On Inductive and Coinductive Models of Logic Programs

František Farka
University of Dundee, and
University of St Andrews
ff32@st-andrews.ac.uk

1. INTRODUCTION

In the seminal work of Lloyd [3] both the declarative semantics and the operational semantics of logic programs is studied. A *logic program* P_Σ is a set of Horn clauses with a certain signature Σ. A *signature* consists of a set of function symbols, a set of predicate symbols. Of particular interest is the set of all finite ground terms with a signature Σ, the *Herbrand universe* $\mathbf{U}_\Sigma$ for Σ. Note that a ground term is a term not containing variables. The *Herbrand base* $\mathbf{B}_\Sigma$ for Σ is the set of all ground *atoms* that can be formed by application of a predicate symbol in Σ to terms in $\mathbf{U}_\Sigma$. An *interpretation* of a signature Σ is then a subset of $\mathbf{B}_\Sigma$. A *model* of a program P_Σ is an interpretation I of Σ that is closed to entailment under clauses in P_Σ. A standard result [3] is the *model intersection property* – an intersection of a set of models is a model. Therefore the intersection of the set of all models of P_Σ, this set being non-empty since $\mathbf{B}_\Sigma$ is a model of P_Σ, is a model of P_Σ. This model is called *the least Herbrand model* $\mathcal{M}_P$ of a program P.

The operational (big-step) semantics of a program P is usually [5] given by the following inference rule:

$$\frac{P \vDash \sigma B_1, \dots P \vDash \sigma B_n}{P \vDash \sigma A} \tag{1}$$

where $A \leftarrow B_1, \dots B_n$ is a clause in P and σ is a grounding substitution. Note that a *substitution* is a total function from variables to terms and it is *grounding* if it is from variables to ground terms. *Application* of a substitution σ to a term t, denoted σt substitutes all variables v in t by $\sigma(v)$. Application is naturally extended to atoms. There is an operator $\mathcal{T}_P : \mathfrak{P}(\mathbf{B}_\Sigma) \to \mathfrak{P}(\mathbf{B}_\Sigma)$, where $\mathfrak{P}$ denotes a power set, that is assigned to a program P_Σ. This operator is defined by $I \mapsto J$ where J are all the terms that can be obtained from I by the above inference rule 1. This operator links declarative and operational semantics of logic programs: for a program P it holds that $lfp(\mathcal{T}_P) = \mathcal{M}_P$ where $lfp(\mathcal{T}_P)$ denotes the least fixed point of the operator $\mathcal{T}_P$.

The above semantics of finite computations can be carried out in a similar manner for possibly non-terminating computations [3]. In particular, the *complete Herbrand universe* $\mathbf{U}'_\Sigma$ is the set of all (possibly infinite) terms with a signature Σ. The notions of the *complete Herbrand base* $\mathbf{B}'_\Sigma$, of a *complete interpretation*, and of a *complete model* are defined accordingly to their above counterparts. A variant of model intersection property still holds—an intersection of a set of complete models is a complete model. Yet again, the complete Herbrand base $\mathbf{B}'_\Sigma$ is a complete model of a program P_Σ and the intersection of all complete models of P is the *least complete Herbrand model* $\mathcal{M}'_P$ of a program P. There is also an appropriate variant of the complete operator $\mathcal{T}'_P : \mathfrak{P}(\mathbf{B}'_\Sigma) \to \mathfrak{P}(\mathbf{B}'_\Sigma)$ that allows to characterise the least complete Herbrand model as a fixed point of this operator: $gfp(\mathcal{T}'_P) = \mathcal{M}'_P$, where $gfp(\mathcal{T}'_P)$ denotes the greatest fixed point.

We can observe a discrepancy here on the ordering of models. The least Herbrand model is the least fixed point of the appropriate operator but the least complete Herbrand model is the greatest fixed point of the appropriate operator. However, the least fixed point $lft(\mathcal{T}'_P)$ of an operator $\mathcal{T}'_P$ coincides with the least Herbrand model $\mathcal{M}_P$ and the greatest complete Herbrand model is the complete Herbrand base B'_P that is a trivial model and therefore not of any interest. Furthermore, the operational semantics does not require a distinction between predicate and function symbols and the operator $\mathcal{T}_P$ can be defined (*cf.* Komendantskaya and Johann [2]) on subsets of the Herbrand universe $\mathbf{U}_\Sigma$ (similarly for $\mathcal{T}'_P$). In the following text we describe such simplified models in the language of category theory. In particular our contributions are: a formulation of operational semantics of as initial algebras and terminal coalgebras that works on term interpretations, and definition of coinductive models of logic programs such that the operational and the declarative semantics are symmetrical.

