Models of Inductive-Coinductive Logic Programs

Henning Basold
Radboud University
CWI Amsterdam
henning@basold.eu

Ekaterina Komendantskaya
Heriot-Watt University
komendantskaya@gmail.com

Traditionally, logic programs have been used to describe relations between finite terms, a fact most prominently reflected in the notion of the least Herbrand model [7, 5]. A typical example is the following logic program, which generates the natural numbers.

$$\begin{aligned} 1: &\quad \mathrm{nat}(0) \longleftarrow \\ 2: &\quad \mathrm{nat}(\mathrm{s}(x)) \longleftarrow \mathrm{nat}(x) \end{aligned} \tag{1}$$

Later [1, 2, 3, 4, 6], also coinductive interpretations of logic programs have been proposed. Under a coinductive interpretation, the program given in (1) generates an extra element s^{ω}. The thus obtained interpretation of nat corresponds to the completion of the natural numbers with a point at infinity. Another example, which is non-trivial only under a coinductive interpretation, are streams over natural numbers:

$$3: \quad \mathrm{str}(\mathrm{cons}(x,y)) \longleftarrow \mathrm{nat}(x), \mathrm{str}(y) \tag{2}$$

Note, however, that a purely coinductive model also contains elements of the form $\mathrm{cons}(\mathrm{s}^{\omega}, t)$ for some (infinite) stream term t. To rule out such spurious terms we would have to interpret nat inductively and str coinductively, something that has not been studied so far.

In the following we will augment logic programs with a function par that assigns to each relation symbol its *parity*, which can be either μ or ν. The parity expresses whether a relation is supposed to be interpreted inductively or coinductively. For example, to obtain the intended interpretation of the clauses in (1) and (2), we would define $\mathrm{par}(\mathrm{nat}) = \mu$ and $\mathrm{par}(\mathrm{str}) = \nu$.

Another, perhaps more interesting, example is the sub-stream relation sub that relates streams s and t if all values of s appear in order in t. We can express the sub-stream relation as logic program by using a helper relation sub_{μ}. The relation sub_{μ} tries to match the head of s with a value in t. An inductive interpretation of sub_{μ} enforces then that the head of s must be found in t after finitely many steps.

$$\begin{aligned} &\mathrm{par}(\mathrm{sub}) = \nu \qquad \mathrm{par}(\mathrm{sub}_{\mu}) = \mu \\ 4: &\quad \mathrm{sub}(x,y) \longleftarrow \mathrm{sub}_{\mu}(x,y) \\ 5: &\quad \mathrm{sub}_{\mu}(\mathrm{cons}(n,x), \mathrm{cons}(n,y)) \longleftarrow \mathrm{nat}(n), \mathrm{sub}(x,y) \\ 6: &\quad \mathrm{sub}_{\mu}(x, \mathrm{cons}(n,y)) \longleftarrow \mathrm{nat}(n), \mathrm{sub}_{\mu}(x,y) \end{aligned} \tag{3}$$

It should be noted that the relation sub is interpreted as the full relation in a purely coinductive model because in such a model, the search of sub_{μ} does not have to terminate.

A first step towards understanding inductive-coinductive logic programs is understanding their denotational models. Here it is important that logic programs Φ are given by means of two signatures Σ_{Φ} and Δ_{Φ}. The signature Σ_{Φ} contains thereby the symbols that are used in the terms, like s and 0 in the examples above. On the other hand, Δ_{Φ} specifies the relation symbols used in Φ. We will denote the arity of a symbol $f \in \Sigma_{\Phi}$ by $\mathrm{ar}\, f$ and similarly for symbols in Δ_{Φ}. A *(term) model* $\mathscr{M}$ for Φ is then required

Submitted to:
CoALP-Ty 2016

to provide interpretations of the relation symbols in Δ_Φ as relations between possibly infinite terms over Σ_Φ. Moreover, for inductive relation symbols their interpretation in $\mathscr{M}$ needs to be forward closed under the clauses of Φ, whereas the interpretation of coinductive relations must be backwards closed.

Let us make this more precise. Suppose we are given signatures Σ and Δ, and a set V of variables. Let $\Sigma^*(V)$ be the set of terms over V, and Σ^∞ be the set of possibly infinite ground terms over Σ. A *formula* φ is given by $Q(\overrightarrow{t})$ for some $Q \in \Delta$ and a tuple $\overrightarrow{t} = (t_1, \dots, t_{\mathrm{ar}\,Q})$ of terms in $\Sigma^*(V)$. We call a finite set of formulas a *sentence*. Finally, a *(Horn) clause* is a pair of a sentence S and a formula φ, denoted by $\varphi \longleftarrow S$. A *logic program* Φ consists of signatures Σ, Δ, a map $\mathrm{par}\colon \Delta \to \{\mu, \nu\}$ and a set of clauses.

