CALL-CORRECT SPECIALISATION OF LOGIC PROGRAMS

A. BOSSI
Dipartimento di Matematica e Informatica, via Torino 155,
30173 Venezia, Italy
E-mail: bossi@dsi.unive.it

S. ROSSI
Dipartimento di Matematica, via Belzoni 7,
35131 Padova, Italy
E-mail: sabina@math.unipd.it

In this paper we introduce the concept of *specialisable call correct* program. It is based on the notion of *specialised derivation* which is intended to describe program behaviour whenever some constraints on procedure calls are assumed. Both operational and fixpoint constructions are defined. They characterize successful derivations of programs where only atoms satisfying a given call-condition are selected. We show that specialisable call correct programs can be transformed into *call-correct* ones. A sufficient condition to verify specialisable call correctness is stated.

1 Introduction

In this paper we introduce a novel notion of correctness for logic programs. It characterizes correct programs wrt to a given pre/post specification [?,?,?,?,?,?] where the only request on the call patterns is that they can be instanziated in order to satisfy a given call-condition. Programs satisfying such property are called *specialisable call correct* (*s.c.c.*, in short) and are proved to be specialisable into correct ones where all the call patterns do satisfy the given call-condition. This allows us to reason on type correctness of logic programs without the need of augmenting programs with type declarations (in the form of Prolog procedure) as in [?,?,?]. Abstract interpretation techniques can be used to provide a finite description of the call-condition.
As an example of a useful call-specialisation, consider the following Prolog program computing the frontier of a binary tree [?]:

$\texttt{front}(\texttt{void}, [\,]).$
$\texttt{front}(\texttt{tree}(X, \texttt{void}, \texttt{void}), [X]).$
$\texttt{front}(\texttt{tree}\ (X, L, R), X_s) \leftarrow \texttt{nel\_tree}(\texttt{tree}(X, L, R)),$
$\qquad \texttt{front}(L, L_s),$
$\qquad \texttt{front}(R, R_s),$
$\qquad \texttt{append}(L_s, R_s, X_s).$

```
nel_tree(tree(_, tree (_, _, _), _)).
nel_tree(tree(_, _, tree (_, _, _))).
```

where the relation **nel-tree** is used to enforce a tree to be neither the empty tree nor a leaf $\texttt{tree}(a, \texttt{void}, \texttt{void})$. Observe that the simpler program obtained by removing the atom $\texttt{nel\_tree}(\texttt{tree}(X, L, R))$ in the body of the third clause and by discarding the relation **nel_tree** is indeed incorrect (see [?]).
Suppose that the domain of application consists of the set of trees whose left subtrees are always leafs and consider the following pre/call/post specification:

$$
\begin{aligned}
\mathit{Pre} &= \{\texttt{front}\ (t, l) \mid t \text{ is a term and } l \text{ is a variable}\} \cup \\
&\quad \{\texttt{nel\_list}\ (t) \mid t \text{ is a term}\} \cup \{\texttt{append}\ (u, v, z) \mid u, v, z \text{ are terms}\} \\
\mathit{Call} &= \{\texttt{front}\ (t, l) \mid t \text{ is either the empty tree or a leaf or a term} \\
&\quad\ \text{of the form } \texttt{tree}(u, r, s) \text{ where } r \text{ is a leaf and } u, s \text{ and } l \text{ are terms}\} \cup \\
&\quad \{\texttt{nel\_list}\ (t) \mid t \text{ is a term}\} \cup \{\texttt{append}\ (u, v, z) \mid u, v, z \text{ are terms}\} \\
\mathit{Post} &= \{\texttt{front}\ (t, l) \mid l \text{ is the frontier of the binary tree } t\} \cup \\
&\quad \{\texttt{nel\_list}\ (t) \mid t \text{ is a term}\} \cup \{\texttt{append}\ (u, v, z) \mid u, v, z \text{ are terms}\}.
\end{aligned}
$$

The program is s.c.c. wrt the *Pre*, *Call* and *Post*. In fact, each derivation starting with a query Q satisfying the pre-condition *Pre* where all the call-patterns can be instantiated to an atom satisfying the call-condition *Call*, produces a computed instance $Q\theta$ satisfying the post-condition *Post*. The program can be specialised into a call-correct one as follows:

```
front (void, [ ]).
front (tree (X, void, void), [X]).
front (tree (X, tree (L, void, void), R), Xs) ←
            nel_tree (tree (X, tree (L, void, void), R)),
            front (tree (L, void, void), Ls),
            front (R, Rs),
            append (Ls, Rs, Xs).
```

augmented by definitions of the relations **nel_tree** and **append**. Note that by unfolding (see [?]) the atoms in the body of the third clause of the relation **front**, one can further optimize the program as follows:

```
front (void, [ ]).
front (tree (X, void, void), [X]).
front (tree (_, tree (L, void, void), R), [L|Rs]) ←  front (R, Rs).
```

which does not use both the relations **nel_tree** and **append**.

In this paper, we define a *specialised semantics* which captures the behaviour of call-correct derivations of a program P. This is obtained by generalizing the s-semantics approach [?,?,?] in order to handle call-conditions. We show that the specialised semantics can be computed both by a top-down and a bottom-up construction.
Moreover, we provide a sufficient condition to prove that a program is s.c.c. wrt to a given pre/call/post specification. It consists in one application of the specialised immediate consequence to the post-condition. A simple program specialisation which transforms s.c.c. programs into call-correct ones is also defined[a].

