \text{Call}(P^a, S) \cap B_{\text{Rec}_P \cup \text{Rec}_P^a}^E$, and since $|\cdot|$ is finitely partitioning on this set and $|A_i| \geq |A_{i+1}|$ for all $i \geq n$ (by the LG-acceptability condition), this gives a contradiction.

$\Leftarrow$: Suppose P is LG-terminating w.r.t. S . We need to design a level mapping $|\cdot|$, which is finitely partitioning on $Call(P^a, S) \cap B_{Rec_P \cup Rec_P^a}^E$ and such that P is LG-acceptable w.r.t. S and $|\cdot|$. The proof of this part is based on a similar proof in [2, Proof of Theorem 3.1].

In order to define $|\cdot|$ on $Call(P^a, S)$, consider the $Call-Gr(P^a, S)$ -graph.

First, note that this graph is finitely branching. To see this, suppose this is not so. Then, there is an atom in $Call(P, S) \cap B_{Rec_P}^E$ which has infinitely many answers. This contradicts LG-termination of P w.r.t. S .

Another crucial point in this part of the proof is that the strongly connected components of $Call-Gr(P^a, S)$ are necessarily all finite. To see this, suppose this is not the case. Then there is an infinitely long path p , starting from an element in S , through elements of this infinite strongly connected component. Note that this infinitely long path p contains only atoms from B_P^E (since atoms from $B_{Rec_P}^E$ don't depend on any other atom). By Proposition A.1, this means that there exists a derivation for that element in S , such that an infinite number of different atoms from $Call(P^a, S) \cap B_P^E = Call(P, S)$ are selected. Obviously, this contradicts LG-termination.

So, all strongly connected components of $Call-Gr(P^a, S)$ are finite. Define $Call-Gr(P^a, S)/reduced$ as the graph obtained from $Call-Gr(P^a, S)$ by replacing any strongly connected component by a single contracting node and replacing any arc from $Call-Gr(P^a, S)$ pointing to (resp. from) any node in that strongly connected component by an arc to (resp. from) that contracting node. $Call-Gr(P^a, S)/reduced$ does not have any (non-trivial) strongly connected components. Moreover, any strongly connected component from $Call-Gr(P^a, S)$ that was collapsed into a contracting node of $Call-Gr(P^a, S)/reduced$ necessarily consists of only a finite number of nodes.

Note now that each path in $Call-Gr(P^a, S)/reduced$ which is not cyclic (note that there are only trivial cycles in $Call-Gr(P^a, S)/reduced$) is finite. This also follows from Proposition A.1.

So, we can consider the layers of $Call-Gr(P^a, S)/reduced$, and there are only a finite number of layers. We define $|\cdot|$ as follows. Let layer-0 be the set of leaves in $Call-Gr(P^a, S)/reduced$. We assign to these nodes a number within $\mathbb{N}$ such that no infinite set of atoms from this layer is assigned the same number.

Then, we move up to the next layer in $Call-Gr(P^a, S)/reduced$. This layer, layer-1, consists of all nodes $Node$ such that $\max(\{length(p) \mid p \text{ is a path starting from } Node \text{ in } Call-Gr(P^a, S)/reduced\}) = 1$. To any element of layer-1, we assign a natural number n , such that n is larger than all the natural numbers assigned to descendants of the element (note that the graph is finitely branching).

We continue this process layer by layer. The value of the level mapping $|\cdot|$ on elements of $Call(P^a, S)$ is defined as follows: all calls in the same strongly connected components of $Call(P^a, S)$ receive the number assigned to their representative in $Call-Gr(P^a, S)/reduced$.

We prove that P is LG-acceptable w.r.t. S and this level mapping $|\cdot|$.

- $|\cdot|$ is finitely partitioning on $Call(P^a, S) \cap B_{Rec_P \cup Rec_P^a}^E$.

We even have that $|\cdot|$ is finitely partitioning on $Call(P^a, S)$. This is because each strongly connected component of $Call(P^a, S)$ is finite and because of the construction of $|\cdot|$ (there are only a finite number of layers in $Call-Gr(P^a, S)/reduced$ and no infinite set of nodes in the same layer is assigned the same natural number).

- Let A be an atom such that $\tilde{A} \in Call(P, S)$, let $H \leftarrow B_1, \dots, B_n$ be

a clause in P^a , such that $\text{mgu}(A, H) = \theta$ exists, let B_i be such that $\text{Rel}(B_i) \simeq \text{Rel}(A)$ or $\text{Rel}(B_i) \in \text{Rec}_P^a$, let θ_{i-1} be an LD-computed answer substitution for $\leftarrow (B_1, \dots, B_{i-1})\theta$, then we have to prove that $|A| \geq |B_i\theta\theta_{i-1}|$. This follows immediately because there is a directed arc from A to $B_i\theta\theta_{i-1}$ in $\text{Call-Gr}(P^a, S)$ and because of the construction of $|\cdot|$.

□

B The Class of Simply-Moded Well-Typed Queries and Programs, with Input-Safe Heads

In Appendix B.1, we give a proof of Proposition 2.1. Next, in Appendix B.2, we first recall some definitions and properties concerning substitutions and derivations, in particular, we recall the Lifting Theorem, then, we prove Theorem 5.1.