Using this setup, we now characterise models for logic programs Φ. First, we associate to Φ a map $\widehat{\Phi}\colon \prod_{Q\in\Delta} \mathrm{Rel}_{\mathrm{ar}\,Q}(\Sigma^\infty_\Phi) \to \prod_{Q\in\Delta} \mathrm{Rel}_{\mathrm{ar}\,Q}(\Sigma^\infty_\Phi)$ by

$$\widehat{\Phi}(F)(Q) := \bigcup_{\substack{\phi \longleftarrow S \in \Phi \\ \phi = Q(\overrightarrow{t})}} \left\{ \overrightarrow{t}[\sigma] \;\middle|\; \sigma\colon V \to \Sigma^\infty_\Phi \text{ and } \forall P(\overrightarrow{s}) \in S.\ \overrightarrow{s}[\sigma] \in F(P) \right\},$$

where $\overrightarrow{t}[\sigma]$ denotes the substitution of σ into all terms in $\overrightarrow{t}$. Moreover, we define components

$$\widehat{\Phi}_\rho : \prod_{Q\in\Delta} \mathrm{Rel}_{\mathrm{ar}\,Q}(\Sigma^\infty) \to \prod_{Q\in\Delta_\rho} \mathrm{Rel}_{\mathrm{ar}\,Q}(\Sigma^\infty)$$

of $\widehat{\Phi}$ by restriction: $\widehat{\Phi}_\rho(Q) := \widehat{\Phi}(Q)|_{\Delta_\rho}$, where $\Delta_\rho := \mathrm{par}^{-1}(\rho)$. A Φ-*model* $\mathscr{M}$ is given by a map $\mathscr{M} \in \prod_{Q\in\Delta} \mathrm{Rel}_{\mathrm{ar}\,Q}(\Sigma^\infty)$, such that

$$\widehat{\Phi}_\mu(\mathscr{M}) \sqsubseteq \mathscr{M}_\mu \quad \text{and} \quad \mathscr{M}_\nu \sqsubseteq \widehat{\Phi}_\nu(\mathscr{M}),$$

where $\sqsubseteq$ is point-wise inclusion and $\mathscr{M}_\rho$ is the restriction $\mathscr{M}|_{\Delta_\rho}$. This encodes precisely the forward and backwards closure conditions.

This characterisation of models in terms of a monotone operator enables us to construct a fixed point model for Φ. This in turn gives us a universe of discourse for exploring other semantics and proof systems for mixed inductive-coinductive logic programs.

In the talk, we will discuss properties of the fixed point model. We will further discuss *standard models*, in which the interpretation of inductive relations is restricted. This allows us to prove weak completeness of the fixed point model with respect to standard models.

References

[1] M. A. Nait Abdallah (1984): *On the Interpretation of Infinite Computations in Logic Programming*. In: *ICALP*, *Lecture Notes in Computer Science* 172, Springer, pp. 358–370, doi:10.1007/3-540-13345-3_32.

[2] Mathieu Jaume (2000): *Logic Programming and Co-Inductive Definitions*. In: *CSL*, *Lecture Notes in Computer Science* 1862, Springer, pp. 343–355, doi:10.1007/3-540-44622-2_23.

[3] Mathieu Jaume (2002): *On Greatest Fixpoint Semantics of Logic Programming*. *J. Log. Comput.* 12(2), pp. 321–342, doi:10.1093/logcom/12.2.321.

[4] Ekaterina Komendantskaya & John Power (2011): *Coalgebraic Semantics for Derivations in Logic Programming*. In: *CALCO*, *LNCS* 6859, Springer, pp. 268–282, doi:10.1007/978-3-642-22944-2_19.

[5] John W. Lloyd (1987): *Foundations of Logic Programming, 2nd Edition*. Springer.

[6] Danny De Schreye & Stefaan Decorte (1994): *Termination of Logic Programs: The Never-Ending Story*. *J. Log. Program.* 19/20, pp. 199–260, doi:10.1016/0743-1066(94)90027-2.

[7] M. H. Van Emden & R. A. Kowalski (1976): *The Semantics of Predicate Logic As a Programming Language*. *J. ACM* 23(4), pp. 733–742, doi:10.1145/321978.321991.