2 Preliminaries

The reader is assumed to be familiar with the terminology of and the basic results in the semantics of logic programs [?,?,?].

Let $\mathcal{L}$ be the first order language consisting of a finite set $\mathcal{C}$ of *data conctructors*, a finite set $\mathcal{P}$ of *predicate symbols*, a denumerable set $\mathcal{V}$ of *variable symbols*. Let $\mathcal{T}$ be the set of terms built on $\mathcal{C}$ and $\mathcal{V}$. Variable-free terms are called *ground*. A *substitution* is a mapping $\theta : \mathcal{V} \rightarrow \mathcal{T}$ such that the set $D(\theta) = \{X \mid \theta(X) \neq X\}$ (*domain* of θ) is finite. If $V \subset \mathcal{V}$, we denote by $\theta_{|V}$ the *restriction* of θ to the variables in V, i.e., $\theta_{|V}(Y) = Y$ for $Y \notin V$. Moreover if E is any expression, we use the abbreviation $\theta_{|E}$ to denote $\theta_{|\,Var(E)}$. ϵ denotes the empty substitution. The *composition* $\theta\sigma$ of the substitutions θ and σ is defined as the functional composition, i.e., $\theta\sigma(X) = \sigma(\theta(X))$. A *renaming* is a substitution ρ for which there exists the inverse ρ^{-1} such that $\rho\rho^{-1} = \rho^{-1}\rho = \epsilon$. The pre-ordering $\leq$ (more general than) on substitutions is such that $\theta \leq \sigma$ iff there exists θ' such that $\theta\theta' = \sigma$. We say that θ and σ are *not comparable* if neither $\theta \leq \sigma$ nor $\sigma \leq \theta$. The result of the application of the substitution θ to a term t is an *instance* of t denoted by $t\theta$. We define $t \leq t'$ (t is more general than t') iff there exists θ such that $t\theta = t'$. We say that t and t' are *not comparable* if neither $t \leq t'$ nor $t' \leq t$. The relation $\leq$ is a preorder. Let $\approx$ be the associated equivalence relation (*variance*). A substitution θ is a *unifier* of terms t and t' if $t\theta = t'\theta$. We denote by $mgu(t_1, t_2)$ any idempotent *most general unifier* (*mgu*, in short) of t_1 and t_2. All the above definitions can be extended to other syntactic objects in the obvious way.

Atoms, queries, clauses and programs in the language $\mathcal{L}$ are defined as follows. An *atom* is an object of the form $p(t_1, \ldots, t_n)$ where $p \in \mathcal{P}$ is an n-ary predicate symbol and $t_1, \ldots, t_n \in \mathcal{T}$. A *query* is a (possibly empty) finite sequence of atoms $A_1, \ldots, A_m$. The empty query is denoted by $\Box$. A *clause* is

[a] For an extended version of this paper the reader is referred to [?].

a formula of the form $H \leftarrow \mathbf{B}$ where H is an atom and $\mathbf{B}$ is a query. H is called the *head* of the clause and $\mathbf{B}$ its *body*. When $\mathbf{B}$ is empty, $H \leftarrow \mathbf{B}$ is written $H \leftarrow$ and is called a *unit clause*. A *program* is a finite set of clauses. Atoms are denoted by $A, B, C, H, \ldots$, queries by $Q, \mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$, clauses by $c, d, \ldots$, and programs by P. The language associated with a program P is obviously defined.

The computation process within the logic programming framework is based on the SLD resolution procedure. Consider a non empty query $\mathbf{A}, B, \mathbf{C}$ and a clause c. Let $H \leftarrow \mathbf{B}$ be a variant of c variable disjoint with $\mathbf{A}, B, \mathbf{C}$. Suppose that B and H unify and let θ be their mgu. We write then

$$\mathbf{A}, B, \mathbf{C} \stackrel{\theta}{\Longrightarrow}_{P,c} (\mathbf{A}, \mathbf{B}, \mathbf{C})\theta$$

and call it *SLD-derivation step of* $\mathbf{A}, B, \mathbf{C}$ *and* c *w.r.t.* B*, with an mgu* θ. $H \leftarrow \mathbf{B}$ is called its *input clause*. B and $B\theta$ are called the *selected atom* and the *selected atom instance*, respectively, of $\mathbf{A}, B, \mathbf{C}$. If the program P is clear from the context or the clause c is irrelevant we drop a reference to them. An SLD-derivation is obtained by iterating SLD-derivation steps. A maximal sequence

$$\delta := Q_0 \stackrel{\theta_1}{\Longrightarrow}_{P,c_1} Q_1 \stackrel{\theta_2}{\Longrightarrow}_{P,c_2} \cdots Q_n \stackrel{\theta_{n+1}}{\Longrightarrow}_{P,c_{n+1}} Q_{n+1} \cdots$$

of SLD-derivation steps is called an *SLD-derivation of* $P \cup \{Q_0\}$ if $Q_0, \ldots, Q_{n+1}, \ldots$ are queries, $\theta_1, \ldots, \theta_{n+1}, \ldots$ are substitutions, $c_1, \ldots, c_{n+1}, \ldots$ are clauses of P, and for every step the input clauses are standardized apart.