We introduce the following notation. For an atom A , we denote by $\text{Var}(A)$ the set of variables occurring in A . By $\text{InVar}(A)$ we denote the set of variables occurring in input positions of A and by $\text{OutVar}(A)$ the set of variables occurring in output positions of A . Similar notation is used for sequences of atoms.

B.1 Proof of Proposition 2.1

The following lemma was proven in [1, Lemma 17].

Lemma B.1 *Consider two variable-disjoint atoms $A = p(\mathbf{u}, \mathbf{v})$ and $H = p(\mathbf{s}, \mathbf{t})$ with the same predicate symbol p . Suppose that*

- *A is correctly typed in its input positions and simply-moded,*
- *H is input-safe.*

If there is an mgu ϕ for the set of equations $\{\mathbf{u} = \mathbf{s}, \mathbf{v} = \mathbf{t}\}$, then an mgu $\theta = \theta_1\theta_2$ exists with

- *θ_1 is an mgu for $\{\mathbf{u} = \mathbf{s}\}$,*
 $\text{Dom}(\theta_1) \subseteq \text{InVar}(H) = \text{Var}(\mathbf{s}),$
 $\text{Ran}(\theta_1) \subseteq \text{InVar}(A) = \text{Var}(\mathbf{u}),$
- *θ_2 is an mgu for $\{\mathbf{v} = \mathbf{t}\theta_1\}$,*
 $\text{Dom}(\theta_2) \subseteq \text{OutVar}(A) = \text{Var}(\mathbf{v}),$
 $\text{Ran}(\theta_2) \subseteq \text{Var}(\mathbf{t}\theta_1) \subseteq \text{OutVar}(H) \cup \text{InVar}(A) = \text{Var}(\mathbf{t}) \cup \text{Var}(\mathbf{u}),$
- *$A\theta = p(\mathbf{u}, \mathbf{v})\theta = p(\mathbf{u}, \mathbf{v}\theta_2) = p(\mathbf{s}\theta_1, \mathbf{t}\theta_1) = p(\mathbf{s}, \mathbf{t})\theta = H\theta,$*

and θ is equal to ϕ modulo variable renaming.

The fact that in the previous lemma, θ is equal to ϕ modulo variable renaming follows from the following lemma proven in [4]: *if θ and ϕ are two mgu's of a set of equations, then for some renaming η , $\phi = \theta\eta$.*

In the rest of this Appendix B, we will refer to an mgu θ (of a simply-moded query which is correctly typed in its input positions and an input-safe head atom) of the form of Lemma B.1 as a *double-matcher*.

Now we can prove Proposition 2.1 of Subsection 2.3.

Proposition 2.1 *Let P be a simply-moded well-typed program and let $\leftarrow B_1, \dots, B_n$ be a simply-moded well-typed goal with $n \geq 1$. Suppose that the heads of the clauses of P are input-safe.*

If $\sigma = \theta^1 \dots \theta^k$ is the substitution of a (partial) LD-derivation of $\leftarrow B_1, \dots, B_n$ in P , then the input arguments of B_1 are equal to the input arguments of $B_1\sigma$ modulo variable renaming.

Proof Recall that the notions of simply-modedness and well-typedness are persistent (Lemma 2.3 and 2.4). Since two *mgu*'s of the same set of equations are equal modulo renaming [4], it is sufficient to prove the proposition in the case that each θ^i , $i = 1, \dots, k$, is a double-matcher (note that these exist, since the conditions of Lemma B.1 are satisfied in each step). We prove that in this case, the input arguments of B_1 are *equal* to the input arguments of $B_1\sigma = B_1\theta^1 \dots \theta^k$, i.e. no substitution θ^i , $i = 1, \dots, k$, is ever applied on variables of $InVar(B_1)$. For this, it is sufficient to prove the following two statements for every $i = 0, \dots, k - 1$:

- the variables of $InVar(B_1)$ do not occur in the output positions of the i 'th goal in the partial LD-derivation of $\leftarrow B_1, \dots, B_n$ in P ,
- in the $(i + 1)$ 'th step of the partial LD-derivation of $\leftarrow B_1, \dots, B_n$ in P , the domain of the *mgu* θ^{i+1} does not include variables of $InVar(B_1)$.

We will prove these two statements by induction on the length of the partial LD-derivation.

If $i = 0$, the first statement follows from the fact that the goal $\leftarrow B_1, \dots, B_n$ is simply-moded. The second statement follows from Lemma B.1.

Suppose $i > 0$ and suppose that the two statements are true for $i - 1$. That is, variables of $InVar(B_1)$ do not occur in the output positions of the $(i - 1)$ 'th goal and the *mgu* θ^i is not applied on variables of $InVar(B_1)$. We prove the two statements for i .

