A Model Theory for Paraconsistent Logic Programming

Carlos Viegas Damásio and Luís Moniz Pereira*

CRIA, Uninova and DCS, U. Nova de Lisboa
2825 Monte da Caparica
Portugal
{cdllmp}@fct.unl.pt

Abstract. We provide a nine-valued logic to characterize the models of logic programs under a paraconsistent well-founded semantics with explicit negation $WFSX_p$. We define a truth-functional logic, $\mathcal{NINE}$, based on the bilattice construction of Ginsberg and Fitting. The models identified by $WFSX_p$ are models of logic $\mathcal{NINE}$. We conclude with a discussion on the conditions to obtain an isomorphism between the two definitions, and thereby characterizing $WFSX_p$ model-theoretically.

1 Introduction

One of the main issues in logic programming is the definition of semantics for negation(s). Quite recently, a second form of negation besides the older default negation, was proposed by several authors [12, 8, 11, 21, 13]) providing a mechanism for explicitly declaring the falsity of literals, which was not available before. The importance of extending LP with a second kind of negation $\neg$, has been stressed for use in deductive databases, knowledge representation, and non-monotonic reasoning. Different semantics for extended LPs with $\neg$-negation have appeared (e.g. [8, 18, 13, 22]). The specific generalization for extended programs of well-founded semantics [7], $WFSX$, defined in [13, 2] using "explicit negation", is taken as the base semantics in this paper.

The introduction of explicit negation requires being able to reason with, or at least detect, contradictory knowledge. Indeed, information is not only normally incomplete but contradictory as well. As remarked by [22] there are three main ways of dealing with inconsistent information:

Explosive approach: If the program is contradictory then every formula is derived from it. This corresponds to the usual approach in mathematical logic, and of several semantics for extended logic programs [18, 8, 13, 2].

Belief revision approach: The program is revised in order to regain consistency. This is the view adopted by some authors in the LP community [17, 14, 10, 1, 2]. It does not necessarily require an explicit paraconsistent semantics: the procedural revision operators suffice.

* We thank Esprit BR project Compulog 2 (no. 6810), and JNICT for their support.
Paraconsistent approach: Accept contradictory information and perform reasoning tasks that take it into account. This is the approach of [4, 19, 22], and the one we will follow in this paper.

The first approach is rather naïve and only makes sense when dealing with mathematical objects. For instance, if we have a large knowledge base being maintained or updated by different agents, it is natural to encounter inconsistencies in the database. Most of the time, this inconsistency is local to some part of the knowledge base and it shouldn’t affect other, independent, information. If we adopt the explosive approach, and a single contradiction is found then we must discard the entire knowledge base. This is uneconomical.

Sometimes the contradictory information can be due to a specification error, and we’d like to fix it through debugging. In other situations the information provided is in itself contradictory. In the former, we can use belief revision techniques. In the latter, a paraconsistent deductive mechanism is necessary. Notice however that to perform belief revision we need in any case to detect the inconsistencies and the reasons supporting them. Thus, paraconsistent reasoning is an, at least implicit, intermediate step to attain belief revision.

Since we want to assign meaning to every program we will make use of the paraconsistent version of WFSX, WFSX_p which can be found in [1, 2]. The semantics complies with two basic principles: coherence and also a form of introspection. The “coherence principle” of [13, 2] relates the two forms of negation, default and explicit: it stipulates that the latter entails the former, i.e. if L (¬L) is entailed then so is not ¬L (not L). In other words, coherence requires that if I’m convinced of the truth of a proposition then I must believe (i.e. be weakly convinced about) the truth of the proposition.

The introspection mechanism too provides the derivation of new weak convictions. To express it we need the notion of “doubt”: I doubt the truth of L iff I have weak conviction for the falsity of L; I have conviction in the truth of L iff I doubt the weak conviction in the falsity of L. Now we can state the principle of introspective doubt: if I doubt all the bodies of rules for L then I’m weakly convinced of the falsity of L. The joint application of these principles is brought out in example 1.

Example 1. Let P be the extended logic program containing the five rules \{a; ¬a; b ← a; c ← not b; d ← not d\}. It is clear that the model of this program must entail a, ¬a and b; therefore I’m convinced of the truth of a and b (a and b are true) and convinced of the falsity of a (¬a is true). By applying coherence we should also have not a, not ¬a and not ¬b, i.e. I’m weakly convinced of the falsity of a (not a is true) and weakly convinced of the truth of a and b (not ¬a and not ¬b are true). By doubt introspection, I believe in the falsity of b (not b is true) if I have doubts about the truth of a; and indeed I’m reserved about the truth of a because I believe in the falsity of a, i.e. not a is true. Therefore not b, and thus c, should belong to the model. By the same introspective doubt, I’m weakly convinced of the falsity of c (not c is true) if I doubt my belief in the falsity of b, i.e. if I’m convinced of the truth of b; this is so (b is true), therefore not c