Online appendix for the paper

Productive Corecursion in Logic Programming

published in Theory and Practice of Logic Programming

EKATERINA KOMENDANTSKAYA
Heriot-Watt University, Edinburgh, Scotland, UK
ek19@hw.ac.uk

YUE LI
Heriot-Watt University, Edinburgh, Scotland, UK
yl55@hw.ac.uk

submitted 02 May 2017; revised 20 June 2017; accepted 04 July 2017

Appendix A Supplementary Materials and Full Proofs

A.1 Least and Greatest Complete Herbrand Models

We recall the least and greatest complete Herbrand model constructions for LP (Lloyd 1988). We express the definitions in the form of a big-step semantics for LP, thereby exposing duality of inductive and coinductive semantics for LP in the style of (Sangiorgi 2011). We start by giving inductive interpretations to logic programs. We say that σ is a *grounding substitution for* t if $\sigma(t) \in \mathbf{GTerm}^{\omega}(\Sigma)$, and is just a *ground substitution* if its codomain is $\mathbf{GTerm}^{\omega}(\Sigma)$.

Definition Appendix A.1
Let P be a logic program. The *big-step rule for* P is given by

$$\frac{P \models \sigma(B_1), \ldots, P \models \sigma(B_n)}{P \models \sigma(A)}$$

where $A \leftarrow B_1, \ldots B_n$ is a clause in P and σ is a grounding substitution.

Following standard terminology (Aczel 1977; Sangiorgi 2011), we say that an inference rule is *applied forward* if it is applied from top to bottom, and that it is *applied backward* if it is applied from bottom to top. If a set of terms is closed under forward (backward) application of an inference rule, we say that it is *closed forward* (resp., *closed backward*) under that rule. If the i^{th} clause of P is involved in an application of the big-step rule for P, then we may say that we have applied the *big-step rule for* $P(i)$.

Definition Appendix A.2
The *least Herbrand model* for a program P is the smallest set $M_P \subseteq \mathbf{GTerm}(\Sigma)$ that is closed forward under the big-step rule for P.

Example Appendix A.1
The least Herbrand model for the program of Example 1.1 is $\{\mathtt{nat(0)}, \mathtt{nat(s(0))}, \mathtt{nat(s^2(0)}, \ldots\}$. We use $\mathtt{s}^2(0)$ for $\mathtt{s(s(0))}$, $s^3(0)$ for $\mathtt{s(s(s(0)))}$ and so on.

The requirement that $M_P \subseteq \mathbf{GTerm}(\Sigma)$ entails that only ground substitutions are used in the forward applications of the big-step rule involved in the construction of M_P. Next we give coinductive interpretations to logic programs. For this we do not impose any finiteness requirement on the codomain terms of σ.

Definition Appendix A.3
The *greatest complete Herbrand model* for a program P is the largest set $M_P^{\omega} \subseteq \mathbf{GTerm}^{\omega}(\Sigma)$ that is closed backward under the big-step rule for P.

Example Appendix A.2 (*Complete Herbrand model*)
The greatest complete Herbrand model for the program of Example 1.1 is $\{\mathtt{nat(0)}, \mathtt{nat(s(0))}, \mathtt{nat(s^2(0))} \ldots\} \bigcup \{\mathtt{nat(s^{\omega})}\}$. Indeed, there is an infinite inference for $\mathtt{nat(s^{\omega})} = \mathtt{nat(s(s(...)))}$ obtained by repeatedly applying the big-step rule for this program backward.

Definitions Appendix A.2 and Appendix A.3 could alternatively be given in terms of least and greatest fixed point operators, as in, e.g., (Lloyd 1988). To ensure that $\mathbf{GTerm}(\Sigma)$ and $\mathbf{GTerm}^{\omega}(\Sigma)$ are non-empty, and thus that the least and greatest Herbrand model constructions are as intended, it is standard in the literature to assume that Σ contains at least one function symbol of arity 0. We will make this assumption throughout the remainder of this paper.