- We prove that the variables of $InVar(B_1)$ do not occur in the output positions of the resulting i 'th goal. Let $\leftarrow C_1, \dots, C_m$ be the $(i - 1)$ 'th goal and $H \leftarrow B'_1, \dots, B'_r$ be the standardized apart program clause with which it is resolved. So, $\theta^i = mgu(C_1, H)$ and the resulting i 'th goal is $\leftarrow (B'_1, \dots, B'_r, C_2, \dots, C_m)\theta^i$. Since we assumed that θ^i is a double-matcher, we can write $\theta^i = \theta_1^i \theta_2^i$ (see Lemma B.1), and so the i 'th goal is equal to $\leftarrow (B'_1, \dots, B'_r)\theta_1^i, (C_2, \dots, C_m)\theta_2^i$. From Lemma B.1 we know that $Dom(\theta_1^i) \subseteq InVar(H)$. From the fact that $H \leftarrow B'_1, \dots, B'_r$ is simply-moded (more specifically from the fact that output positions in the body don't share variables with the input positions of the head), we can then conclude that $(B'_1, \dots, B'_r)\theta_1^i$ does not contain variables of $InVar(B_1)$ in the output positions. From Lemma B.1 we also know that $Dom(\theta_2^i) \subseteq OutVar(C_1)$. From the induction hypothesis, we know that $\leftarrow C_1, \dots, C_m$ does not contain variables of $InVar(B_1)$ in the output positions. Because $\leftarrow C_1, \dots, C_m$ is simply-moded, and more specifically because the output arguments form a linear family of variables, we hence have that $(C_2, \dots, C_m)\theta_2^i$ does not contain variables of $InVar(B_1)$ in the output positions.
- Next we prove that the domain of the *mgu* of the $(i + 1)$ 'th step, θ^{i+1} , does not include variables of $InVar(B_1)$. This follows from Lemma B.1: the domain of θ^{i+1} is included in the set of input variables of the head of the

clause used in the $(i + 1)$ 'th step union the set of output variables of $B_1'\theta^i$ (which, as we already proved, doesn't contain variables of $InVar(B_1)$).

□

B.2 Proof of Theorem 5.1

First, we recall the necessary concepts concerning substitutions and derivations. In particular, we recall the Lifting Theorem. Then, we prove Theorem 5.1, which states that for simply-moded well-typed programs and queries such that the heads of the program clauses are input-safe, quasi-termination and LG-termination are closed under substitution.

B.2.1 Preliminaries: Substitutions, Derivations and the Lifting Theorem

We recall some notations and properties concerning substitutions and derivations, which we will need in the next subsection. We recall the Lifting Theorem. We closely follow [3].

A *substitution* is a function θ mapping variables to terms such that its domain $Dom(\theta) = \{X \mid X\theta \neq X\}$ is finite. The notation $\{X_1/t_1, \dots, X_n/t_n\}$, where $X_1, \dots, X_n$ are pairwise different variables, for θ is used in case $Dom(\theta) = \{X_1, \dots, X_n\}$ and for $1 \leq i \leq n$, θ assigns the term t_i to the variable X_i . If $\theta = \{X_1/t_1, \dots, X_n/t_n\}$, then $Ran(\theta) = Var(t_1, \dots, t_n)$ and $Var(\theta) = Dom(\theta) \cup Ran(\theta)$. By ϵ we denote the identity substitution (which has an empty domain). If V is a set of variables, then $\theta|_V := \{X/X\theta \mid X \in V \cap Dom(\theta)\}$ is the *restriction* of θ to V . If θ and λ are substitutions, then the substitution $\theta\lambda$, the *composition* of θ and λ , is defined by: $X(\theta\lambda) = (X\theta)\lambda$. Substitutions θ and λ are considered *equal*, $\theta = \lambda$, iff $Dom(\theta) = Dom(\lambda)$ and for every $X \in Dom(\theta)$: $X\theta = X\lambda$, or equivalently, iff for every variable X : $X\theta = X\lambda$. Associativity holds for substitutions, that is, for substitutions θ, λ, σ : $(\theta\lambda)\sigma = \theta(\lambda\sigma)$. A substitution ρ is called a *renaming* for an expression E iff ρ is injective on $Var(E)$ and maps variables to variables. $E\rho$ is called a *variant* of E . The substitution θ is *more general than* the substitution λ if for some σ we have that $\lambda = \theta\sigma$. The substitution θ *unifies* or is a *unifier* of the expressions A and B iff $A\theta = B\theta$. A unifier θ for (sequences of) expressions is called *most general* or a *most general unifier (mgu)* iff it is more general than any other unifier of these (sequences of) expressions. A substitution θ is called *idempotent* iff $\theta\theta = \theta$. A substitution θ is idempotent iff $Dom(\theta) \cap Ran(\theta) = \emptyset$. A unifier θ of E is called *relevant* iff $Var(\theta) \subseteq Var(E)$.

The *resultant* of a finite sequence of SLD-resolution steps $D : (\leftarrow Q_0) \rightarrow^{\theta^1} (\leftarrow Q_1) \rightarrow^{\theta^2} \dots \rightarrow^{\theta^n} (\leftarrow Q_n)$ is the implication $res(D) = Q_n \rightarrow Q_0\theta^1 \dots \theta^k$. An SLD-resolution step $(\leftarrow Q_1) \rightarrow^\theta (\leftarrow Q_2)$ *releases* the variable X iff $X \in Var(Q_1\theta) \setminus Var(Q_2)$. A variable is *released* by a sequence of SLD-resolution steps if it is released by one of the steps. Recall from [3] that a sequence of SLD-resolution steps is called an *SLD-derivation* iff no variable released at some step occurs in the resultant associated with a later step. We will need the following lemma [3].