The length of an SLD-derivarion δ, denoted by $len(\delta)$, is the number of SLD-derivarion steps in δ. We denote by $Sel(\delta)$ the set of all the selected atom instances, one for each derivation step, of δ. SLD-derivations can be finite or infinite. Consider a finite SLD-derivation $\delta := Q_0 \stackrel{\theta_1}{\Longrightarrow}_{P,c_1} Q_1 \cdots \stackrel{\theta_n}{\Longrightarrow}_{P,c_n} Q_n$ of a query $Q := Q_0$, also denoted by $\delta := Q_0 \stackrel{\theta}{\longmapsto} Q_n$ with $\theta = \theta_1 \cdots \theta_n$. If $Q_n = \Box$ then δ is called *successful*. The restriction of θ to the variables of Q, denoted by $\theta_{|Q}$ is called a *computed answer substitution* (*c.a.s.*, in short) of Q and $Q\theta$ is called a *computed instance* of Q. If Q_n is non-empty and there is no input clause whose head unifies with the selected atom of Q_n, then the SLD-derivation δ is called *failed*.

3 Interpretations

By the *extended Herbrand base* $\mathcal{B}^{\mathcal{E}}_{\mathcal{L}}$ for a language $\mathcal{L}$ we mean the quotient set of all the atoms of $\mathcal{L}$ with respect to $\approx$. The ordering induced by $\leq$ on $\mathcal{B}^{\mathcal{E}}_{\mathcal{L}}$ will still be denoted by $\leq$. For the sake of simplicity, we will represent the equivalence class of an atom A by A itself. An *interpretation* I is any subset of $\mathcal{B}_{\mathcal{L}}^{\mathcal{E}}$. When the language is clear from the context then we drop a reference to it. We denote by $inst(I)$ the set of all instances of atoms in I and by $ground(I)$ the set of all ground instances of atoms in I. Moreover, we denote by $Min(I)$ the *set of minimal elements* of I defined as [?]: $Min(I) = \{A \in I \mid \forall A' \in I \text{ if } A' \leq A \text{ then } A = A'\}$.

Example 3.1 Let I be the set $\{\texttt{sublist}(u,v) \mid u$ is a list of at most two vowels and v is any term$\}$. Then

$Min(I) = \{\texttt{sublist}([\,], Y_s),\ \texttt{sublist}([a], Y_s), \ldots, \texttt{sublist}([a,e], Y_s), \ldots, \texttt{sublist}([o,u], Y_s)\}$
where Y_s is a variable.

The notion of truth extends the classical one to account for non-ground formulas in the interpretations. So, if A is an atom then $I \models A$ iff $A \in inst(I)$; whereas, if Q is a query of the form $A_1, \ldots, A_n$ then $I \models Q$ iff $A_i \in inst(I)$ for all $i \in \{1, \ldots, n\}$. Note that $I \models A$ iff there exists $A' \in I$ such that $A' \leq A$; moreover, if $I \models A$ then for all A' such that $A \leq A'$, $I \models A'$.

Definition 3.1 (Minimal Instances of an Atom Satisfying I) Let I be an interpretation and A be an atom. The *set of minimal instances of A satisfying I* is the set

$$Min_I(A) = Min(\{A' \in inst(A) \mid I \models A'\}).$$

Example 3.2 Consider the interpretation I of the Example 3.1. Then

$$Min_I(\texttt{sublist}([X|X_s], Y_s)) = \{\texttt{sublist}([a], Y_s), \ldots, \texttt{sublist}([a,e], Y_s), \ldots, \texttt{sublist}([o,u], Y_s)\}.$$

Observe that although I is infinite, both $Min(I)$ and $Min_I(\texttt{sublist}([X|X_s], Y_s))$ are finite.

The following notion of specialised unifier is the basic concept upon which specialised derivations are defined.

Definition 3.2 (Specialised Unifiers) Let I be an interpretation and t_1 and t_2 be terms. A substitution θ is a *I-unifier* of t_1 and t_2 if $t_1\theta = t_2\theta$ and $I \models t_1\theta$. A *most general I-unifier* (mgu_I, in short) of t_1 and t_2, denoted by

$$mgu_I(t_1, t_2),$$

is any idempotent I-unifier θ such that for any other I-unifier θ', either $\theta \leq \theta'$ or θ and θ' are not comparable.

Example 3.3 Consider again the interpretation I of the Example 3.1. Then

$$mgu_I(\texttt{sublist}([a, X], Y_s), \texttt{sublist}(Z_s, Y_s))$$

denotes both substitutions

$\theta_1 = \{X/e, Z_s/[a,e]\}$, $\theta_2 = \{X/i, Z_s/[a,i]\}$, $\theta_3 = \{X/o, Z_s/[a,o]\}$, $\theta_4 = \{X/u, Z_s/[a,u]\}$.

Note that the substitutions $\theta_1, \ldots, \theta_4$ are pairwise not comparable.
For the sake of simplicity, we will write $\theta = mgu_I(t_1, t_2)$ even if $mgu_I(t_1, t_2)$ is not uniquely determined.

It is well known that set inclusion does not adequately reflect the property of non-ground atoms of being representatives of all their ground instances. So, in this paper we refer to the partial ordering $\sqsubseteq$ on interpretations defined by Falaschi *et al.* [?] as follows:

- $I_1 \leq I_2$ iff $\forall A_1 \in I_1, \exists A_2 \in I_2$ such that $A_2 \leq A_1$.
- $I_1 \sqsubseteq I_2$ iff $(I_1 \leq I_2)$ and $(I_2 \leq I_1$ implies $I_1 \subseteq I_2)$.