A.2 Proof of Productivity Lemma 4.1

Let P be an observationally productive and universal program and let $t \in \mathbf{Term}(\Sigma)$. Let D be an infinite S-resolution derivation given by $G_0 = t \rightsquigarrow^S G_1 \rightsquigarrow^S G_2 \rightsquigarrow^S \ldots$, then for every $G_i \in D$, there is a $G_j \in D$, with $j > i$, such that, given computed mgus $\theta_i, \ldots, \theta_1$ up to G_i and the computed mgus $\theta_j, \ldots, \theta_1$ up to G_j, $d(t^{\infty}, \theta_i, \ldots, \theta_1(t)) > d(t^{\infty}, \theta_j, \ldots, \theta_1(t))$, for some term $t^{\infty} \in \mathbf{Term}^{\infty}(\Sigma)$.

The proof has two parts, as follows. Part 1 shows that, under the imposed productivity and universality conditions, no infinite sequence of trivial unifiers is possible for infinite S-resolution derivations. Therefore, an infinite S-resolution derivation must contain an infinite number of non-trivial substitutions. Part 2 uses this fact and shows that a composition of an infinite number of non-trivial substitutions must result in an infinite term (this holds under universality condition only).

Proof
Recall that, by definition of S-resolution reductions, each step $G_k \rightsquigarrow^S G_{k+1}$ is a combination of a finite number of steps $G_k \rightarrow^n [A_1, \ldots, A_n]$ and one substitution+rewriting step $[A_1, \ldots, A_j, \ldots, A_n] \rightarrow \circ \hookrightarrow G_{k+1}$, this final step involves computation of an mgu (but not mgm) θ_{k+1} of some clause $C \leftarrow C_1, \ldots, C_n$ and some A_j. So in fact

$$G_{k+1} = [\theta_{k+1}(A_1), \ldots, \theta_{k+1}(A_{j-1}), \theta_{k+1}(C_1), \ldots, \theta_{k+1}(C_n), \theta_{k+1}(A_{j+1}), \ldots, \theta_{k+1}(A_n)]$$

Moreover, since θ_{k+1} is not an mgm, we have that:
θ_{k+1} is a non-trivial substitution for at least one variable X in A_j. (**)

We will use the above facts implicitly in the proof below.

To proceed with our proof, first we need to show that

(1) For all $k > 1$ in $G_0 = t \rightsquigarrow^S G_1 \rightsquigarrow^S G_2 \rightsquigarrow^S \ldots \rightsquigarrow^S G_k \rightsquigarrow^S \ldots$, the composition $\theta_k \ldots \theta_1$ is non-trivial for t.

We prove this by induction.

Base case. By $(**)$, θ_1 is necessarily non-trivial for initial goal t.

Inductive case. If θ_k is non-trivial for term $\theta_{k-1} \ldots \theta_1(t)$, then by universality of P and $(**)$, θ_{k+1} is non-trivial for term $\theta_k \ldots \theta_1(t)$. Then by induction, for all $k > 1$, the composition $\theta_k \ldots \theta_1$ is non-trivial for t.

Next, we need to show that the property (1) implies that

(2) we can define the limit term t^∞ using the infinite sequence

$$t, \quad \theta_1(t), \quad \theta_2\theta_1(t), \quad \theta_3\theta_2\theta_1(t), \quad \ldots$$

To prove (2), we prove the following property:

(2.1) For each $n \in \mathbb{N}$, there exists θ_{k_n}, so that for all k, if $k > k_n$, then truncation of $\theta_k \ldots \theta_1(t)$ at depth n is the same as the truncation of $\theta_{k_n} \ldots \theta_1(t)$ at depth n.

We prove this fact by contradiction. Assume the negation of our proposition, which says there exists depth value n such that for all substitution subscript k, there exists some k_n, so that $k_n > k$ and truncation of $\theta_{k_n} \ldots \theta_1(t)$ at depth n is different from the truncation of $\theta_k \ldots \theta_1(t)$ at depth n. This is impossible because this implies that non-trivial substitution can be infinitely applied within the truncation at depth n but no finitely branching tree can accommodate infinite amount of variables up to any fixed depth.