2. OPERATIONAL SEMANTICS AS ALGEBRAS AND COALGEBRAS

We describe operational semantics of logic programs in language of category theory. We assume that a reader is familiar with the basic concepts as category, isomorphism, functor, algebra and coalgebra, and initial and terminal object. First we establish category that models interpretations of a certain signature:

DEFINITION 1. *The category of* **ΣInt** *of interpretations of a signature* Σ *has*

objects I, $I \subseteq \mathbf{U}'_\Sigma$ *are (possibly complete) interpretations of the signature* Σ, *and*

arrows $I \to J$, *whenever* $I \subseteq J$.

Note that both Herbrand interpretations and complete Herbrand interpretations are objects of our model category. Later on this will allow us to provide a uniform description of the least and the greatest Herbrand model. As in the previous section a program P with a signature Σ gives rise to an operator on interpretations of the signature Σ:

DEFINITION 2. *The operator* $\mathcal{T}_P : \mathbf{\Sigma Int} \to \mathbf{\Sigma Int}$ *is defined as follows:*

- *for* I *an interpretation,*

$$\mathcal{T}_P(I) = \{A\theta \in \mathbf{U}'_\Sigma \mid \exists A \leftarrow B_1, \ldots B_n \in P, \exists\{B'_1, \ldots B'_n\} \subseteq I\}$$

  where θ *is a substitution, and*

- *for* $f : I \to J$ *let* $\mathcal{T}_P(f)$ *be the morphism* $\mathcal{T}_P(I) \to \mathcal{T}_P(J)$ *if and only if this morphism exists and is unique.*

PROPOSITION 1. *The operator* $\mathcal{T}_P : \mathbf{\Sigma Int} \to \mathbf{\Sigma Int}$ *is a functor on* $\mathbf{\Sigma Int}$.

The understanding of the operator $\mathcal{T}_P$ as a functor on the category $\mathbf{\Sigma Int}$ captures the same operational behaviour as the inference rule 1 in the previous section. However, the above functor encompasses both the original operator $\mathcal{T}_P$ on Herbrand interpretations and the operator $\mathcal{T}'_P$ on complete Herbrand interpretations and allows us to draw a direct relation to initial algebras and terminal coalgebras as these notions are usually preferred to a study of fixed points [1]. In order to do so it is convenient to have an explicit notion of a *fixed point* of a functor: a fixed point of a functor $F : \mathcal{C} \to \mathcal{C}$ is a pair (X, v) such that X is an object of $\mathcal{C}$ and $v : FX \to X$ is an isomorphism. Then the *least* fixed point is the initial object among all the fixed points and the *greatest* fixed point is the terminal object. The following variant of Lambek's lemma provides the desired relation:

LEMMA 1 (LAMBEK). *Let* (A, f) *be an initial* F*-algebra of a functor* $F : \mathcal{C} \to \mathcal{C}$. *Then* (A, f) *is a fixed point of* F *and it is the least such fixed point.*

Let (B, g) *be a terminal* F*-coalgebra of a functor* $F : \mathcal{C} \to \mathcal{C}$. *Then* (B, f^{-1}) *is a fixed point of* F *and it is the greatest such fixed point.*

The following propositions state existence of fixed points of the $\mathcal{T}_P$ functor:

PROPOSITION 2. *Let* P *be a program. Then there is an initial* $\mathcal{T}_P$*-algebra, and a terminal* $\mathcal{T}_P$*-coalgebra.*

COROLLARY 1. *The least and the greatest fixed point of* $\mathcal{T}_P$ *exist.*

3. INDUCTIVE AND COINDUCTIVE MODELS

In this section we restore symmetry between declarative models of logic programs and operational semantics. Recall that the standard definition [3] of model of a program P is as an Herbrand interpretation of P that is closed to entailment under clauses in P. We restate this definition in concordance with the previous section:

DEFINITION 3 (INDUCTIVE MODEL). *Let* Σ *be a signature and* $I \subseteq \mathbf{U}'_\Sigma$. I *is an* inductive model *of a program* P_Σ *if and only if* $\forall\{B'_1, \ldots B'_n\} \subseteq I$ *and* $\forall\sigma$ *a substitution if* $\exists A \leftarrow B_1, \ldots, B_n \in P$ *s.t.* $\sigma B_1 = B'_1, \ldots, \sigma B_n = B'_n$ *then* $\sigma A \in I$.

