On the Epistemological foundations of Logic Programming and its Extensions

Marc Denecker
Department of Computer Science, K.U.Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium.
Phone: +32 16 327555 — Fax: +32 16 327996
email: marcd@cs.kuleuven.ac.be

Extended Abstract

One of the major roles of logic in computer science and A.I. is for knowledge representation and specification. In the context of knowledge representation, formal logic is widely praised as the tool par excellence for expressing information in a precise way and for correct and precise communication of information between different experts. In the scenario of communicating experts, the experts express their knowledge about some external problem domain in logic specifications which subsequently are communicated to other experts. In the computational logic approach to AI, computers participate in this and assist the human experts by solving computational problems using the logic specifications. The use of formal logic helps to avoid the ambiguities and lacunas that appear frequently in natural language specifications and, in a absence of computer systems that understand natural language, is the most promising way of transferring knowledge to a computer system.

Obviously, when different human experts communicate their knowledge about some *external* problem domain via logical specifications, correct communication is only possible to the extent that they *interpret* the logical formulas in the same way about this problem domain. They should agree on what the formulas of the specification say about the problem domain. The way human experts interpret the logic formulas of a formal logic in the problem world is what logicians call the *declarative reading* of the logic. One of the fundamental prexassumptions underlying the use of logic for knowledge representation, specification and problem solving is that *a formal logic has a well-understood and objective declarative reading.*

The epistemological study of a logic aims to clarify its declarative reading. A standard technique to do so is by explaining in what states of the world the formulas of the logic are *true*. The states of the world in which a given formula is true represent the *possible states* of the world. An epistemological theory that expresses when logical statements are true is what logicians also call a *truth conditional semantics* (e.g. (Moore 1995), p. 3).

The goal of this study is to investigate the declarative reading of logic programming (LP) and its extensions: stable logic programming, answer set programming and abductive logic programming.

One of the central goals of logic programming was and still is to combine the advantages of formal *declarative* logic and *procedural languages* (Kowalski 1974). Originally, a definite logic program was seen as a classical logic Horn theory. The use of SLD-resolution induced the procedural interpretation upon logic programs. When the negation as failure inference rule was introduced in Prolog systems, the view of logic programs as classical logic implications broke down. It was realized that the soundness of this rule could not be justified on the basis of the classical logic semantics of a logic program. Important efforts were made to propose alternative formal semantics for logic programming for which negation as failure is sound. This resulted in a number of different semantics of which the completion semantics (Clark 1978), the stable semantics (Gelfond & Lifschitz 1988) and the well-founded semantics (Van Gelder, Ross, & Schlipf 1991) are the most important ones. An important tool for defining these semantics was the use of embeddings of logic programming in other logics, in particular in classical logic (CL), default logic (DL) and autoepistemic logic (AEL).

The question investigated here is what these formal semantics say about the declarative reading of logic programs. One view that appeared very early on is that logic programs represent definitions; recursive programs represent inductive definitions. This view is appealing in the context of many prototypical logic programs:

```
member(X,[X|Y]).
member(X,[Z,L]) :- member(X,L).

append([],L,L).
append([H|T],L,[H|T2]) :- append(T,L,T2)

trans(X,Y) :- p(X,Y).
trans(X,Z) :- tr(X,Y), tr(Y,Z).
```

Each of these Horn programs can be seen as the natural representation of an inductive definition. The least model semantics of (van Emden & Kowalski 1976) formalises the view of Horn programs as inductive definitions. In (Denecker 1998), I pointed to the strong relationships between Horn programs under least model semantics and standard formalisations of monotone inductive definitions such as (Moschovakis 1974) and (Aczel 1977).

The declarative reading of logic programs as (inductive) definitions is often natural in the context of programs with negation as well:

```
child(X) :- person(X),not adult(X).

even(0).
even(s(N)) :- not even(N).
```

For example, the rules defining *even* represent the inductive definition:

- 0 is even.
- if n is not even, then $n+1$ is even, otherwise $n+1$ is not even.