This gives us a way to prove (2):

We build t^∞ inductively, for each depth n of t^∞. For depth $n = 0$, we let t^∞ have as its root symbol the predicate symbol $t(\varepsilon)$ of initial atomic goal t. If t^∞ is defined up till depth $n \geq 0$, then, by (2.1) we know that there is some k_n such that for all $k > k_n$,

$$\gamma'(n, \theta_k \ldots \theta_1(t)) = \gamma'(n, \theta_{k_n} \ldots \theta_1(t))$$

We also know by (2.1) that there is some k_{n+1} such that for all $k > k_{n+1}$,

$$\gamma'((n+1), \theta_k \ldots \theta_1(t)) = \gamma'((n+1), \theta_{k_{n+1}} \ldots \theta_1(t))$$

Then we find the greater value κ in $\{k_n, k_{n+1}\}$, or set $\kappa = k_n$ if $k_n = k_{n+1}$, and define the nodes at depth $n+1$ for t^∞ in the same way as $\theta_\kappa \ldots \theta_1(t)$.

□

A.3 Proof of Theorem 4.1 Soundness and Completeness of Infinite S-resolution Relative to SLD-computations at Infinity

Let P be an observationally productive and universal program, and let $t \in \mathbf{Term}(\Sigma)$. There is an infinite fair S-resolution derivation for t iff there is a $t' \in \mathbf{Term}^\infty(\Sigma)$, such that t' is SLD-computable at infinity by t.

Proofs in both directions start with establishing operational equivalence of *infinite* S-resolution and SLD-resolution derivations. Coinductive proof principle is employed in this part of the proof. The proof in the left-to-right direction proceeds by using this equivalence,

and by applying Lemma 4.1 to show that an infinite fair S-resolution derivation must result in an SLD-computation of an infinite term at infinity. The other direction is proven trivially from the operational equivalence of infinite S-resolution and SLD-resolution derivations.

Proof

1. Suppose $D = t \rightsquigarrow^S G_1 \rightsquigarrow^S G_2 \rightsquigarrow^S \ldots$ is an infinite fair S-derivation. It is easy to construct a corresponding SLD-resolution derivation D^*, we prove this fact by coinduction. Consider $t \rightsquigarrow^S G_1$, which in fact can be given by one of two cases:
   (a) $t \hookrightarrow \theta(t) \rightarrow G_1$, i.e. if t does not match, but is unifiable with some clause $P(i)$ via a substitution θ. In this case, the first step in D^* will be to apply SLD-resolution reduction to t and $P(i)$: $t \rightsquigarrow G_1$.
   (b) $t \rightarrow^n [A_1, \ldots, A_j, \ldots, A_n] \hookrightarrow [\theta(A_1), \ldots, \theta(A_j), \ldots, \theta(A_n)] \rightarrow G_1$; obtained by resolving A_j with a clause $P(i)$ and computing a substitution θ. Then, in D^*, we will have n steps by SLD-resolution reductions involving exactly the resolvents of goal atoms and clauses used in $t \rightarrow^n [A_1, \ldots, A_n]$ (note that mgms used in $\rightarrow^n$ are also mgus by definition). These n steps in D^* will be followed by one step of SLD-resolution reduction, resolving A_j with $P(i)$ using substitution θ.
   We can proceed coinductively to construct D^* from D starting with $G_1 \in D$.