Intuitively, $I_1 \leq I_2$ means that every atom verified by I_1 is also verified by I_2 (I_2 contains more *positive information*). Note that $\leq$ has different meanings for atoms and interpretations. $I_1 \sqsubseteq I_2$ means either that I_2 strictly contains more positive information than I_1 or (if the amount of positive information is the same) that I_1 expresses it by fewer elements than I_2 (I_2 is more redundant). The relation $\leq$ is a preorder, whereas the relation $\sqsubseteq$ is an ordering. If $I_1 \subseteq I_2$, then $I_1 \sqsubseteq I_2$. It is easy to see that for every interpretation I, $I \leq Min(I)$, but also $Min(I) \sqsubseteq I$. Moreover, for every atom A, $Min_I(A) \sqsubseteq Min(I)$.

The set of all the interpretations $\mathcal{I}$ with the ordering $\sqsubseteq$ is a complete lattice, noted by $\langle \mathcal{I}, \sqsubseteq \rangle$, where $\mathcal{B}^{\mathcal{E}}$ is the top element and $\emptyset$ is the bottom element.

4 Specialised Derivations

We are now ready to introduce our specialised derivations. They differ from the standard SLD-derivations for the fact that at each derivation step specialised most general unifiers are computed instead of standard mgus.

Definition 4.1 Assume given a program P and an interpretation I. Let $\mathbf{A}, B, \mathbf{C}$ be a non empty query, c be a clause, $H \leftarrow \mathbf{B}$ be a variant of c variable disjoint with $\mathbf{A}, B, \mathbf{C}$. Suppose that B and H unify and $\theta = mgu_I(B, H)$. We write then

$$\mathbf{A}, B, \mathbf{C} \xRightarrow{\theta}_{P,c,I} (\mathbf{A}, \mathbf{B}, \mathbf{C})\theta$$

and call it *I-derivation step of* $\mathbf{A}, B, \mathbf{C}$ *and* c *w.r.t.* B, *with an mgu* θ. $H \leftarrow \mathbf{B}$ is called its *I-input clause.* B and $B\theta$ are called the *I-selected atom* and the

I-selected atom instance, respectively, of $\mathbf{A}, B, \mathbf{C}$. An *I-derivation of* $P \cup \{Q_0\}$ is a maximal sequence

$$\delta := Q_0 \xrightarrow{\theta_1}_{P,c_1,I} Q_1 \xrightarrow{\theta_2}_{P,c_2,I} \cdots Q_n \xrightarrow{\theta_{n+1}}_{P,c_{n+1},I} Q_{n+1} \cdots$$

of I-derivation steps where the I-input clauses are standardized apart. Consider a finite I-derivation $\delta := Q_0 \xrightarrow{\theta_1}_{P,c_1,I} Q_1 \cdots \xrightarrow{\theta_n}_{P,c_n,I} Q_n$ of a query $Q := Q_0$, also denoted by $\delta := Q_0 \overset{\theta}{\longmapsto}_I Q_n$ with $\theta = \theta_1 \cdots \theta_n$. If $Q_n = \Box$ then δ is called *successful*. The restriction of θ to the variables of Q is called a *I-computed answer substitution* (*I-c.a.s.* in short) of Q and $Q\theta$ is called a *I-computed instance* of Q. If Q_n is non-empty and there is no I-input clause $H \leftarrow \mathbf{B}$ such that H unifies with the I-selected atom B of Q_n with a substitution $\theta = mgu_I(B, H)$, then δ is called *failed*.

Note that for every I-derivation δ, $Sel(\delta) \leq I$, or equivalently, $I \models A$ for all $A \in Sel(\delta)$.

Whenever I is omitted, we implicitly assume that $I = \mathcal{B}^{\mathcal{E}}$. It is easy to see that if I is the extended Herbrand base $\mathcal{B}^{\mathcal{E}}$, then I-derivations are indeed SLD-derivations.

Example 4.1 Consider the interpretation I of Example 3.1 and the program `SUBLIST`:

$\texttt{sublist}([\,], Y_s).$
$\texttt{sublist}([X|X_s], [X|Y_s]) : -\texttt{sublist}(X_s, Y_s).$
$\texttt{sublist}(X_s, [Y|Y_s]) : -\texttt{sublist}(X_s, Y_s).$

It produces two I-c.a.s. for the query $\texttt{sublist}([a, X], [a, e, a, b, a, u])$ that are $\theta_1 = \{X_s/[a, e]\}$ and $\theta_2 = \{X_s/[a, u]\}$, whereas any I-derivation of the query $\texttt{sublist}([a, b], [a, e, a, b, a, u])$ fails.

A Lifting Lemma for specialised derivations holds.

Lemma 4.1 (Specialised Lifting Lemma) *Let I be an interpretation and $\delta := Q\theta \overset{\sigma}{\longmapsto}_I \Box$ be a successful I-derivation of a query $Q\theta$. Then, there exists a successful I-derivation $\delta' := Q \overset{\sigma'}{\longmapsto}_I \Box$ where $\sigma' \leq \theta\sigma$.*

Proof 1 By induction on $len(\delta)$.