It is well-known that the least model semantics does not extend to logic programs with negation. In (Clark 1978), Clark proposed an alternative approach to formalise the same intuition. He maps a logic program to its completion, the set of *completed definitions.* A completed definition is the way (non-recursive) definitions are expressed in first order logic. For example, the completion of the program

```
p :-  q, not r
```

is the CL theory:

$$\{q \leftrightarrow false \;\;,\;\; r \leftrightarrow false \;\;,\;\; p \leftrightarrow q \wedge \neg r\}$$

Unfortunately, Clark completion semantics is in general too weak to formalise the declarative reading of definite programs as inductive definitions. It accepts unintended models in programs with positive loops such as the transitive closure program[1]. Recently, in (Denecker 1998), I pointed to the tight relationship of logic programming under the perfect model semantics (Przymusinski 1988; Apt, Blair, & Walker 1988) and the well-founded semantics (Van Gelder, Ross, & Schlipf 1991) with Iterated Inductive Definitions, a nonmonotone form of inductive definition. This study puts forward the *thesis* that the well-founded semantics formalises a principle of generalised monotone and nonmonotone induction definition. In (Denecker 2000b), this view was further elaborated in ID-logic, a logic extending classical logic with generalised inductive definitions.

Another view on logic programs came from the area of Nonmonotonic Reasoning. (Gelfond 1987) proposed to interpret a logic program as an autoepistemic theory. In this view, *failure to prove* literals `not p` are interpreted as modal literals $\neg Kp$ in autoepistemic logic. Gelfond proposed to interpret a logic programming rule

```
p :-  q, not r
```

as the following AEL formula:

$$p \leftarrow q \wedge \neg Kr$$

In the sequel, I will refer to this embedding as *ael* and denote the mapping of a logic program P as $ael(P)$.

(Gelfond & Lifschitz 1988) based the stable semantics on this embedding. They showed that a stable model of a program P is the set of atoms in an autoepistemic expansion of $ael(P)$. They showed also that the least model of a Horn theory is its unique stable model and argued that the stable semantics correctly models the meaning of e.g. the transitive closure program. Importantly, this result suggest that the nonmonotone view on Horn programs coincides with the view of Horn programs as inductive definitions. However, as will be shown, there is a snake under the grass.

A number of alternative embeddings based on a similar idea have been proposed. I only mention another seminal one of (Marek & Truszczyński 1989) in Default Logic (DL). It maps a rule

```
p :-  q, not r
```

to the default:

$$\frac{q : \neg r}{p}$$

This embedding of program P will be denoted $dl(P)$. Marek and Truszczyński showed that a stable model of P is the set of atoms in a default extension of $dl(P)$.

In view of the fact that a declarative logic should have a non-ambiguous declarative meaning, the question rises to what extend these different readings correspond to each other. Logic programming can be seen as a family of logics, each induced by a pair of a syntax and a formal semantics. The logics of normal programs with the completion semantics, the stable semantics and the well-founded semantics partially coincide, for example in the case of hierarchical programs. In general, we should expect that (a) the declarative reading underlying these semantics are *equivalent* in these coinciding cases. Also, we should expect that (b) each logic in this family has a unique declarative reading. If it has been claimed that one and the same semantics formalises different readings, then these readings must be equivalent in some deep sense.

In fact, it is easy to show that neither (a) nor (b) holds. Consider the case of Horn programs and as an illustration, take the simple non-recursive Horn program:

$$P_1 = \{p \leftarrow q\}$$

The first point is that P_1 is a non-recursive Horn program. For such programs, all semantics coincide. In this case, the empty set {} is the unique least model, the unique model of the completion, the unique stable model and the unique well-founded model. So, all formal model semantics coincide on P_1.

Now, let us compare the meaning of P_1 as expressed by the different embeddings on which these model semantics are based. The comparison can be done on

[1] A well-known fact is that inductive definable concepts such as the concept of transitive closure cannot be expressed in first order logic.

a formal basis, by comparing the *belief sets* associated with these embeddings. A belief set is a set T of first order formulas that contains each of its implications. For any first order theory T, let $Cn(T)$ denote the set of its logical implications: $Cn(T) = \{\phi | T \models \phi\}$. Then T is a belief set iff $T = Cn(T)$. A belief set is often taken as a representation of the belief state of some ideally rational agent. Both the semantics of default logic and autoepistemic logic are expressed via belief sets.