Lemma B.2 *If $D : (\leftarrow Q_0) \rightarrow^{\theta^1} (\leftarrow Q_1) \rightarrow^{\theta^2} \dots \rightarrow^{\theta^n} (\leftarrow Q_n)$ is an SLD-derivation and $X \in Var(Q_0\theta^1 \dots \theta^n) \setminus Var(Q_n)$, then X is released by D .*

A finite SLD-derivation $D : (\leftarrow Q_0) \rightarrow^{\theta^1} (\leftarrow Q_1) \rightarrow^{\theta^2} \dots \rightarrow^{\theta^n} \square$ ending in the empty query $\square$ is called *successful*. Recall that in that case, the restriction $\theta :=$

$(\theta^1 \dots \theta^n)_{|Var(Q_0)}$ of the composition $\theta^1 \dots \theta^n$ to the variables of the initial query Q_0 is called a *(SLD-)computed answer substitution* for Q_0 .

If $D : G_0 \rightarrow^{\theta^1} G_1 \rightarrow^{\theta^2} \dots \rightarrow^{\theta^n} G_n \rightarrow^{\theta^{n+1}} \dots$ is a (partial) SLD-derivation and $i < j$, then $D_{i,j}$ denotes the subderivation $G_i \rightarrow^{\theta^{i+1}} \dots \rightarrow^{\theta^j} G_j$ which derives G_j from G_i and $\theta^{i,j}$ stands for the composition $\theta^{i+1} \dots \theta^j$ of the *mgu*'s in $D_{i,j}$.

A (partial) SLD-derivation D is called a *lift* of a given SLD-derivation D' iff D and D' have the same length, the initial goal of D' is an instance of the initial goal of D and, at corresponding places, D and D' apply the same rule to selected atoms in the same position.

We recall an important theorem (from [3, Lifting Theorem 5.37]) which states that: (i) lifts always exist and (ii) a lift of an SLD-derivation D' is related to D' by means of a sequence of substitutions satisfying instantiation and commutation properties.

Theorem B.1 (Lifting) *Let P be a program and $\leftarrow Q$ a goal. Suppose that $\leftarrow Q' = \leftarrow Q\sigma$, with $Dom(\sigma) \subseteq Var(Q)$.*

1. *Every (finite or infinite) SLD-derivation starting from $\leftarrow Q'$ has a lift starting from $\leftarrow Q$.*
2. *Let $D' : G'_0 = \leftarrow Q' \rightarrow^{\psi^1} G'_1 \rightarrow^{\psi^2} G'_2 \rightarrow^{\psi^3} \dots$ be an SLD-derivation of $\leftarrow Q'$. If the (partial) SLD-derivation $D : G_0 = \leftarrow Q \rightarrow^{\theta^1} G_1 \rightarrow^{\theta^2} G_2 \rightarrow^{\theta^3} \dots$ lifts D' , then there exists a sequence $\sigma^0, \sigma^1, \sigma^2, \dots$ of substitutions such that for $i < j$:*

- (a) $\sigma = \sigma^0$,
- (b) $G'_j = G_j\sigma^j$ (“instantiation”),
- (c) $(\sigma^i\psi^{i,j})_{|Var(G_i)} = (\theta^{i,j}\sigma^j)_{|Var(G_i)}$ (“commutation”),
- (d) $res(D'_{i,j}) = res(D_{i,j})\sigma^j$.

With *similar* SLD-derivations, we mean SLD-derivations of the same length starting with the same goal and, at corresponding places, applying the same rule to selected atoms in the same position.

Corollary B.1 *Let P be a program and $\leftarrow Q$ a goal. Let $D : G_0 = \leftarrow Q \rightarrow^{\theta^1} G_1 \rightarrow^{\theta^2} G_2 \rightarrow^{\theta^3} \dots$ and $E : G'_0 = \leftarrow Q \rightarrow^{\psi^1} G'_1 \rightarrow^{\psi^2} G'_2 \rightarrow^{\psi^3} \dots$ be similar SLD-derivations of $\leftarrow Q$ in P . Then, there exists a sequence of renamings $\rho^0, \rho^1, \rho^2, \dots$ such that for $i < j$:*

1. $\epsilon = \rho^0$,
2. $G'_j = G_j\rho^j$ (“variance”),
3. $(\rho^i\psi^{i,j})_{|Var(G_i)} = (\theta^{i,j}\rho^j)_{|Var(G_i)}$ (“commutation”),
4. $res(E_{i,j}) = res(D_{i,j})\rho^j$.

Proof Note first that two derivations which are similar are lifts of each other. So, the corollary follows directly by applying the Lifting Theorem B.1 in two directions (for every j , G_j and G'_j are instances of each other and hence they are renamings of each other, so the sequence of substitutions $\sigma^0, \sigma^1, \sigma^2, \dots$ from Theorem B.1 is a sequence of renamings). $\square$

B.2.2 Substitution-Closedness of Termination under a Tabled Execution for the Class of Simply-Moded Well-Typed Queries and Programs with Input-Safe Heads

We restrict ourselves to LD-derivations, i.e. SLD-derivations with the left-to-right selection rule. In the sequel, we consider the special case of a simply-moded well-typed program P and query Q such that the heads of the clauses in P are input-safe. The parameter passing mechanism in the unification of a simply-moded query A , which is correctly typed in its input positions, with an input-safe head atom H was described in Lemma B.1: first the input values are passed from the atom A to the head H , then the output values are passed from H to A . Recall that an *mgu* θ of the form of Lemma B.1 is called a *double-matcher*. Note that a double-matcher is idempotent and relevant. Note also that, given atoms A and H satisfying the conditions of Lemma B.1, there is a unique double-matcher θ for A and H .