Basis. Let $len(\delta) = 1$. In this case Q consists of only one atom B and

$$\delta := B\theta \xrightarrow{\sigma}_I \Box$$

where the I-input clause used in the I-derivation step is a unit clause $H \leftarrow$ and $\sigma = mgu_I(B\theta, H)$. We can assume that $\theta_{|H} = \epsilon$. Then, $\theta\sigma$ is a I-unifier

of B and H. Hence, there exists $\sigma' = mgu_I(B, H)$ such that $\sigma' \leq \theta\sigma$.
Induction step. Let $len(\delta) > 1$. Then $Q := \mathbf{A}, B, \mathbf{C}$ and

$$\delta := (\mathbf{A}, B, \mathbf{C})\theta \overset{\sigma_1}{\Longrightarrow}_I (\mathbf{A}, \mathbf{B}, \mathbf{C})\theta\sigma_1 \overset{\sigma_2}{\longmapsto}_I \Box$$

where B is the I-selected atom of Q, $c := H \leftarrow \mathbf{B}$ is the first I-input clause, $\sigma_1 = mgu_I(B\theta, H)$ and $\sigma = \sigma_1\sigma_2$. We can assume that $\theta_{|c} = \epsilon$. Then, $\theta\sigma_1$ is a I-unifier of B and H. Hence, there exists $\sigma'_1 = mgu_I(B, H)$ such that $\sigma'_1 \leq \theta\sigma$. Let γ be a substitution such that $\sigma'_1\gamma = \theta\sigma_1$. By the inductive hypothesis, there exists a successful I-derivation

$$(\mathbf{A}, \mathbf{B}, \mathbf{C})\sigma'_1 \overset{\sigma'_2}{\longmapsto}_I \Box$$

where $\sigma'_2 \leq \gamma\sigma_2$. Therefore,

$$\delta' := (\mathbf{A}, B, \mathbf{C}) \overset{\sigma'_1}{\Longrightarrow}_I (\mathbf{A}, \mathbf{B}, \mathbf{C})\sigma'_1 \overset{\sigma'_2}{\longmapsto}_I \Box$$

with $\sigma' = \sigma'_1\sigma'_2$ is a successful I-derivation such that $\sigma' \leq \sigma'_1\gamma\sigma_2 = \theta\sigma_1\sigma_2 = \theta\sigma$.
$\Box$

5 Pre/Call/Post Specifications

In this section we propose a characterization of program behaviour in terms of pre/call/post specifications based on the notion of specialised derivarion defined above.

Definition 5.1 Let P be a program and *Pre*, *Call* and *Post* be interpretations. We say that P is *specialisable call correct* (*s.c.c.*, in short) wrt the pre-condition *Pre*, the call-condition *Call* and the post-condition *Post*, denoted by

$$\{Pre, Call\}P\{Post\}_{spec},$$

if and only if for any query Q,

$$Pre \models Q \text{ and } Q \overset{\theta}{\longmapsto}_{P,Call} \Box \text{ implies } Post \models Q\theta.$$

Example 5.1 Consider the program `SUBLIST` and the interpretations

$$\begin{aligned} Pre &= \{\texttt{sublist}(u, v) \mid u \text{ is a variable and } v \text{ is a ground list}\} \\ Call &= \{\texttt{sublist}(u, v) \mid u \text{ is a list of at most two vowels and } v \text{ is a term}\} \\ Post &= \{\texttt{sublist}(u, v) \mid u \text{ is a sublist of at most two vowels of the list } v\} \end{aligned}$$

The program `SUBLIST` is s.c.c. wrt *Pre*, *Call* and *Post*.

We define the strongest post-condition of P wrt *Pre* and *Call* as below.

Definition 5.2 (Strongest Post-condition) Let P be a program. The *strongest post-condition of P with respect to a pre-condition Pre and a call-condition Call*, noted $sp(P, Pre, Call)$, is the smallest interpretation $Post$ wrt to $\sqsubseteq$ such that $\{Pre, Call\}P\{Post\}_{spec}$.

Definition 5.3 Let P be a program and $Call$ be an interpretation. We say that P is *call-correct* wrt $Call$ iff for any SLD-derivation δ, $Sel(\delta) \leq Call$, i.e., δ is a $Call$-derivation.

6 Specialised Operational and Fixpoint Semantics

Based on the s-semantics approach[?], we define both a top-down and a bottom-up construction which model the specialised computed answer substitutions of the specialised derivations.

Definition 6.1 (Specialised Operational Semantics) Let P be a program and $Call$ be an interpretation. The *Call-computed answer substitution semantics of P* is

$$\begin{aligned}\mathcal{O}_{Call}(P) = \{A \in \mathcal{B}^{\mathcal{E}} \mid\ & \exists p \in \mathcal{P}, \exists X_1, \ldots, X_n \text{ distinct variables in } \mathcal{V},\ \exists\theta,\\ & p(X_1, \ldots, X_n) \overset{\theta}{\longmapsto}_{P,Call} \Box,\\ & A = p(X_1, \ldots, X_n)\theta\}.\end{aligned}$$

Note that if $Call$ is the extended Herbrand base $\mathcal{B}^{\mathcal{E}}$, then $\mathcal{O}_{\mathcal{B}^{\mathcal{E}}}(P)$ is the original s-semantics defined by Falaschi *et al.* in[?].

Example 6.1 Consider the program `SUBLIST` and the interpretation $Call$ of Example **??**.

$$\begin{aligned}\mathcal{O}_{Call}(\texttt{SUBLIST}) = &\ \{\texttt{sublist}([\,], [X_1, \ldots, X_n]),\ n \geq 0\} \cup\\ &\ \{\texttt{sublist}([a], [X_1, \ldots, X_n, a, X_{n+1}, \ldots, X_{n+m}]),\ n, m \geq 0\} \cup\\ &\ \ldots\\ &\ \{\texttt{sublist}([a, e], [X_1, \ldots, X_n, a, e, X_{n+1}, \ldots, X_{n+m}]),\ n, m \geq 0\} \cup \ldots\end{aligned}$$

We define a projection operator on the set of interpretations $\mathcal{I}$. It allows us to characterize the strongest post-condition of a program P with respect to a pre-condition Pre and a call-condition $Call$.