   We now need to show that such D^* is fair and non-failing. By definition, $t \rightsquigarrow^S G_1 \rightsquigarrow^S G_2 \rightsquigarrow^S \ldots$ should contain atoms which are resolved against finitely often. This means that corresponding derivation D^* will be fair. Because D is non-terminating and non-failing, D^* using the same resolvents will be non-terminating and non-failing, too.
   Finally, we need to show that $D^* = t \rightsquigarrow G_1^* \rightsquigarrow G_2^* \rightsquigarrow \ldots$ constructed as described above involves computation of an infinite term t' at infinity. This can only happen if, for every $G_i^* \in D^*$, there is a $G_j^* \in D^*$, with $j > i$, such that, given computed mgus $\theta_i, \ldots, \theta_1$ up to G_i^* and the computed mgus $\theta_j, \ldots, \theta_1$ up to G_j^*, $d(t', \theta_i, \ldots, \theta_1(t)) > d(t', \theta_j, \ldots, \theta_1(t))$. For this to hold, the S-resolution derivation D should satisfy the same property, but this follows from Lemma 4.1.
2. The proof proceeds by coinduction. Consider the SLD-resolution derivation $D^* = t \rightsquigarrow G_1^* \rightsquigarrow G_2^* \rightsquigarrow \ldots$ that computes an infinite term t' at infinity. Consider the substitution θ associated with $t \rightsquigarrow G_1^*$. If it is an mgm of t and some clause $P(i)$, then we can construct the first step of S-resolution reduction using the rewriting reduction: $t \rightarrow G_1^*$. If θ is not an mgm, i.e. it is an mgu, then we can construct first two steps of the S-resolution reduction: $t \hookrightarrow \theta(t) \rightarrow G_1^*$. We can proceed building D from D^* in the same way, now starting from G_1^*. We only need to show that D is fair and non-failing, but that follows trivially from properties of D^*.

□

A.4 Standard co-SLD-resolution and Proof of Soundness of Co-S-resolution

In this subsection, we introduce the standard definition of co-SLD-derivations (Ancona and Dovier 2015), and re-use the proof of their soundness with respect to the greatest complete Herbrand models to establish a similar result for co-S-resolution.

Definition Appendix A.4 (*Co-SLD-reductions (Ancona and Dovier 2015)*)

Given a logic program P, we distinguish the following reductions in the context of co-inductive logic programming.

- *SLD reduction ($G \rightsquigarrow G'$):* Let $G = [(A_1, S_1), \ldots, (A_n, S_n)]$. If $B_0 \approx_\theta A_k$ for some program clause $B_0 \leftarrow B_1, \ldots, B_m$ and some k, then let $S' = S_k \cup \{A_k\}$, we derive

  $$G' = \theta\big([(A_1, S_1), \ldots, (A_{k-1}, S_{k-1}), (B_1, S'), \ldots, (B_m, S'), (A_{k+1}, S_{k+1}), \ldots, (A_n, S_n)] \big)$$

- *loop detection ($G \rightarrow_\infty G'$):* Let $G = [(A_1, S_1), \ldots, (A_n, S_n)]$. If $A_k \approx_\theta B$ for some k and some $B \in S_k$, we derive

  $$G' = \theta\big([(A_1, S_1), \ldots, (A_{k-1}, S_{k-1}), (A_{k+1}, S_{k+1}), \ldots, (A_n, S_n)] \big)$$

- *co-SLD reduction ($G \rightsquigarrow_{co} G'$):* $G \rightsquigarrow_{\text{co}} G'$ if $G \rightsquigarrow G'$ or $G \rightarrow_\infty G'$.

Co-SLD-resolution is proven sound in (Ancona and Dovier 2015; Simon et al. 2006), i.e. if a logic program P and an atomic goal G have a co-SLD-refutation with computed answer substitution θ, then all ground instances of $\theta(G)$ are in the greatest complete Herbrand model of P.

An important property of co-S-resolution is coinductive soundness.

Proposition Appendix A.1 (*Soundness of co-S-resolution*)
If a logic program P and an atomic initial goal G have a co-S-refutation with computed answer substitution θ, then all ground instances of $\theta(G)$ are in the greatest complete Herbrand model of P.

We will base the proof on the soundness of co-SLD resolution (Ancona and Dovier 2015; Simon et al. 2006).

Proof
If loop detection is not used at all in the co-S-refutation, then the co-S-refutation reduces to a S-refutation, which is sound w.r.t to least Herbrand model, thus also being sound w.r.t the greatest complete Herbrand model.