Lemma B.1 is useful in the context of a simply-moded well-typed program P and query Q , such that the heads of the clauses of P are input-safe. Indeed, in that case, in every step of an LD-derivation of $\leftarrow Q$ in P , the conditions of the lemma are satisfied (recall from Lemma 2.3 and 2.4 that the notions simply-modedness and well-typedness are persistent). Hence, we have that for every LD-derivation of $\leftarrow Q$ in P , there exists a similar LD-derivation of $\leftarrow Q$ in P using double-matchers as *mgu*'s. We next describe some interesting properties of LD-derivations using double-matchers of a simply-moded well-typed query in a simply-moded well-typed program such that the heads of the clauses are input-safe. We have the following lemma.

Lemma B.3 *Let $\leftarrow Q = \leftarrow B_1, \dots, B_n$ be a simply-moded well-typed goal and P be a simply-moded well-typed program such that the heads of the clauses of P are input-safe. Let D be a (partial) LD-derivation of $\leftarrow Q$ in P using double-matchers $\theta^1, \theta^2, \dots, \theta^k$.*

Then, the input arguments of B_1 are equal to these of $B_1\theta^1 \dots \theta^k$.

Also, for all $i \geq 0$, the variables of $\text{InVar}(B_1)$ do not occur in the output positions of the i 'th goal in D and, in the $(i+1)$ 'th step in D , the domain of the mgu (double-matcher) θ^{i+1} does not include variables of $\text{InVar}(B_1)$: $\text{Dom}(\theta^{i+1}) \cap \text{InVar}(B_1) = \emptyset$.

Proof The proof follows directly from the proof of Proposition 2.1 in Appendix B.1. $\square$

We next consider a particular case of the Lifting Theorem B.1. Namely, we consider the case of a simply-moded well-typed program P and goal $\leftarrow Q$, such that the heads of the clauses of P are input-safe, and we require that the substitution σ is such that $\leftarrow Q\sigma$ is again simply-moded well-typed. In that case, we show that for LD-derivations using double-matchers, the substitutions σ^i of Theorem B.1 are all equal to the initial substitution σ .

Proposition B.1 *Let $\leftarrow Q = \leftarrow q(\mathbf{x}, \mathbf{y})$ be a simply-moded well-typed goal and P be a simply-moded well-typed program such that the heads of the clauses of P are input-safe. Let σ be a substitution with $\text{Dom}(\sigma) \subseteq \text{InVar}(Q) = \text{Var}(\mathbf{x})$, and $\text{Ran}(\sigma) \cap \text{Var}(Q) = \emptyset$, such that $\leftarrow Q' = \leftarrow Q\sigma = \leftarrow q(\mathbf{x}\sigma, \mathbf{y})$ is again simply-moded and well-typed. Let $D' : G_0 = \leftarrow Q' \rightarrow^{\psi^1} G_1' \rightarrow^{\psi^2} G_2' \rightarrow^{\psi^3} \dots$ be an LD-derivation of $\leftarrow Q'$ in P using double-matchers. Then there exists a (partial) LD-derivation $D : G_0 = \leftarrow Q \rightarrow^{\theta^1} G_1 \rightarrow^{\theta^2} G_2 \rightarrow^{\theta^3} \dots$ of $\leftarrow Q$ in P using double-matchers, which is a lift of D' such that for $i < j$:*

1. $G'_j = G_j \sigma$,
2. $(\sigma \psi^{i,j})|_{Var(G_i)} = (\theta^{i,j} \sigma)|_{Var(G_i)}$,
3. $res(D'_{i,j}) = res(D_{i,j}) \sigma$.

Proof Let D' be an LD-derivation $G'_0 = \leftarrow Q' \rightarrow^{\psi^1} G'_1 \rightarrow^{\psi^2} G'_2 \rightarrow^{\psi^3} \dots$ of $\leftarrow Q'$ in P using double-matchers $\psi^1, \psi^2, \psi^3, \dots$. By the Lifting Theorem B.1, there exists a lift D of D' starting from $\leftarrow Q$. Since there exists a similar derivation to D which uses double-matchers and which is hence also a lift of D' , we may assume that D uses double-matchers. We will also assume that, in all corresponding resolution steps of both LD-derivations, the clause of P used in that resolution step, is standardized apart using one and the same renaming. This means in particular that no variable of $Ran(\sigma)$ is introduced in a resolution step of D . Let $D : G_0 = \leftarrow Q \rightarrow^{\theta^1} G_1 \rightarrow^{\theta^2} G_2 \rightarrow^{\theta^3} \dots$.