Definition 6.2 (Π_I) Let I and J be interpretations. The projection of J on the interpretation I is:

$$\begin{aligned}\Pi_I(J) = \{A \in \mathcal{B}^{\mathcal{E}} \mid\ & \exists A' \in Min(I), \exists A'' \in J\\ & \exists\theta = mgu(A', A'')\\ & A = A'\theta\}.\end{aligned}$$

Proposition 6.1 *Let P be a program and Pre and $Call$ be interpretations. Then $Min(\Pi_{Pre}(\mathcal{O}_{Call}(P))) = sp(P, Pre, Call)$.*

Proof 2 We prove that $\{Pre, Call\}P\{Post\}_{spec}$ holds, i.e., for any query Q such that $Pre \models Q$ and $Q \overset{\theta}{\longmapsto}_{Call} \Box$, then $Min(\Pi_{Pre}(\mathcal{O}_{Call}(P))) \models Q\theta$. Let $Q := A_1, \ldots, A_n$. It is easy to prove that for all $j \in \{1, \ldots, n\}$, there exists a successful *Call*-derivation $A_j\theta \overset{\gamma_j}{\longmapsto}_{Call} \Box$ where $A_j\theta\gamma_j = A_j\theta$. For all $j \in \{1, \ldots, n\}$, let $p_j \in \mathcal{P}$ and $X_1, \ldots, X_n$ be distinct variables in $\mathcal{V}$ such that $p_j(X_1, \ldots, X_n) \leq A_j\theta$. By Lemma 4.1, for all $j \in \{1, \ldots, n\}$, there exists a successful *Call*-derivation $p_j(X_1, \ldots, X_n) \overset{\theta_j}{\longmapsto}_{Call} \Box$ where $p_j(X_1, \ldots, X_n)\theta_j \leq A_j\theta$. By Definition ??, $p_j(X_1, \ldots, X_n)\theta_j \in \mathcal{O}_{Call}(P)$. Moreover, since $Pre \models A_j$, there exists $A'_j \in Min(Pre)$ such that $A'_j \leq A_j\theta$. Let $\theta'_j = mgu(p_j(X_1, \ldots, X_n)\theta_j, A'_j)$. By properties of substitutions, $A'_j\theta'_j \leq A_j\theta$ and by Definition ??, $A'_j\theta'_j \in \Pi_{Pre}(\mathcal{O}_{Call}(P))$. This proves that for all j, $Min(\Pi_{Pre}(\mathcal{O}_{Call}(P))) \models A_j\theta$ and then $Min(\Pi_{Pre}(\mathcal{O}_{Call}(P))) \models Q\theta$.

Further, for any interpretation J such that $\{Pre, Call\}P\{J\}_{spec}$, $Min(\Pi_{Pre}(\mathcal{O}_{Call}(P))) \sqsubseteq J$. First, $Min(\Pi_{Pre}(\mathcal{O}_{Call}(P))) \leq J$. In fact, if $A \in Min(\Pi_{Pre}(\mathcal{O}_{Call}(P)))$ then, by Definitions ?? and ??, there exist $p \in \mathcal{P}$, $X_1, \ldots, X_n$ distinct variables in $\mathcal{V}$ and a substitution θ such that $p(X_1, \ldots, X_n) \overset{\theta}{\longmapsto}_{Call} \Box$ and $A' \in Min(Pre)$ with $\theta' = mgu(p(X_1, \ldots, X_n)\theta, A')$ and $A = A'\theta'$. Therefore, by the hypothesis $\{Pre, Call\}P\{J\}_{spec}$, there exists $A'' \in J$ such that $A'' \leq A$. Finally, if $J \leq Min(\Pi_{Pre}(\mathcal{O}_{Call}(P)))$ then $Min(\Pi_{Pre}(\mathcal{O}_{Call}(P))) \subseteq J$. This follows immediately by Definition of operator Min. $\Box$

Observe that if Pre is the extended Herbrand base $\mathcal{B}^{\mathcal{E}}$, then $\Pi_{Pre}(\mathcal{O}_{Call}(P)) = \mathcal{O}_{Call}(P)$ and then, $sp(P, \mathcal{B}^{\mathcal{E}}, Call) = Min(\mathcal{O}_{Call}(P))$.

Lemma 6.1 *Let P be a program and Pre, $Call$ and $Post$ be interpretations. Then, $\{Pre, Call\}P\{Post\}_{spec}$ holds iff $sp(P, Pre, Call) \sqsubseteq Post$.*

In order to define the specialised fixpoint semantics, we introduce an immediate consequence operator $T_{P,I}$ on the set of interpretations $\mathcal{I}$. Its least fixpoint can be shown to be equivalent to the specialised operational semantics $\mathcal{O}_I(P)$

Definition 6.3 ($T_{P,I}$ Transformation) Let P be a program and I and J be interpretations.

$$\begin{aligned} T_{P,I}(J) = \{A \in \mathcal{B}^{\mathcal{E}} \mid\ & \exists H \leftarrow B_1, \ldots, B_n \in P, \\ & \exists B'_1, \ldots, B'_n \text{ variant of atoms in } J \text{ and renamed apart}, \\ & \exists \theta = mgu_I((B_1, \ldots, B_n), (B'_1, \ldots, B'_n)), \\ & A \in Min_I(H\theta)\}. \end{aligned}$$

Note that if I is the extended Herbrand base $\mathcal{B}^{\mathcal{E}}$, then $T_{P,\mathcal{B}^{\mathcal{E}}}$ coincides with the S-transformation T_S defined in [?].

