Models of Coinductive First-order Horn Clauses

Yue Li

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

Abstract: Proof-theoretic study of coinductive Horn clauses needs to establish soundness of proposed coinductive algorithms. Soundness is with respect to coinductive models of logic programs. In this paper we give examples on coinductive Horn clauses, and we review the two standard coinductive models for logic programs.

1 Introduction

Coinduction in logic programming refers to phenomena of non-terminating SLD-derivations and perhaps computation of infinite data therein. The name "coinduction" alludes to some association with "induction", explained later. Work on coinductive Horn clauses started in the 80s up till now, with significant progress made on model-theoretic semantics of infinite computation [4] and finite modelling of coinduction using loop detection [2, 3], and in the reference literatures there are motivating applications.

There are different ways to categorize coinductive Horn clauses, such as, whether or not the derivation is subject to loop detection, and, whether or not infinite data is computed. We give examples of coinductive Horn clauses below.

Example 1. Given program $P_1 : r(X) \supset r(X)$ (we use $\supset$ for implication), the goal $r(a)$ gives rise to an infinite SLD-derivation $r(a) - r(a) - r(a) - \cdots$ which does not compute infinite data but is subject to loop detection as there exist a sub-goal $r(a)$ that can be unified with an ancestor goal $r(a)$.

Example 2. Given $P_2 : p(s(X)) \supset p(X)$, the goal $p(a)$ gives rise to an infinite SLD-derivation $p(a) - p(s(a)) - p(s(s(a))) - \cdots$ which does not compute infinite data and is not subject to loop detection as there does not exist a sub-goal that can be unified with its ancestor goal.

Example 3. Given $P_3 : p(X) \supset p(s(X))$, the goal $p(X)$ gives rise to an infinite SLD-derivation $p(X) - p(X_1) - p(X_2) - \cdots$. In this case loop detection is applicable, for instance, sub-goal $p(X_1)$ unifies with its ancestor $p(X)$ and produces infinite data $X = s(s(\cdots))$. Meanwhile, the SLD-derivation itself also computes towards the same infinite data as given by loop detection.

Example 4. Given $P_4 : q(s(X), L) \supset q(X, [X|L])$, the goal $q(0, L)$ gives rise to an infinite SLD-derivation $q(0, L) - q(s(0), L_1) - q(s(s(0)), L_2) - \cdots$. The infinite data $L = [0, s(0), s(s(0)), \ldots]$ is computed by the SLD-derivation but loop detection is not applicable.

The challenge is to extract finite patterns of *irregular* non-terminating SLD-derivations, which cannot be done by loop detection, as shown in Examples 2 and 4. Above all, within the wider mathematical and computing community, the concept of *proof by coinduction* itself is much less renowned and appears much more obscure than the concept of *proof by mathematical induction*, despite that a fair amount of work on coinduction has been done in some branches of theoretical computing [5].

Our ongoing work promises to solve both problems mentioned above. Particularly, in our proof-theoretic study of coinduction, we explain about what kind of logical reasoning constitutes coinductive reasoning, and in what sense a coinductively derived conclusion is no less valid than an inductively derived one. For instance, a coinductive proof of a statement H, given a set P of premises, is merely an ordinary logical reasoning (i.e. based on classical or intuitionistic logic) that derives H from the augmented set $P \cup \{H\}$ of premises, such that H itself is asserted as known truth and is used in its own proof. Indeed it seems like a circular proof with self-referencing, but this is done in a *guarded* way so that the asserted H is not used *directly* to prove itself. The consequence of such reasoning is the conclusion that H is *coinductively true* with respect to P, which means that H cannot be refuted based on P.

In a mathematical sense, an inductively proved statement F from P is in the *least fixed point* of P which collects all conclusions of P that has a finite inductive proof, while a coinductively proved statement H from P is in the *greatest fixed point* of P, which additionally has all conclusions of P such that no attempt of inductive proof can terminate. In Section 2 we review these fixed points in order to set up a necessary mathematical context for discussion of logical soundness of our proof-theoretic presentation of coinduction. We conclude and discuss related topics in Section 3.

