Declarative Semantics for Inconsistent Database Programs

Marion Sarkis Mircheva
Fac. of Mathematics and Computer Science
Dep. of Logic, Sofia University
James Bouichier str., 5, Sofia 1126

Abstract. In this paper an ongoing research on inconsistent database programs is presented, which can be viewed as an extension of logic programming. By Literal Database Program (LDP) we mean a finite set of universally closed clauses of the form \( L_0 \leftarrow L_1, \ldots, L_m \) where \( m \geq 0 \), \( L_i \)'s are literals. We define a set of intended models \( MOD(P) \) for a given program \( P \) with the help of interpretations defined on an extension of the Herbrand base. Instead of resolving inconsistency at all costs, our system is aimed to supply an inference control after the appearance of some primitive inconsistent units. It keeps the good conflict knowledge, namely those, that come from different, equally trusted sources and proposes mechanism telling one how to act when the primitive inconsistency arises.

A characterization of the semantics for the proposed literal database programs is given by means of general properties of non-monotonic inference operations. Some points for future considerations are also announced.

1 Introduction

The formalization of a reasoning process based upon contradictory databases is a critical issue, as most large knowledge bases are suspected to be inconsistent. We consider an inconsistent database where all data in the base are equally trusted. Such systems are useful for multi-expert databases, where knowledge of several experts is pooled in a single program. We allow for Logic programs to have rules with both \( A \) and \( \neg A \) in heads and bodies and have any number of independent advisors. The incoming information could be provided from different, equally reliable sources and inconsistency may appear as a result of merging this information. Then the system have to consider what happens when several advisors (expert rules or facts) appear to be in conflict. Generally, the contradictions may involve several steps of reasoning, produce new contradictions or nested themselves in complicated deductions. Either we may have several simple data items or some formal reasonable rules, which together give the contradictory answers, but no single item is to blame. How do we cope with our system in this case?

The existing proposals concerning the problem of reasoning in inconsistent databases can be classified according to the methods and underlying philosophy of treating contradictory knowledge. We separate systems, that define new inference machinery, mainly by modifying the semantics of implication and the other logical connectives. These types of methods are based on some relevant, many-valued or paraconsistent logics. The paper of Lin [10] describes computational mechanisms for defining relevant conclusions from a set of formulae and specify the correct answers that do not depend on "erroneous data". The relevance character of the implication is defined in a syntactic way with the help of the restriction only from the clauses which are used explicitly in a refutation, when a resolution strategy is applied. Some advantages in using paraconsistent logic for knowledge representation in the presence of contradictions are discussed in [2]. The second type of methods offer the possibility of supporting conflicting conclusions
by isolating them in multiple extensions. These methods share the popular idea of creating consistent contexts in the inconsistent database. The classical example is default logic [16] and its variations. Although we can consider default rules as clauses with appropriate semantics, such systems are well adapted to the representation of rules with exceptions, rather than general productions. The paper of Nute [14] is an example of the approach based on using the specialized rules in order to resolve the inconsistency. The work of Martins and Shapiro [13] offers a relevant approach for dealing with inconsistency database where any formula of the knowledge base is associated with the consistent set of axioms from which one can derive it. Similar ideas for management with different consistent subbases of the inconsistent database are presented in the paper of Cholvy [3]. His formalism is supposed to limit the impact of errors to information “related” to erroneous information and is aimed to give the answers to queries by distinguishing reliable information and suspicious ones. In that case suspicious means produced by the union of special minimal inconsistent subsets of the given ones. As in the majority of clausal form reasoning systems, the database consists of definite clauses and a set of integrity constrains presented as a disjunction of negative literals. The presentation of that kind (even without allowing normal programs) is too restrictive to cope with contradictions in the deduction.

A special place among systems devoted to resolve consistency in the inconsistent contexts belongs to the Truth Maintenance System of Doyle [5] and ATMS of De Kleer [4]. Strictly speaking their systems do not perform an inference but are designed with the aim of maintaining the consistency of a set of deductions. They record inferences transmitted to them from an external deduction system and they are in charge of the maintenance of the consistency of a set of assertions. They allow for the management of assumptions, default values and contradictions. According to the incoming values they propose one set of deducible information units. Similarly to “multiple extension” approach the underlying philosophy of TMS is to eradicate inconsistency at all costs. Gabbay and Hunter [8] propose some form of truth maintenance systems to be used in restoring consistency as a result of some action, provided by an object-level database. Hence TMS could serve as a supplementary tool for debugging an inconsistent database. The papers [7,8] suggest a general framework for treating inconsistency as a good thing, especially as “the norm we should feel happy to be able to formalize”. They suggest an approach quite different from the existing at present philosophy in dealing with inconsistency in the deductive databases. According to them, inconsistency in logical systems should provide the effect of reasoning more similar to that of human reasoning, namely:

$$\text{inconsistency} \rightarrow \text{action}$$

In the paper [8] inconsistency is classified according to the appropriate action that is required and might be adopted in a supplementary to original database. The work [6] is an example how inconsistency could be studied as a Labelled Deductive System, where the basic units of information are labelled formulae and the deduction is defined on labelled formulae. Some special cases of deductions can serve as a trigger for performing an action. The idea of taking an appropriate control over the derivation exists in different formalisms dealing with inconsistent knowledge base [15], but in the case of labelled formulae, the labels are necessary in deciding which action to call.

Reasoning in inconsistent contexts can be naturally considered as an integral part of non-monotonic systems. In [9] a certain number of criteria of a technical nature have been proposed for purposes of comparison.

The approach we took, differs from all mentioned above by the method of knowledge representation and by the treatment of contradictions which are deductible in the course of the inference process. We define a Literal Database Program (LDP) as a finite set of clauses of the form:

$$L_0 \leftarrow L_1, \ldots, L_n$$