- For non-recursive Horn programs, the definition view is correctly expressed by the completion. $comp(P_1)$ is the theory

  $$\left\{ \begin{array}{l} q \leftrightarrow false \\ p \leftrightarrow q \end{array} \right\}$$

  The reason why q is known to be false is that it is assumed to have the empty definition. The unique belief set of the view of P_1 as a definition is:

  $$Cn(comp(P_1)) = Cn(\{\neg q, \neg p\})$$

- According to Gelfond's mapping, the meaning of P_1 is the autoepistemic theory $ael(P_1)$:

  $$\{p \leftarrow q\}$$

  In (Moore 1983), Moore defined the semantics of autoepistemic logic in terms of *autoepistemic expansions*. As shown in the same paper, the set of classical logic formulas in an expansion E is deductively closed and completely determines E. Moreover, Moore also showed that autoepistemic logic extends classical logic in the following sense: if an autoepistemic theory T contains no occurences of the modal operator, then it has a unique expansion which is completely determined by $Cn(T)$. Consequently, ael interprets each Horn program as a Horn theory. $ael(P_0)$ has one autoepistemic expansion extending the unique belief set:

  $$Cn(\{p \leftarrow q\})$$

- The meaning of P_1 according to the embedding in default logic is the default $dl(P_1)$:

  $$\left\{ \frac{q :}{p} \right\}$$

  In (Reiter 1980), Reiter characterised the semantics of a default theory by default extensions, which are belief sets. The unique default extension of this set is:

  $$Cn(\{\})$$

  I.e. this is the set of tautologies. The reason that this is the unique default extension is that the justification of the default cannot be derived, and hence, the default does not apply. Note that the implication $p \leftarrow q \notin Cn(\{\})$. So the belief represented by this default is even weaker than the classical logic interpretation of the rule.

On the one hand, we observe that the least model, the model of the completion, the stable and the well-founded model of P_1 coincide. On the other hand, we observe that the three embeddings assign a different belief set to P_1: respectively $Cn(\{\neg p, \neg q\})$, $Cn(\{p \leftarrow q\}$ and $Cn(\{\})$. These three belief sets are non-equivalent and have different first order models. Consequently, P_1 is an example of a program for which all formal model semantics coincide, but for which the declarative reading is different under the three different views. So it illustrates that point (a) does not hold.

P_1 also illustrates that ael and dl assign a different meaning to logic programs. Nevertheless, both embeddings have been shown to induce stable semantics. So, this example also illustrates that point (b) does not hold.

The ambiguity illustrated by P_1 boils down to the following phenomenon. Assume that we have a logic program written by some expert to represent his knowledge. Assume moreover that we know which formal model semantics was intended by the expert and that we even know what are the models of the program. Still, we are unable to know what is the knowledge of the expert and how the world looks like according to the expert.

A famous case of a study showing epistemological ambiguity of a knowledge representation formalism is Woods' study of semantic networks in 1975 (Woods 1975). Woods showed that a link in semantic networks could be and had been interpreted in at least 3 different ways. A major conclusion drawn from this paper by the AI community (e.g. in Hayes "In defense of logic" (Hayes 1977) or Newell's "The knowledge level" (Newell 1980)) was that a declarative language needs a formal account of its meaning. The phenomenon of the ambiguity of LP addressed in the current paper is similar in nature to the ambiguity of links pointed out by Woods. What makes LP's ambiguity even more surprising and worrysome is that unlike semantic nets in the middle seventies, it arises in the context of logics with model semantics. In the context of logic programming, *formal semantics do not seem to guarantee a non-ambiguous declarative reading*!

How is it possible that even if we fix the formal semantics, logic programs are still ambiguous?

Half of the explanation is that the definition view and the nonmonotonic views assign a different role to a "model". In the definition view, a model represents a *possible state of the world*. In the nonmonotone views, it represents a *set of believed atoms*. In both views, the same mathematical structures are used to represent two very different sorts of entities.

This observation has some unsettling consequences. Comparing sets of believed atoms and possible worlds is as comparing apples and oranges. Many mathematical results relating different model semantics are meaningless at the epistemological level! E.g. at the epistemological level, the mathematical result that a stable model is a model of the completion is as informative as to say that 2 kilometer is less than 4 kilogram. The same holds for the mathematical result that the stable model of a Horn program P is its least model. This is true, but as illustrated by P_1, the view of P as a monotone inductive definition does not coincide with $ael(P)$ nor $dl(P)$.

The different roles of "models" cannot be the only explanation of the ambiguity since in the embeddings dl and ael, the role of the stable model is the same. Here another explanation is in order, namely that a set of believed atoms only provides weak and incomplete information about the meaning of a theory. In general, many different belief sets may share the same atoms[2]. In this particular case, for almost all programs P, the theories $dl(P)$ and $ael(P)$ have different belief sets, but there the sets of believed atoms in these belief sets correspond.