2 Models for logic programs

We review the two coinductive models of logic programs. We assume readers' familiarity with syntax of first order Horn clause and the definition of terms and atoms as trees. We follow the terminology of [4].

A *first order Horn clause* (Horn clause, in short) K has the form

$$\forall x_1 \ldots x_m \quad A_1 \wedge \ldots \wedge A_n \supset A \quad (m, n \geq 0)$$

where A and variants thereof denote first order atoms. We denote the atom A by *head* K and we denote the set $\{A_1, \ldots, A_n\}$ by *body* K. A *logic program* is a finite set of Horn clauses.

Given a logic program P on signature Σ, we define the following sets. The *Herbrand universe*, denoted $\mathcal{H}$, is the set of all finite ground terms on Σ. The *Herbrand base*, denoted $\mathcal{B}$, is the set of all finite ground atoms on Σ. A *Herbrand interpretation* is any subset of $\mathcal{B}$. The *complete Herbrand universe*, denoted $\mathcal{H}'$, is the set of all finite and infinite ground terms on Σ. The *complete Herbrand base*, denoted $\mathcal{B}'$, is the set of all finite and infinite ground atoms on Σ. A *complete Herbrand interpretation* is any subset of $\mathcal{B}'$.

Given a Horn clause F, its *ground instance* on $\mathcal{H}$ is denoted $\lfloor F \rfloor$, and its ground instance on $\mathcal{H}'$ is denoted $\lfloor F \rfloor'$. Given a set S, its power set is denoted $Pow(S)$. There are two *immediate consequence operators* with respect to a program P, which are $\mathcal{T} : Pow\,(\mathcal{B}) \mapsto Pow\,(\mathcal{B})$ and $\mathcal{T}' : Pow\,(\mathcal{B}') \mapsto Pow\,(\mathcal{B}')$, defined respectively as:

$$\mathcal{T}(I) = \{ t \in \mathcal{B} \mid F \in P,\ head\ \lfloor F \rfloor = t,\ body\ \lfloor F \rfloor \subseteq I\ \}$$
$$\mathcal{T}'(J) = \{ s \in \mathcal{B}' \mid F \in P,\ head\ \lfloor F \rfloor' = s,\ body\ \lfloor F \rfloor' \subseteq J\ \}$$

Note that $\langle Pow(\mathcal{B}), \subseteq \rangle$ and $\langle Pow(\mathcal{B}'), \subseteq \rangle$ are complete lattices. It is also known that $\mathcal{T}$ and $\mathcal{T}'$ are increasing. Then, based on Knaster-Tarski fixed point theorem, each operator has a greatest fixed point (denoted $gfp(\mathcal{T})$ and $gfp(\mathcal{T}')$ respectively), which we take as *coinductive models* for a logic program P, as follows.

$$\mathcal{M} = gfp(\mathcal{T}) = \bigcup \{ I \mid I = \mathcal{T}(I) \} = \bigcup \{ I \mid I \subseteq \mathcal{T}(I) \}$$
$$\mathcal{M}' = gfp(\mathcal{T}') = \bigcup \{ I \mid I = \mathcal{T}'(I) \} = \bigcup \{ I \mid I \subseteq \mathcal{T}'(I) \}$$

We call $\mathcal{M}$ the *finite-term coinductive model*, and $\mathcal{M}'$ the *infinite-term coinductive model*. Note that $\mathcal{M} \subseteq \mathcal{M}'$ by definition, as $\mathcal{M}$ gathers all finite ground atoms that either has a finite proof or has an infinite derivation, while $\mathcal{M}'$ additionally contains all such, but infinite, atoms.