Proposition 6.2 (Monotonicity and Continuity of $T_{P,I}$) *Let P be a program and I be an interpretation. The transformation $T_{P,I}$ is monotonic and continuous in the complete lattice $\langle \mathcal{I}, \subseteq \rangle$.*

Proof 3 Analogous to monotonicity and continuity of T_S in [?]. $\Box$

Definition 6.4 (Powers of $T_{P,I}$) As usual, we define powers of transformation $T_{P,I}$ as follows:

$$\begin{aligned} T_{P,I} \uparrow 0 &= \emptyset, \\ T_{P,I} \uparrow n+1 &= T_{P,I}(T_{P,I} \uparrow n), \\ T_{P,I} \uparrow \omega &= \bigcup_{n \geq 0}(T_{P,I} \uparrow n). \end{aligned}$$

Proposition 6.3 *$T_{P,I} \uparrow \omega$ is the least fixpoint of $T_{P,I}$ in the complete lattice $\langle \mathcal{I}, \sqsubseteq \rangle$.*

Proof 4 By proposition ??, $T_{P,I} \uparrow \omega$ is the least fixpoint of $T_{P,I}$ with respect to set inclusion. Moreover, for any fixpoint J of $T_{P,I}$, $T_{P,I} \uparrow \omega \subseteq J$, i.e., by Definition of $\sqsubseteq$, $T_{P,I} \uparrow \omega \sqsubseteq J$. $\Box$

We are now ready to formally define the specialised fixpoint semantics.

Definition 6.5 (Specialised Fixpoint Semantics) Let P be a program and *Call* be an interpretation. The *Call-fixpoint semantics of P* is defined as

$$\mathcal{F}_{Call}(P) = T_{P,Call} \uparrow \omega.$$

7 Specialised Programs

In this section we show that any s.c.c. program P wrt a given pre/call/post specification *Pre*, *Call* and *Post*, i.e., such that $\{Pre, Call\}P\{Post\}_{spec}$, can be transformed into a specialised program P_{Call} such that $\{Pre, \mathcal{B}^{\mathcal{E}}\}P_{Call}\{Post\}_{spec}$ holds and P_{Call} is call-correct wrt *Call*.

Specialised programs are formally defined as follows.

Definition 7.1 (Specialised Program) Let P be a program and *Call* be an interpretation. The *Call-program corresponding to P*, denoted by P_{Call}, is defined as:

$$P_{Call} = \{(H \leftarrow \mathbf{B})\gamma \mid H \leftarrow \mathbf{B} \in P \text{ and } H\gamma \in Min_{Call}(H)\}.$$

Observe that a variant of a clause of P_{Call} can be viewed as a clause of the form $(H \leftarrow \mathbf{B})\gamma$ where $H \leftarrow \mathbf{B}$ is a variant of a clause of P and $H\gamma \in Min_{Call}(H)$.

Example 7.1 Consider the program `SUBLIST` and the interpretations *Pre*, *Call* and *Post* given in the Example ??. The program `SUBLIST` can be transformed into a specialised program $\texttt{SUBLIST}_{Call}$ as follows.

$\texttt{sublist}([\,], Y_s).$
$\texttt{sublist}([a], [a|Y_s]) \colon - \texttt{sublist}([\,], Y_s).$
...
$\texttt{sublist}([a, e], [a|Y_s]) \colon - \texttt{sublist}([e], Y_s).$
...
$\texttt{sublist}([a], [Y|Y_s]) \colon - \texttt{sublist}([a], Y_s).$
...
$\texttt{sublist}([a, e], [Y|Y_s]) \colon - \texttt{sublist}([a, e], Y_s).$
...

It is easy to see that the assertion $\{Pre, \mathcal{B}^{\mathcal{E}}\}$ $\texttt{SUBLIST}_{Call}$ $\{Post\}_{spec}$ holds, meaning that for any query Q such that $Pre \models Q$ and successful SLD-derivation δ of $\texttt{SUBLIST}_{Call} \cup \{Q\}$ with computed answer substitution θ, $Post \models Q\theta$. Moreover, for any selected atom instance $A \in Sel(\delta)$, $Call \models A$.

Proposition 7.1 *Let P be a program such that $\{Pre, Call\}P\{Post\}_{spec}$. Then, $\{Pre, \mathcal{B}^{\mathcal{E}}\}P_{Call}\{Post\}_{spec}$ holds.*