Let us assume loop detection is used for at least once. We show that for any co-S-refutation there exists a corresponding co-SLD refutation. Any substitution+rewriting step $G_i \hookrightarrow G_{i+1} \rightarrow G_{i+2}$ corresponds to one step of SLD reduction (in co-SLD setting) $G_i \rightsquigarrow G_{i+2}$. Any rewriting reduction step $G_i \rightarrow G_{i+1}$ that does not follow a substitution reduction step also constitutes a SLD-reduction step $G_i \rightsquigarrow G_{i+1}$. In this way any co-S-refutation can be converted to a refutation that only involves SLD-reduction and loop detection, thus constituting a co-SLD refutation, which is sound w.r.t the greatest complete Herbrand model.
□

A.5 Proof of Theorem 5.1 of Soundness of Co-S-resolution Relative to SLD-Computations at Infinity

Let P be an observationally productive and universal logic program, and $t \in \mathbf{Term}(\Sigma)$ be an atomic goal. If there exists a co-S-refutation for P and t that involves the restricted loop detection rule, and computes the substitution θ then

1. *there exists an infinite fair S-derivation for P and t, and*

2. *there is a term* $t^\infty \in \mathbf{Term}^\infty(\Sigma)$ *SLD-computed at infinity that is a variant of* $\theta(t)$.

The proof will proceed according to the following scheme. For the sake of the argument, we take some arbitrary logic program that satisfies the productivity and universality conditions. We first show that the use of (any) loop detection necessarily results in computation of circular substitutions. Next, we analyse the effect of the restriction that was introduced to loop detection in Definition 5.1 and build the infinite regular S-derivation starting at the point where the restricted loop detection was once used. Finally, we show that the sequence of (non-circular) unifiers computed by the infinite S-derivation is equivalent to the single (circular) unifier computed by the restricted loop detection of Definition 5.1. If there are several uses of the restricted loop detection rule, then each implies a separate infinite derivation, and they can be interleaved to form an infinite fair S-derivation. This argument relies only on the observational productivity, leading to the conclusion that *for an observationally productive program, if it has a co-S-refutation then there exists an infinite fair S-derivation in which a sequence of computed unifiers "unfolds" the circular unifier*. A program that is also universal is a special case, where (by Theorem 4.1) the infinite sequence of unifiers instantiates the initial goal into an infinite formula, which shall be a variant of the formula computed by co-S-refutation.

Before proceeding with the full proof, we first need to introduce the method of *decircularization*. A circular substitution means a substitution of infinite regular terms for variables. For e.g. $\{X \mapsto s(X)\}$ is equivalent to $\{X \mapsto s^\omega\}$, where s^ω is obtained by continued substitution of $s(X)$ for X, which can further be regarded as applying an infinite succession of non-circular substitutions $s(X_1)$ for X, $s(X_2)$ for X_1, $s(X_3)$ for X_2, and so on. We coin the term *decircularization* for the process of obtaining from a circular substitution σ an equivalent infinite succession of non-circular substitutions $\sigma_1, \sigma_2, \ldots$. In the following proof we relate co-S-resolution's answers to terms SLD-computable at infinity by showing that the circular substitutions computed by co-S-refutation have decircularization computed by infinite S-derivation. Formal definition of decircularization (with motivating examples) is given below.

Definition Appendix A.5 (*Decircularization*)
Let $\sigma = \{\ldots, X_k \mapsto t, \ldots\}$ be a circular substitution where $X_k \mapsto t$ a circular component and $FV(t) = \{X_1, \ldots, X_k, \ldots, X_m\}$. $X_k \mapsto t$ can be *decircularized* into an infinite set $R = \{X_k \mapsto t_{(1)}, X_{k_{(1)}} \mapsto t_{(2)}, X_{k_{(2)}} \mapsto t_{(3)}, \ldots\}$ where $t_{(n)}$ is a variant of t obtained by applying renaming $\{X_i \mapsto X_{i_{(n)}} \mid \forall i \in [1,m]\}$ to t. We call the set R a *decircularization* of $X_k \mapsto t$. The *decircularization* of σ is the union of all decircularizations of individual circular components of σ.

