Abstract. We extend logic programming’s semantics with the semantic dual of traditional Herbrand semantics by using greatest fixed-points in place of least fixed-points. Executing a logic program then involves using coinduction to check inclusion in the greatest fixed-point. The resulting coinductive logic programming language is syntactically identical to, yet semantically subsumes logic programming with rational terms and lazy evaluation. We present a novel formal operational semantics that is based on synthesizing a coinductive hypothesis for this coinductive logic programming language. We prove that this new operational semantics is equivalent to the declarative semantics. Our operational semantics lends itself to an elegant and efficient goal directed proof search in the presence of rational terms and proofs. We describe a prototype implementation of this operational semantics along with applications of coinductive logic programming.

But look! What was that? One of the snakes had seized hold of its own tail, and the form whirled mockingly before my eyes.

—Friedrich A. Kekule, 1864

1 Introduction

The traditional declarative and operational semantics for logic programming (LP) is inadequate for various programming practices such as programming with infinite data structures and corecursion [2]. While such programs are theoretically interesting, their practical applications include improved modularization of programs as seen in lazy functional programming languages, rational terms, and applications to model checking as discussed in section 5. For example, we would like programs such as the following program, which describes infinite binary streams, to be semantically meaningful and finitely derivable.

\[
\text{bit}(0).
\]
\[
\text{bit}(1).
\]
\[
\text{bitstream}([H \mid T]) :- \text{bit}(H), \text{bitstream}(T).
\]
\[
\mid ?- X = [0, 1, 1, 0 \mid X], \text{bitstream}(X).
\]

We would like the above query to return a positive answer in finite time, however, aside from the \text{bit} predicate, the least fixed-point (lfp) semantics of the above
program is null, and no finite SLD derivation exists for the query. Hence the problems are two-fold. The Herbrand universe does not allow for infinite terms such as \( X \) and the least Herbrand model does not allow for infinite proofs, such as the proof of \( \text{bitstream}(X) \). However, the traditional declarative semantics of LP can be extended in order to give declarative semantics to such infinite structures and properties, as seen in numerous accounts of rational terms and infinite derivations [14,10,12,11]. Furthermore, the operational semantics must be extended, so as to be able to finitely represent an otherwise infinite derivation. This paper proposes such an operational semantics which is based on synthesizing a coinductive hypothesis, and discusses its implementation and applications. We refer to this variation of logic programming as “coinductive logic programming” [2]. The novel contribution of our work is the development of an efficient top-down operational semantic for computing the greatest fixed-point of a logic program.

2 Syntax and Semantics

Traditionally, declarative semantics for LP has been given using the notions of Herbrand universe, Herbrand base, and minimal model [12]. Each is defined as a least fixed-point, and the set is manifested in traditional set theory. The declarative semantics of coinductive LP, on the other hand, takes the dual of each of these notions, in hyperset theory with the \textit{axiom of plenitude} [2]. This variation of the declarative semantics of a logic program has appeared before [14,10,12,11] in order to describe rational trees and infinite SLD derivations. However, here it is used to finitely describe potentially infinite derivations in our new operational semantics, which we call co-SLD in section 2.4.

2.1 Induction and Coinduction

A naive attempt to prove a property of the natural numbers involves demonstrating the property for 0, 1, 2,…. In order for such a proof to be comprehensive, it must be infinite. However, since an explicitly infinite proof cannot be written, the principle of proof by induction can be used to represent such an infinite proof in a finite form. This is precisely what the operational semantics of coinductive LP does as well. That is, coinductive LP uses the principle of proof by coinduction for representing infinite proofs or derivations in a finite form. The difference between induction and coinduction will be made more obvious later.

Following the account given in Barwise [2] and Pierce [15], we briefly review the set theoretic notions of induction and coinduction, which are defined in terms of monotonic functions on sets and least and greatest fixed-points, which

---

1 Note that coinductive LP defined in this paper is not at all related to \textit{inductive} LP which is the common term used to refer to LP systems for learning rules. In fact, sometimes we’ll use the term inductive LP itself to refer to traditional SLD (or OLDT) resolution-based LP.