We first prove point 1 by induction on n . Note that point 1 holds for $n = 0$: $G'_0 = \leftarrow Q' = \leftarrow Q \sigma = G_0 \sigma$. Suppose $G'_j = G_j \sigma$ holds for $j \leq n$. We prove that $G'_{n+1} = G_{n+1} \sigma$. Let $G_n = \leftarrow A, \mathbf{A}$, then $G'_n = \leftarrow A \sigma, \mathbf{A} \sigma$. Let $C = H \leftarrow \mathbf{B}$ be the (standardized apart) clause used in the $(n+1)$ 'th resolution step of D and D' . So, θ^{n+1} is the double-matcher which unifies A and H . Let $A = p(\mathbf{u}, \mathbf{v})$ and $H = p(\mathbf{s}, \mathbf{t})$. Then, $\theta^{n+1} = \theta_1^{n+1} \theta_2^{n+1}$ with θ_1^{n+1} the *mg*u for $\{\mathbf{u} = \mathbf{s}\}$ and θ_2^{n+1} the *mg*u for $\{\mathbf{v} = \mathbf{t} \theta_1^{n+1}\}$. Consider the double-matcher $\psi^{n+1} = \psi_1^{n+1} \psi_2^{n+1}$ which unifies $A \sigma$ and H . Since $Dom(\sigma) \subseteq InVar(Q)$ and because the input variables of Q do not occur in output positions of A (Proposition B.3), $A \sigma = p(\mathbf{u} \sigma, \mathbf{v})$. So, ψ_1^{n+1} is the *mg*u for $\{\mathbf{u} \sigma = \mathbf{s}\}$ and ψ_2^{n+1} is the *mg*u for $\{\mathbf{v} = \mathbf{t} \psi_1^{n+1}\}$. Inspecting the set $\{\mathbf{u} \sigma = \mathbf{s}\}$, it follows by Lemma B.1 that, for all $Y \notin Dom(\sigma)$: $Y \psi_1^{n+1} = Y \theta_1^{n+1} \sigma$. Also, inspecting the set $\{\mathbf{v} = \mathbf{t} \psi_1^{n+1}\}$ which is then equal to the set $\{\mathbf{v} = \mathbf{t} \theta_1^{n+1} \sigma\}$, it follows that, for all $Y \notin Dom(\sigma)$: $Y \psi_2^{n+1} = Y \theta_2^{n+1} \sigma$. Consider next the LD-resolvent G'_{n+1} of G'_n and C using $\psi^{n+1} = \psi_1^{n+1} \psi_2^{n+1}$:

$$\begin{aligned} G'_{n+1} &= \leftarrow \mathbf{B} \psi_1^{n+1}, \mathbf{A} \sigma \psi_2^{n+1} \\ &= \leftarrow \mathbf{B} \theta_1^{n+1} \sigma, \mathbf{A} \sigma \theta_2^{n+1} \sigma. \end{aligned}$$

Indeed, this follows from the fact that $Var(\mathbf{B}) \cap Dom(\sigma) = \emptyset$ and $Var(\mathbf{A} \sigma) \cap Dom(\sigma) = \emptyset$. We will prove that $\mathbf{A} \sigma \theta_2^{n+1} \sigma = \mathbf{A} \theta_2^{n+1} \sigma$, from which we then can conclude that

$$\begin{aligned} G'_{n+1} &= \leftarrow \mathbf{B} \theta_1^{n+1} \sigma, \mathbf{A} \theta_2^{n+1} \sigma \\ &= G_{n+1} \sigma. \end{aligned}$$

We prove that for all $X \in Var(\mathbf{A})$, $X \sigma \theta_2^{n+1} \sigma = X \theta_2^{n+1} \sigma$.

Let $X \in OutVar(\mathbf{A})$, then $X \sigma = X$ (by Proposition B.3 and the fact that $Dom(\sigma) \subseteq InVar(Q)$), hence $X \sigma \theta_2^{n+1} \sigma = X \theta_2^{n+1} \sigma$.

Next, suppose that $X \in InVar(\mathbf{A})$. We consider two cases: $X \notin Dom(\sigma)$ and $X \in Dom(\sigma)$.

If $X \notin Dom(\sigma)$, then $X \sigma = X$ and hence $X \sigma \theta_2^{n+1} \sigma = X \theta_2^{n+1} \sigma$.

If $X \in Dom(\sigma)$, then $Var(X \sigma) \subseteq Ran(\sigma)$. By our assumption on the derivation D (i.e., no variable of $Ran(\sigma)$ is introduced in a resolution step of D), $Ran(\sigma) \cap Dom(\theta_2^{n+1}) = \emptyset$, hence, $X \sigma \theta_2^{n+1} \sigma = X \sigma \sigma = X \sigma$ (the last equality follows from the fact that σ is assumed to be idempotent, $Dom(\sigma) \cap Ran(\sigma) = \emptyset$).

By Proposition B.3 and by the fact that $X \in \text{Dom}(\sigma) \subseteq \text{InVar}(Q)$, we have that $X \notin \text{Dom}(\theta_2^{n+1})$. So, $X\theta_2^{n+1}\sigma = X\sigma$. Together, we obtain that $X\sigma\theta_2^{n+1}\sigma = X\sigma = X\theta_2^{n+1}\sigma$.