Example Appendix A.3 (*Decircularization*)
Let $\sigma = \{A_1 \mapsto f(A_1, B_1, C_1), B_1 \mapsto s(B_1)\}$. The decircularization of σ is $R_A \cup R_B$ where $R_A = \{A_1 \mapsto f(A_{1_{(1)}}, B_{1_{(1)}}, C_{1_{(1)}}), A_{1_{(1)}} \mapsto f(A_{1_{(2)}}, B_{1_{(2)}}, C_{1_{(2)}}), A_{1_{(2)}} \mapsto f(A_{1_{(3)}}, B_{1_{(3)}}, C_{1_{(3)}}), \ldots\}$. and $R_B = \{B_1 \mapsto s(B_{1_{(1)}}), B_{1_{(1)}} \mapsto s(B_{1_{(2)}}), B_{1_{(2)}} \mapsto s(B_{1_{(3)}}), \ldots\}$. With the understanding that subscriptions merely serve the purpose of distinguishing names, we can simplify the decircularization into $R_A = \{A_1 \mapsto f(A_2, B_2, C_2), A_2 \mapsto f(A_3, B_3, C_3), A_3 \mapsto f(A_4, B_4, C_4), \ldots\}$ and $R_B = \{B_1 \mapsto s(B_2), B_2 \mapsto s(B_3), B_3 \mapsto s(B_4), \ldots\}$. Note the way A's and B's interact in the decircularization.

Example Appendix A.4
Consider program:
r(f(A,B,C), s(B)) ← r(A,B).

A co-S-refutation D is:
$[\big(r(X,Y),\emptyset\big)] \hookrightarrow^{X\mapsto f(A_1,B_1,C_1),Y\mapsto s(B_1)} [\big(r(f(A_1,B_1,C_1),s(B_1)),\emptyset\big)] \rightarrow$
$\underline{[\big(r(A_1,B_1),\{r(f(A_1,B_1,C_1),s(B_1))\}\big)]} \rightarrow_{\text{co}}^{A_1\mapsto f(A_1,B_1,C_1),B_1\mapsto s(B_1)} [\,]$

If we continue the derivation from the underlined goal in D, but now use S-resolution, we have an infinite S-derivation D^* as follows:
$[r(X,Y)] \hookrightarrow^{X\mapsto f(A_1,B_1,C_1),Y\mapsto s(B_1)} [r(f(A_1,B_1,C_1),s(B_1))] \rightarrow$
$\underline{[r(A_1,B_1)]} \hookrightarrow^{A_1\mapsto f(A_2,B_2,C_2),B_1\mapsto s(B_2)} [r(f(A_2,B_2,C_2),s(B_2))] \rightarrow$
$[r(A_2,B_2)] \hookrightarrow^{A_2\mapsto f(A_3,B_3,C_3),B_2\mapsto s(B_3)} [r(f(A_3,B_3,C_3),s(B_3))] \rightarrow$
$[r(A_3,B_3)] \cdots$

Note that the mgu's computed by infinite S-derivation starting from the underlined goal in D^* is a decircularization of the circular substitution computed by co-S-resolution from the corresponding goal in D. The details of computing decircularization for the circular substitution of this example is given in Example Appendix A.3. We see that in this example co-S-resolution is a perfect finite model for corresponding infinite S-resolution.

Proof
We first take an arbitrary coinductive logic program that satisfies our productivity and universality conditions.