We could verify, regarding Examples 1–4, that results by infinite SLD-derivations and by loop detection (whenever applicable) are true with respect to coinductive models. Regarding Example 1,

$$\mathcal{M}(P_1) = \mathcal{M}'(P_1) = \{ r(a) \}$$

Obviously the result $r(a)$, which is from both loop detection and infinite derivation, is in the models. Regarding Example 2,

$$\mathcal{M}(P_2) = \{ p(a),\ p(s(a)),\ p(s(s(a))),\ \ldots \}$$
$$\mathcal{M}'(P_2) = \mathcal{M}(P_2) \cup \{ p(s(s(\cdots))) \}$$

where there is the result $p(a)$ by infinite derivation. Regarding Example 3,

$$\mathcal{M}(P_3) = \emptyset \qquad \mathcal{M}'(P_3) = \{ p(s(s(\cdots))) \}$$

We find the result of loop detection and infinite derivation in $\mathcal{M}'(P_3)$. Regarding Example 4,

$$\mathcal{M}(P_4) = \emptyset \qquad \mathcal{M}'(P_4) = \{ q(t, \geq t) \mid t \in \mathcal{H}'(P_4) \}$$

where we use $\geq t$ as a short hand for the infinite list $[t, s(t), s(s(t)), \ldots]$, and note that $\geq s(s(\cdots))$ is $[s(s(\cdots)), s(s(\cdots)), s(s(\cdots)), \ldots]$. The result $q(0, \geq 0)$ by infinite SLD-derivation is in this model. Note that $\mathcal{M}(P_3)$ and $\mathcal{M}(P_4)$ are empty, since with respect to P_3 and P_4, no finite atom has a finite proof or an infinite derivation whatsoever.

3 Conclusion and discussion

We work with the finite- and infinite-term coinductive models when studying coinductive Horn clauses. A coinductive reasoning scheme (such as infinite SLD-derivation and loop detection, and our current proof-theoretic study) for Horn clauses is *sound* if all statements provable by the scheme are true with respect to coinductive models.

Infinite SLD-derivations given in Examples 2 and 4, corresponding to programs P_2 and P_4 respectively, are not subject to loop detection. However, there still exist goals for these two programs, such that the goals give rise to infinite SLD-derivations to which loop detection is applicable. For instance, providing goal $p(X)$ for P_2, we have an infinite SLD-derivation $p(X) - p(s(X)) - p(s(s(X))) - \cdots$, and loop detection produces infinite data $X = s(s(\cdots))$ by unifying $p(s(X))$ and $p(X)$, and the result $p(s(s(\cdots)))$ is in $\mathcal{M}'(P_2)$; providing goal $q(X, L)$ for P_4, we have an infinite SLD-derivation $q(X, L) - q(s(X), L_1) - q(s(s(X)), L_2) - \cdots$ from which loop detection produces infinite data $X = s(s(\cdots))$ and $L = \geq s(s(\cdots))$ by unifying $q(X, L)$ with $q(s(X), L_1)$, and the result $q(s(s(\cdots)), \geq s(s(\cdots)))$ is in $\mathcal{M}'(P_4)$. Given a program, we define a *Gupta goal* as a goal that gives rise to an infinite SLD-derivation to which loop detection is applicable. We can see that for all four programs from Examples 1–4, there exist a Gupta goal. An interesting question to ask is, *Can we find a program, for which every non-terminating goal is not a Gupta goal* ? So far we did not find such a program.

The two decisive steps in a coinductive soundness proof is the construction of a post-fixed point candidate, and then the validation of the candidate. The paper [1] presents instructive techniques involved in automating the first step.

Acknowledgement

Thanks for Dr. Ekaterina Komendantskaya for commenting the draft paper, and for Prof. Stephen H. Muggleton for raising the question about Gupta goal.

References

[1] L. Dennis et al. Using a generalisation critic to find bisimulations for coinductive proofs. In *Automated Deduction—CADE-14*, pages 276–290. Springer Berlin Heidelberg, 1997.

[2] G. Gupta et al. Coinductive logic programming and its applications. In *ICALP'07*, pages 27–44, 2007.

[3] E. Komendantskaya and Y. Li. Productive corecursion in logic programming. *J. TPLP (ICLP'17 post-proc.)*, 17(5-6):906–923, 2017.

[4] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd edition, 1987.

[5] D. Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2011.