We next prove point 2. Note first that from above it follows that, for all $n \in \mathbb{N}$:

$$\begin{aligned} G_n\sigma\psi^{n+1} &= \leftarrow A\sigma\psi^{n+1}, \mathbf{A}\sigma\psi^{n+1} \\ &= \leftarrow p(\mathbf{u}\sigma, \mathbf{v}\psi_2^{n+1}), \mathbf{A}\sigma\psi_2^{n+1} \\ &= \leftarrow p(\mathbf{u}\sigma, \mathbf{v}\theta_2^{n+1}\sigma), \mathbf{A}\sigma\theta_2^{n+1}\sigma \\ &= \leftarrow p(\mathbf{u}, \mathbf{v}\theta_2^{n+1})\sigma, \mathbf{A}\theta_2^{n+1}\sigma \\ &= \leftarrow A\theta^{n+1}\sigma, \mathbf{A}\theta^{n+1}\sigma \\ &= G_n\theta^{n+1}\sigma \quad (*). \end{aligned}$$

The third equality holds because $\text{Var}(\mathbf{v}) \cap \text{Dom}(\sigma) = \emptyset$ and $\text{Var}(\mathbf{A}\sigma) \cap \text{Dom}(\sigma) = \emptyset$. Hence, $(\sigma\psi^{n+1})|_{\text{Var}(G_n)} = (\theta^{n+1}\sigma)|_{\text{Var}(G_n)}$, for all $n \in \mathbb{N}$. We show by induction on n that, for $m < n$, $G_m\sigma\psi^{m,n} = G_m\theta^{m,n}\sigma$ (which proves point 2). Note that this holds for $n = 1$ (and $m = 0$), since, by the equality (*) above, $\leftarrow Q\sigma\psi^{0,1} = \leftarrow Q\sigma\psi^1 = \leftarrow Q\theta^1\sigma = \leftarrow Q_0\theta^{0,1}\sigma$. Suppose that it holds for n , then we prove that for $m < n+1$, $G_m\sigma\psi^{m,n+1} = G_m\theta^{m,n+1}\sigma$. For $m = n$, this follows by the equality (*) above. So, let $m < n$. Let $Y \in \text{Var}(G_m)$ and suppose that $Y\theta^{m,n} = t$. By Lemma B.2, every variable in t either occurs in G_n or is released along $D_{m,n}$.

Suppose $X \in \text{Var}(t)$ and $X \in G_n$.

Then, because $(\theta^{n+1}\sigma)|_{\text{Var}(G_n)} = (\sigma\psi^{n+1})|_{\text{Var}(G_n)}$, we have that $X\theta^{n+1}\sigma = X\sigma\psi^{n+1}$.

Suppose $X \in \text{Var}(t)$ and X is released along $D_{m,n}$.

Then, we prove that $X\theta^{n+1}\sigma = X\sigma\psi^{n+1}$ also holds. Note that, since $X \notin G_n$, we have that $X\theta^{n+1} = X$.

If $X \in \text{Dom}(\sigma)$, then the variables in $X\sigma$ occur in input positions of $Q\sigma$ and, by Proposition B.3, $X\sigma$ does not belong to $\text{Dom}(\psi^{n+1})$, hence $X\sigma\psi^{n+1} = X\sigma$ and we can conclude that $X\theta^{n+1}\sigma = X\sigma = X\sigma\psi^{n+1}$.

If $X \notin \text{Dom}(\sigma)$, then $X\sigma = X$. Note that $X \notin G_n' = G_n\sigma$, since $X \notin G_n$ and $X \notin \text{Ran}(\sigma)$ because variables of $\text{Ran}(\sigma)$ do not occur in the derivation D . So, $X\psi^{n+1} = X$ and we can conclude that $X\theta^{n+1}\sigma = X\sigma = X = X\sigma\psi^{n+1}$. Thus,

$$\begin{aligned} Y\theta^{m,n+1}\sigma &= Y\theta^{m+1} \dots \theta^n\theta^{n+1}\sigma \\ &= t\theta^{n+1}\sigma \\ &= t\sigma\psi^{n+1} \\ &= Y\theta^{m,n}\sigma\psi^{n+1} \\ &= Y\sigma\psi^{m,n}\psi^{n+1} \\ &= Y\sigma\psi^{m,n+1}. \end{aligned}$$

The second last step follows from the induction hypothesis and the fact that $m < n$. This proves point 2.

Finally, note that point 3 follows from points 1 and 2. $\square$

Corollary B.2 *Let $\leftarrow Q = \leftarrow q(\mathbf{x}, \mathbf{y})$ be a simply-moded well-typed goal and P be a simply-moded well-typed program such that the heads of the clauses of P are input-safe. Let σ be a substitution with $\text{Dom}(\sigma) \subseteq \text{InVar}(Q) = \text{Var}(\mathbf{x})$, and $\text{Ran}(\sigma) \cap \text{Var}(Q) = \emptyset$, such that $\leftarrow Q\sigma$ is also simply-moded well-typed.*

If ψ is an LD-computed answer substitution of a derivation using double-matchers for $\leftarrow Q\sigma$, then there exists an LD-computed answer θ for $\leftarrow Q$ with $Q\sigma\psi = Q\theta\sigma$.