1. *We first show that the use of loop detection (no matter restricted or not) necessarily results in creation of circular substitutions.*
   Generally, consider some subgoal of the form $(A,\{A_1,\ldots A_n\})$ where for all A_i in the ancestors set of A, A_i is added to the ancestors set later than A_{i+1}. By definition of co-S-derivation, there exists some program clause instances (or variants)
   $$\begin{aligned} A_n &\leftarrow \ldots, A_{n-1}, \ldots \\ &\vdots \\ A_2 &\leftarrow \ldots, A_1, \ldots \\ A_1 &\leftarrow \ldots, A, \ldots \end{aligned}$$
   where all A and A_k $(1 \leq k \leq n)$ are finite, so are the omitted atoms (which are represented by "…") in the above set of clauses.
   Assume *restricted loop detection* is applicable for the subgoal under consideration. This means that A unifies with some A_i (occurs check switched off) under mgu σ, and $A \prec A_i'$ where A_i' is a fresh-variable variant of A_i. Each of the two conditions $A \approx_\sigma A_i$ and $A \prec A_i'$ has implication.
   Note that σ is a circular substitution: if σ is not a circular substitution, then we have the set of program clause instances (or variants)
   $$\begin{aligned} \sigma(A_i) &\leftarrow \ldots, \sigma(A_{i-1}), \ldots \\ &\vdots \\ \sigma(A_2) &\leftarrow \ldots, \sigma(A_1), \ldots \\ \sigma(A_1) &\leftarrow \ldots, \sigma(A), \ldots \end{aligned}$$
   where $\sigma(A) = \sigma(A_i)$ (because $A \approx_\sigma A_i$) and all atoms are finite. Then there exists

non-terminating rewriting reduction steps for $\sigma(A_i)$ and thus breaks the observational productivity condition. Therefore σ is circular.

2. *Next we construct the general form of the repeating derivation pattern by analysing the effect of the restricted loop detection. This pattern is then used to build the infinite regular derivation that can start at the point where the restricted loop detection was once used.*
   An infinite sequence $\{\gamma_n\}_{n\geq 0}$ of trivial substitutions is defined as follows. Let γ_0 be the empty substitution and γ_1 be a variable renaming substitution for A_i with fresh names. Let $\{\gamma_n\}_{n\geq 1}$ be an infinite sequence of renaming substitutions, such that, for all $n \geq 2$, the domain of γ_n equals to the image of γ_{n-1}, while the image of γ_n is disjoint from the set of all variables that occur in the domain of one of γ_k $(1 \leq k \leq n)$.
   For example, if $\gamma_1 = \{X_1 \mapsto X_2\}$, then the sequence of renaming substitutions $\{X_1 \mapsto X_2\}, \{X_2 \mapsto X_3\}, \{X_3 \mapsto X_4\}, \ldots$ conforms to the above description for $\{\gamma_n\}_{n\geq 1}$.
   For all $n \geq 0$, $\gamma_{n+1} \cdots \gamma_0(A_i)$ is an instance of $\gamma_n \cdots \gamma_0(A)$, as is implied by $A \prec A'_i$. Let σ_{n+1} denote the matcher for the pair $\gamma_n \cdots \gamma_0(A)$ and $\gamma_{n+1} \cdots \gamma_0(A_i)$, we have the important equation

$$\sigma_{n+1}\gamma_n \ldots \gamma_0(A) = \gamma_{n+1} \ldots \gamma_0(A_i), \quad \text{for all } n \geq 0$$

   We also have a set of program clause instances (or variants) for all $n \geq 0$:

$$\begin{array}{lcl}
\gamma_{n+1} \cdots \gamma_0(A_i) & \leftarrow & \ldots, \quad \gamma_{n+1} \cdots \gamma_0(A_{i-1}), \quad \ldots \\
\vdots & & \\
\gamma_{n+1} \ldots \gamma_0(A_2) & \leftarrow & \ldots, \quad \gamma_{n+1} \ldots \gamma_0(A_1), \quad \ldots \\
\gamma_{n+1} \ldots \gamma_0(A_1) & \leftarrow & \ldots, \quad \gamma_{n+1} \ldots \gamma_0(A), \quad \ldots
\end{array}$$

   Using the above clauses for rewriting reductions, we have that for all $n \geq 0$, there exists the repeating S-derivation *pattern*:

$$[\ldots, \gamma_n \cdots \gamma_0(A), \ldots] \hookrightarrow [\ldots, \sigma_{n+1}\gamma_n \ldots \gamma_0(A), \ldots] \rightarrow^i [\ldots, \gamma_{n+1} \ldots \gamma_0(A), \ldots]$$