Proof 5 Let Q be a query such that $Pre \models Q$ and $\delta := Q \stackrel{\theta}{\longmapsto}_{P_{Call}, \mathcal{B}^{\mathcal{E}}} \Box$ be a successful SLD-derivation of $P_{Call} \cup \{Q\}$. We prove that $Post \models Q\theta$. In order to obtain this result, we prove that there exists a successful $Call$-derivation $\delta' := Q\theta \stackrel{\sigma}{\longmapsto}_{P, Call} \Box$ of $P \cup \{Q\theta\}$ where $Q\theta\sigma = Q\theta$. The fact that $Post \models Q\theta$ follows from the hypothesis $\{Pre, Call\}P\{Post\}_{spec}$.
By induction on $len(\delta)$.
Basis. Let $len(\delta) = 1$. In this case, Q consists of only one atom B and

$$\delta := B \stackrel{\theta}{\Longrightarrow}_{P_{Call}, \mathcal{B}^{\mathcal{E}}} \Box$$

where the input clause is a unit clause of the form $(H \leftarrow)\gamma$ such that $H \leftarrow$ is a variant of a clause of P, $H\gamma \in Min_{Call}(H)$ and $\theta = mgu(B, H\gamma)$. We can assume that $H \leftarrow$ is variable disjoint with $B\theta$. Let σ be a substitution such that $\sigma_{|B\theta} = \epsilon$ and $\sigma_{|H} = \gamma\theta$. By properties of substitutions and Definition 3.2, $\sigma = mgu_{Call}(B\theta, H)$. Hence,

$$\delta' := B\theta \stackrel{\sigma}{\Longrightarrow}_{P, Call} \Box$$

is a successful $Call$-derivation of $B\theta$ such that $B\theta\sigma = B\theta$.
Induction step. Let $len(\delta) > 1$. In this case $Q := \mathbf{A}, B, \mathbf{C}$ and

$$\delta := \mathbf{A}, B, \mathbf{C} \stackrel{\theta_1}{\Longrightarrow}_{P_{Call}, \mathcal{B}^{\mathcal{E}}} (\mathbf{A}, \mathbf{B}\gamma, \mathbf{C})\theta_1 \stackrel{\theta_2}{\longmapsto}_{P_{Call}, \mathcal{B}^{\mathcal{E}}} \Box$$

where the first input clause has the form $(H \leftarrow \mathbf{B})\gamma$ such that $H \leftarrow \mathbf{B}$ is a variant of a clause of P, $H\gamma \in Min_{Call}(H)$ and $\theta_1 = mgu(B, H\gamma)$. Let

$\theta = \theta_1\theta_2$. We can assume that $H \leftarrow \mathbf{B}$ is variable disjoint with $Q\theta$. Let σ_1 be a substitution such that $\sigma_{1|Q\theta} = \epsilon$ and $\sigma_{1|H} = \gamma\theta$. By properties of substitutions and Definition 3.2, $\sigma_1 = mgu_{Call}(B\theta, H)$. Hence, by the inductive hypothesis and Definition 4.1, there exists a successful I-derivation

$$\delta' := (\mathbf{A}, B, \mathbf{C})\theta \xRightarrow{\sigma_1}_{P,Call} (\mathbf{A}, \mathbf{B}, \mathbf{C})\sigma_1 \xmapsto{\sigma_2}_{P,Call} \Box$$

where $Q\theta\sigma_1\sigma_2 = Q\theta$. $\Box$

Proposition 7.2 *Let P be a program such that $\{Pre, Call\}P\{Post\}_{spec}$. Then, P is call-correct wrt Call.*

Proof 6 Let δ be an SLD-derivation of $P_{Call} \cup \{Q\}$. We prove that for all $A \in Sel(\delta)$, $Call \models A$. Indeed, for all $A \in Sel(\delta)$ there exists and SLD-derivation step

$$\mathbf{A}, B, \mathbf{C} \xRightarrow{\theta}_{P_{Call}, \mathcal{B}^{\varepsilon}} (\mathbf{A}, \mathbf{B}, \mathbf{C})\theta$$

of δ where $A = B\theta$, B is the selected atom of the query $\mathbf{A}, B, \mathbf{C}$, $(H \leftarrow \mathbf{B})\gamma$ is the input clause used in the SLD-derivation step, $H \leftarrow \mathbf{B}$ is a variant of a clause of P, $H\gamma \in Min_{Call}(H)$ and $\theta = mgu(B, H\gamma)$. Hence, $Call \models H\gamma\theta = B\theta = A$. $\Box$

Both the operational and the fixpoint semantics of specialised programs are equivalent to the corresponding specialised semantics of the original programs, i.e., for any program P and interpretation I, both $\mathcal{O}_I(P) = \mathcal{O}(P_I)$ and $\mathcal{F}_I(P) = \mathcal{F}(P_I)$ hold. Due to space limitation, the reader is referred to [?] for a rigorous proof of Theorems stated below.

Theorem 7.1 *Let P be a program and Call be an interpretation. Then $\mathcal{O}(P_{Call}) = \mathcal{O}_{Call}(P)$.*

Proof 7 Recall that $\mathcal{O}(P_{Call}) = \mathcal{O}_{\mathcal{B}^{\varepsilon}}(P_{Call})$. The result follows from the following claim.

Claim: There exists a successful SLD-derivation $\delta := Q \xmapsto{\theta}_{P_{Call}, \mathcal{B}^{\varepsilon}} \Box$ of a query Q iff there exists a successful *Call*-derivation $\delta' := Q \xmapsto{\theta'}_{P,Call} \Box$ where $Q\theta = Q\theta'$. $\Box$

Theorem 7.2 *Let P be a program and Call be an interpretation. Then $\mathcal{F}(P_{Call}) = \mathcal{F}_{Call}(P)$.*

Proof 8 Recall that $\mathcal{F}(P_{Call}) = \mathcal{F}_{\mathcal{B}^{\varepsilon}}(P_{Call})$. The result follows from the following claim.

Claim: For all $n > 0$, $A \in T_{P_{Call}, \mathcal{B}^{\varepsilon}} \uparrow n$ iff $A \in T_{P,Call} \uparrow n$. $\Box$

The equivalence of the specialised operational and fixpoint semantics follows immediately.