Proof This follows directly from point 3 of Proposition B.1. $\square$

Finally, we can prove our results on substitution-closedness of termination of a tabled execution, namely that, for the class of simply-moded well-typed programs and queries such that the heads of the program clauses are input-safe, quasi-termination and LG-termination are substitution-closed (Theorem 5.1 of Section 5).

Theorem 5.1 *Let $Q = q(\mathbf{x}, \mathbf{y})$ be a simply-moded well-typed query and P a simply-moded well-typed program such that the heads of the clauses of P are input-safe. Let σ be a substitution on Q such that $Q\sigma$ is also simply-moded, well-typed. If P quasi-terminates w.r.t. $\{Q\}$, then P quasi-terminates w.r.t. $\{Q\sigma\}$. If P LG-terminates w.r.t. $\{Q\}$, then P LG-terminates w.r.t. $\{Q\sigma\}$.*

Proof We first prove the result for quasi-termination. We may assume that $\text{Dom}(\sigma) \subseteq \text{InVar}(Q) = \text{Var}(\mathbf{x})$ and $\text{Ran}(\sigma) \cap \text{Var}(Q) = \emptyset$ (since quasi-termination is closed under renamings). We give a proof by contradiction. Assume that P quasi-terminates w.r.t. $\{Q\}$ that is, the LG-forest of $P \cup \{\leftarrow Q\}$ consists of a finite number of trees without infinite branches. Assume also that P doesn't quasi-terminate w.r.t. $\{Q\sigma\}$. Since LD-termination is closed under substitution, this means that the LG-forest $\mathcal{F}'$ of $P \cup \{\leftarrow Q\sigma\}$ consists of an infinite number of trees without infinite branches. Since $\mathcal{F}'$ consists of an infinite number of trees, there are an infinite number of different, non-variant calls in $\mathcal{F}'$. Or, since the set of predicates is finite, there is a predicate p/n such that $\mathcal{F}'$ contains an infinite number of non-variant calls to p -atoms. We may assume that double-matchers are used in $\mathcal{F}'$. Let's denote this infinite set of non-variant calls as $\{p(\mathbf{s}_1), p(\mathbf{s}_2), \dots, p(\mathbf{s}_i), \dots\}$, with $\mathbf{s}_i$ an n -tuple of terms. Let $\mathcal{F}$ be the LG-forest of $P \cup \{\leftarrow Q\}$, and assume $\mathcal{F}$ uses double-matchers. By Proposition B.1, for all $p(\mathbf{s}_i)$, there is a p -atom $p(\mathbf{t}_i)$ such that $p(\mathbf{t}_i)$ is a selected atom in $\mathcal{F}$ and such that $p(\mathbf{s}_i) = p(\mathbf{t}_i)\sigma$. Since $\mathcal{F}$ consists of a finite number of trees, the set $\{p(\mathbf{t}_1), p(\mathbf{t}_2), \dots, p(\mathbf{t}_i), \dots\}$ is finite up to variable renaming. But, this is in contradiction with the fact that the set $\{p(\mathbf{s}_1), p(\mathbf{s}_2), \dots, p(\mathbf{s}_i), \dots\} = \{p(\mathbf{t}_1)\sigma, p(\mathbf{t}_2)\sigma, \dots, p(\mathbf{t}_i)\sigma, \dots\}$ consists of an infinite number of non-variant atoms.

We next prove that if P LG-terminates w.r.t. $\{Q\}$, then P LG-terminates w.r.t. $\{Q\sigma\}$. Let P^a be the a -transform of P . Put for predicates p^a/n : $m_{p^a}(i) = \text{In}$ and $t_{p^a}(i) = t_p(i)$ for all $i \in \{1, \dots, n\}$. Then, if P is a simply-moded well-typed program with clauses with an input-safe heads, P^a is also simply-moded well-typed and its clauses have also input-safe heads. Recall from Section 4 that P LG-terminates w.r.t. $\{Q\}$ iff P^a quasi-terminates w.r.t. $\{Q\}$. Now, given that P LG-terminates w.r.t. $\{Q\}$, we obtain that P^a quasi-terminates w.r.t. $\{Q\}$. Note that P^a , Q and σ satisfy the conditions of Theorem 5.1. Hence, as we just proved, P^a quasi-terminates w.r.t. $\{Q\sigma\}$. This gives us that P LG-terminates w.r.t. $\{Q\sigma\}$. $\square$

References

- [1] K.R. Apt and S. Etalle. On the unification free Prolog programs. In A. Borzyszkowski and S. Sokolowski, editors, *Proceedings of the Conference on Mathematical Foundations of Computer Science (MFCS'93)*, number 711 in Lecture Notes in Computer Science, pages 1–19. Springer-Verlag, 1993.
- [2] S. Decorte, D. De Schreye, M. Leuschel, B. Martens, and K. Sagonas. Termination Analysis for Tabled Logic Programming. In N. Fuchs, editor, *Proceedings of LOPSTR'97: Logic Program Synthesis and Transformation*, number 1463 in Lecture Notes in Computer Science, pages 107–123. Springer-Verlag, 1997.
- [3] K. Doets. *From Logic to Logic Programming*. MIT Press, 1994.
- [4] J.-L. Lassez, M.J. Maher, and K. Marriott. Unification revisited. In J. Minker, editor, *Foundations of deductive databases and logic programming*, pages 587–625. Morgan-Kaufman, 1988.