   Therefore an infinite S-derivation starting from goal $[\ldots, A, \ldots]$ can be given in the form

$$\begin{array}{l}
[\ldots, \quad \gamma_0(A), \quad \ldots] \hookrightarrow [\ldots, \quad \sigma_1\gamma_0(A), \quad \ldots] \rightarrow^i \\
[\ldots, \quad \gamma_1\gamma_0(A), \quad \ldots] \hookrightarrow [\ldots, \quad \sigma_2\gamma_1\gamma_0(A), \quad \ldots] \rightarrow^i \\
[\ldots, \quad \gamma_2\gamma_1\gamma_0(A), \quad \ldots] \hookrightarrow \quad \cdots
\end{array}$$

3. *Finally we show that the collection of (non-circular) unifiers computed by the infinite derivation is equivalent to the single (circular) unifier computed by restricted loop detection.*
   Consider a circular component $X \mapsto t[X]$ of σ. This circular component corresponds to a mapping $X_n \mapsto t[X_{n+1}]$ in each matcher σ_{n+1} $(n \geq 0)$. Therefore the collection of all such $X_n \mapsto t[X_{n+1}]$ constitutes a decircularization of $X \mapsto t[X]$, and all other circular components of σ are similarly decircularized. Therefore σ has the decircularization $\bigcup_{n=1}^{\infty} \sigma_n$.

If there are several use of restricted loop detection, then each implies a separate infinite derivation, which can be interleaved to form an infinite fair S-derivation.

So far only the observational productivity condition has been used, and the conclusion is that

For observationally productive programs, if there is a co-S-refutation involving restricted loop detection and a circular unifier θ, then 1) there exists an infinite fair co-S-derivation that 2) computes an infinite sequence of unifiers equivalent to the circular unifier θ. ($\star$)

A program that is also universal is a special case of ($\star$), which, by 1) of ($\star$), has an infinite fair S-derivation whose unifiers, instantiate the initial goal into an infinite formula (by Theorem 4.1). But then by 2) of ($\star$), a composition of these unifiers must compute a variant of the formula computed by co-S-refutation. □

A.6 Implementation of Co-S-resolution

The co-S-resolution meta-interpreter is written in SWI-Prolog, and is available at `https://github.com/coalp/Productive-Corecursion`. It adopts left first computation rule and depth first search rule in the SLD tree.

The entry procedure requires a unary predicate named `clause_tree`, which takes a conjunctive goal or an atomic goal as an input. After assignment of an empty ancestors set to the goal, a case analysis on the shape of the goal passes an atomic goal to procedures corresponding to reduction rules, or disassembles a conjunctive goal into its head and tail, and processes the head and the tail separately and recursively, starting with a case analysis on their shape.

Three kinds of reduction rule: rewriting reduction, substitution reduction and restricted loop detection are coded separately as three alternative procedures to process an atomic goal. Since object programs to be processed by the meta-interpreter are intended to be non-terminating, and given the execution model of Prolog, it is necessary to put the loop detection rule ahead of other rules, otherwise in a non-terminating derivation it will never be called. The rewriting reduction is put at the second place, and the substitution reduction is tried only when both the loop detection and the rewriting reduction are not applicable. The ordering of the rules, therefore, also makes sure that rewriting happens after each substitution reduction, because if a sub-goal cannot be reduced by loop detection, an instance of this sub-goal from substitution reduction still cannot be reduced by loop detection.

References

ACZEL, P. 1977. An introduction to inductive definitions. *Studies in Logic and the Foundations of Mathematics 90*, 739 – 782.

ANCONA, D. AND DOVIER, A. 2015. A theoretical perspective of coinductive logic programming. *Fundam. Inform. 140,* 3-4, 221–246.

LLOYD, J. 1988. *Foundations of Logic Programming*, 2nd ed. Springer-Verlag.

SANGIORGI, D. 2011. *Introduction to Bisimulation and Coinduction*. Cambridge University Press.

SIMON, L., MALLYA, A., BANSAL, A., AND GUPTA, G. 2006. Coinductive logic programming. In *ICLP'06*. 330–345.