Note that an Herbrand model of a program P is also an inductive model of P and that inductive models posses the standard model intersection property [3]:

PROPOSITION 3. *Let* P *be a program and* $\{M_i\}_{i\in I}$ *be a non-empty set of inductive models of* P. *Then* $\bigcap_{i\in I} M_i$ *is an inductive model.*

The set of all terms $\mathbf{U}'_\Sigma$ in an inductive model of P_Σ and the set of all models of P_Σ is non-empty. Hence the intersection of all inductive models of P_Σ is a model, the *least inductive model* of P_Σ. However, we dualise the standard definition of a complete model:

DEFINITION 4 (COINDUCTIVE MODEL). *Let* Σ *be a signature and* $I \subseteq \mathbf{U}'_\Sigma$. I *is a* coinductive model *of a program* P_Σ *if and only if* $\forall A' \in I$ $\exists\sigma$ *a substitution s.t.* $\exists A \leftarrow B_1, \ldots, B_n$ *if* $\sigma A = A'$ *then* $\{\sigma B_1, \ldots, \sigma B_n\} \subseteq I$.

For coinductive models, we get a dual counterpart to model intersection property:

PROPOSITION 4 (UNION PROPERTY). *Let* P *be a program and* $\{M_i\}_{i\in I}$ *be a non-empty set of coinductive models of* P. *Then* $\bigcup_{i\in I} M_i$ *is a coinductive model.*

Since every program P_Σ has the empty set as a coinductive model the set of all coinductive models of P_Σ is non-empty. Hence the union of all coinductive models of P_Σ is a coinductive model, the *greatest coinductive model* of P_Σ.

PROPOSITION 5. *Let* P *be a program*

- *If* (A, f) *is an initial* $\mathcal{T}_P$*-algebra then* A *is the least inductive model* $\mathcal{M}_P$ *of* P.
- *If* (B, g) *is a terminal* $\mathcal{T}_P$*-coalgebra then* B *is the greatest coinductive model* $\mathcal{M}'_P$ *of* P.

The above proposition justifies our definitions of models as *inductive* and *coinductive* (cf. Backhouse *et al.* [1]).

4. INDUCTIVE-COINDUCTIVE MODELS

Inductive and the coinductive models in the previous section do not characterise interleaved inductive-coiunductive terms well-enough. Consider the following program:

```
forest(nil).
forest(fcons(X, Y)) ← tree(X), forest(Y).

tree(X) ← forest(X).
```

The least inductive model of this program contains all finitely branching finite trees and the greatest model contains possibly infinitely branching trees that can also have infinite branches. However, assume that the intended interpretation is the set of finitely branching possibly infinite trees. This interpretation can be expressed by anotating function symbols of the program as either inductive or coinductive:

DEFINITION 5 (MARKING). *Let* Σ *be a signature of a program. A* marking *of a program is a function* $m : \Sigma \to \{\mu, \nu\}$.

We understand the symnbol μ to mark function symbols with inductive intended interpretation and the symbol ν to mark function symdols with coinductive intended interpretation. Now we can refine our definitions of models from the previous section:

Definition 6 (Inductive model). *Let Σ be a signature, m be a marking of Σ, $p \in \Sigma$ a symbol, P_Σ a program, and $I \subseteq \mathbf{U}'_\Sigma$. I is an* inductive model *of p if and only if $\forall\{B'_1, \dots B'_n\} \subseteq I$ and $\forall\sigma$ a substitution if $\exists p(A_1, \dots, A_m) \leftarrow B_1, \dots, B_n \in P$ s.t. $\sigma B_1 = B'_1, \dots, \sigma B_n = B'_n$ then $p(\sigma A_1, \dots, A_m) \in I$.*

For a program with inductive symbols only, an interpretation that is inductive model of all symbols in signature is an inductive model in the sense of the previous section.