Epistemological clarity is a sine qua non for declarative logic. For anyone who wishes to consider logic programming logics as declarative logic, the ambiguity phenomenon poses a fundamental problem that must be resolved. The first aim of this abstract is to bring about the epistemological ambiguity. A more extended study can be found at (Denecker 2000a). Its main goal are as follows:

- There is currently little understanding and appreciation for the issue of epistemological foundations of logic in the field of logic programming, and more in general, in many logic-based disciplines in AI. This study is a plea for clear epistemological foundations of logic and uses logic programming as a test case to show what can go wrong in absence of such foundations. To this end, it investigates the impact of the epistemological ambiguity of LP at practical and fundamental levels.
- The goal is not only to point to the problem and its effects but also to clarify the different declarative views on logic programming and its extensions Abductive Logic Programming and Answer set programming. This is the first step towards solving the epistemological ambiguity. The paper points out some directions.

References

Aczel, P. 1977. An Introduction to Inductive Definitions. In Barwise, J., ed., *Handbook of Mathematical Logic.* North-Holland Publishing Company. 739–782.

Apt, K.; Blair, H.; and Walker, A. 1988. Towards a theory of Declarative Knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming.* Morgan Kaufmann.

Clark, K. 1978. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Databases.* Plenum Press. 293–322.

Denecker, M. 1998. The well-founded semantics is the principle of inductive definition. In Dix, J.; nas del Cerro, L. F.; and Furbach, U., eds., *Logics in Artificial Intelligence*, 1–16. Schloss Daghstull: Springer-Verlag, Lecture notes in Artificial Intelligence 1489.

Denecker, M. 2000a. What is in a model? Epistemological ambiguity of Logic Programming. Draft; available on the internet: http://www.kuleuven.ac.be/ marcd/papers/ambiguity.ps.

Denecker, M. 2000b. Extending classical logic with inductive definitions. In et al., J., ed., *First International Conference on Computational Logic (CL2000)*, volume 1861 of *Lecture notes in Artificial Intelligence*, 703–717. London: Springer.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of the International Joint Conference and Symposium on Logic Programming*, 1070–1080. IEEE.

Gelfond, M. 1987. On Stratified Autoepistemic Theories. In *Proc. of AAAI87*, 207–211. Morgan Kaufman.

Hayes, P. 1977. In defense of logic. In *Proc. of the 5th IJCAI77.*

Kowalski, R. 1974. Predicate logic as a programming language. In *Proc. of IFIP 74*, 569–574. North-Holland.

Marek, V., and Truszczyński, M. 1989. Stable semantics for logic programs and default reasoning. In E.Lust, and Overbeek, R., eds., *Proc. of the North American Conference on Logic Programming and Non-monotonic Reasoning*, 243–257.

Moore, R. 1983. Semantical Considerations on non-monotonic logic. In *Proc. of IJCAI-83*, 272–279.

Moore, R. 1995. *Logic and Representation*, volume CSLI Lecture Notes No39. Center for the Study of Language and Information, Standford, California: CSLI Publications.

Moschovakis, Y. N. 1974. *Elementary Induction on Abstract Structures.* North-Holland Publishing Company, Amsterdam- New York.

Newell, A. 1980. The knowledge level. *artifical Intelligence* 18:87–127. Also presidential addres at AAAI 80.

Przymusinski, T. 1988. On the semantics of Stratified Databases. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming.* Morgan Kaufman.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81–132.

van Emden, M., and Kowalski, R. 1976. The semantics of Predicate Logic as a Programming Language. *Journal of the ACM* 4(4):733–742.

[2]The readings of a Horn program P as a classical logic Horn theory, or as an inductive definition or as $ael(P)$ or as $dl(P)$ differ but yield the same set of believed atoms.

Van Gelder, A.; Ross, K.; and Schlipf, J. 1991. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM* 38(3):620–650.

Woods, W. 1975. What's in a link: Foundations of semantic networks. In Bobrow, D., and Collins, A., eds., *Representation and understanding: Studies in cognitive science*. Academic Press, New York. Also in Brachman and Levesque, *Readings in Knowledge Representation*, Morgan Kaufman, 1985.