Definition 7 (Coinductive model). *Let Σ be a signature, $q \in \Sigma$ a symbol, and $I \subseteq \mathbf{U}'_\Sigma$. I is a* coinductive model *of q given a program P_Σ if and only if $\forall q(A'_1, \dots, A'_m) \in I$ $\exists\sigma$ a substitution s.t. $\exists q(A_1, \dots, A_m) \leftarrow B_1, \dots, B_n$ if $\forall i \in 1..m$ $\sigma A_i = A'_i$ then $\{\sigma B_1, \dots, \sigma B_n\} \subseteq I$.*

Yet again, an interpretation of a program with coinductive symbols only that is a coinductive model of all symbols in the signature is a coinductive model in the sense of the previous section. Having a marking where some symbols are inductive and other are coinductive gives rise to a notion of a mixed model:

Definition 8 (Inductive-coinductive model). *Let Σ be a signature and m be a marking of Σ. An interpretation $I \subset \mathbf{U}'_\Sigma$ is an* inductive-coinductive *model of a program P_Σ if and only if it is an inductive model of all the inductive symbols in Σ given P_Σ and it is a coinductive model of all the coinductive symbols in Σ given P_Σ.*

Marking of symbols in a signature gives rise to two projections from an interpretation: $\mu : \mathbf{U}'_\Sigma \to \mathbf{U}'_\Sigma$ is defined as $I \mapsto \{t(t_1, \dots, t_m) \in I \mid m(t) = \mu\}$ and $\nu : \mathbf{U}'_\Sigma \to \mathbf{U}'_\Sigma$ is defined as $I \mapsto \{t(t_1, \dots, t_m) \in I \mid m(t) = \nu\}$. Since for any interpretation I the sets $\mu(I)$ and $\nu(I)$ are disjoint the pair $\langle \mu(I), \nu(I) \rangle$ is isomorphic to I and we understand these by an abuse of notation identical. The above projections allow us to define two different orders on inductive-coiunductive models; the order $\preceq_\mu$ is defined as follows: for inductive-coinductive models M_1, M_2, $M_1 \preceq_\mu M_2$ iff $\nu(M_1) = \nu(M_2)$ and $\mu(M_1) \subseteq \mu(M_2)$. The order $\preceq_\nu$ is defined symmetricaly: for inductive-coinductive models M_1, M_2, $M_1 \preceq_\nu M_2$ iff $\mu(M_1) = \mu(M_2)$ and $\nu(M_1) \subseteq \nu(M_2)$.

Inductive-coinductive models posses a generalisation of the intersection property:

Lemma 2. *Let P be a program and $\{M_i\}_{i \in I}$ be a non-empty set of inductive-coinductive models of P such that $\exists N \forall i \in I, \nu(M_i) = N$. Then $\langle \bigcap_{i \in I} M_i, N \rangle$ is an inductive-coinductive model.*

If for a set $N \subseteq \mathbf{U}'_\Sigma$ if the set of all inductive-coinductive models with $\nu(M_i) = N$ is non-empty than the model $\langle \bigcap_{i \in I} M_i, N \rangle$ is a minimal model in the order $\preceq_\mu$. We say that such a model is the *minimal* inductive-coinductive model for N.

There is also a generalisation of the union property:

Lemma 3. *Let P be a program and $\{M_i\}_{i \in I}$ be a non-empty set of inductive-coinductive models of P such that $\exists N \forall i \in I, \mu(M_i) = N$. Then $N \times \bigcup_{i \in I} M_i$ is an inductive-coinductive model.*

If for a set $N \subseteq \mathbf{U}'_\Sigma$ the set $\{M_i\}_{i \in I}$ of all inductive-coinductive models with $\mu(M_i) = N$ is non-empty then the model $\langle N, \bigcup_{i \in I} M_i \rangle$ is a maximal model in the order $\preceq_\nu$. We say that such a model is the *maximal* inductive-coinductive model for N.

Definition 9 (MM-IC model). *Let Σ be a signature, m a marking of Σ and $I \subseteq \mathbf{U}'_\Sigma$. I is* minimal-maximal inductive-coinductive *(mm-ic) model if and only if I is the minimal inductive model for $\nu(I)$ and it is the maximal coinductive model for $\mu(I)$.*

Mm-ic models do not posses neither pairwise intersection nor pairwise union property. However, the following lemmata hold:

Lemma 4. *Let P be a program and $\{M_i\}_{i \in I}$ be a non-empty set of all mm-ic models of P. Then $\bigcap_{i \in I} M_i$ is an mm-ic model.*

We denote this model if it exists $\mathcal{M}_P$.

Lemma 5. *Let P be a program and $\{M_i\}_{i \in I}$ be a non-empty set of all mm-ic models of P. Then $\bigcup_{i \in I} M_i$ is an mm-ic model.*

We denote this model if it exists $\mathcal{M}'_P$. The following propostion leads to our final result:

Proposition 6. *Let P be a program. There is an mm-ic model of P.*

Corollary 2. *Let P be a program. The models $\mathcal{M}_P$ and $\mathcal{M}'_P$ exist.*

5. ALGEBRAS AND COALGEBRAS?

6. CONCLUSION

We have described operational semantics of logic programs as initial algebras and terminal coalgebras on term interpretations. Up to our knowledge, there is no such description in the existing literature. We have also restated declarative semantics such that models in declarative semantics are symmetrical to fixed points of operational semantics.

7. RELATED AND FUTURE WORK

The study of algebras and coalgebras in computer science is well established and many constructs in the theory of programming languages are described as algebras and coalgebras, e.g. algebraic data types [1, Chap. 5], or W-types and M-types (*e.g.* [4, 6]). We believe that our description in this uniform way allows for easier application of methods of logic programming in type inference algorithms for functional programming languages. The desired next step is to find a description of interleaved inductive-coinductive models.

8. REFERENCES

[1] R. C. Backhouse, R. L. Crole, and J. Gibbons, editors. *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction, International Summer School and Workshop, Oxford, UK, April 10-14, 2000, Revised Lectures*, volume 2297 of *Lecture Notes in Computer Science*. Springer, 2002.

[2] E. Komendantskaya and P. Johann. Structural resolution: a framework for coinductive proof search and proof construction in horn clause logic. *ACM Transcations in Computational Logic*, submitted, 2015.

[3] J. W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.
[4] I. Moerdijk and E. Palmgren. Wellfounded trees in categories. *Ann. Pure Appl. Logic*, 104(1-3):189–218, 2000.
[5] D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4):15:1–15:41, May 2009.
[6] B. van den Berg and F. D. Marchi. Non-well-founded trees in categories. *Ann. Pure Appl. Logic*, 146(1):40–59, 2007.

APPENDIX

A. EXAMPLES

EXAMPLE 1. *Let P be the following program:*

```
nat(z).
nat(s(X)) ← nat(X).
```

These are inductive models of P:

- $M_1 = \{s^i(z) \mid i \in \mathbb{N}\}$
- $M_2 = M_1 \cup \{s^\omega\}$

EXAMPLE 2. *Let P be the following program:*

```
stream(a, cons(a)) ← stream(X)
```

These are coinductive models of P:

- $M_1 = \emptyset$
- $M_2 = \{stream(a, \nu X.cons(a, X))\}$

EXAMPLE 3. *Let P be the following program:*

```
p(a) ← p(a).
q(z, a).
q(s(X), Y) ← q(X, Y), p(Y).
```

Inductive models:

- $M_1 = \{q(z, a)\}$
- $M_2 = \{q(z, a), p(a)\} \cup \{q(s^i(z), a) \mid i \in \mathbb{N}\}$
- $M_3 = \{q(z, a), p(a)\} \cup \{q(s^i(z), a) \mid i \in \mathbb{N}\} \cup \{p(s^\omega, a)\}$

Coinductive models:

- $M_4 = \emptyset$
- M_1, M_2, M_3

Desired ($p \mapsto \mu, q \mapsto \nu$):

- $M_2 = \{q(z, a), p(a)\} \cup \{q(s^i(z), a) \mid i \in \mathbb{N}\}$

And for inductive-coinductive models:

EXAMPLE 4. *Let P be the following program:*

```
ind(z, z).
ind(s(X), Y)) ← ind(X, X), coi(Y, Y).
coi(X, Y) ← ind(X, Y), coi(X).
coi(X) ← coi(X).
```

with a marking $ind \mapsto \mu$, $coind \mapsto \nu$.

Inductive-coinductive models:

- $M_1 = \langle \{ind(z, z)\}, \emptyset \rangle$
- $M_2 = M_1 \cup \langle \{ind(s^i(z), z) \mid i \in I \subseteq \mathbb{N}\}, \emptyset \rangle$
- $M_2 = \langle \{ind(z, z)\}, \{coi(z)\} \rangle$