INTRODUCTION

Nowadays, data management on the World Wide Web needs to consider very large knowledge databases (KDB). The larger is a KDB, the smaller the possibility of being consistent. Consistency checking algorithms and systems fail to analyze very large KDBs, and so many have to work every day with inconsistent information.

Database revision—transformation of the KDB into another, consistent database—is a solution to this inconsistency, but the task is computationally untractable. Paraconsistent logics are also a useful option to work with inconsistent databases. These logics work on inconsistent KDBs but prohibit undesired inferences. From a philosophical (logical) point of view, the paraconsistent reasoning is a need that the self-human discourse practices. From a computational, logical point of view, we need to design logical formalisms that allow us to extract useful information from an inconsistent database, taking into account diverse aspects of the semantics that are “attached” to deductive databases reasoning (see Table 1).

The arrival of the semantic web (SW) will force the database users to work with a KDB that is expressed by logic formulas with higher syntactic complexity than are classic logic databases.

BACKGROUND

Logic databases are based on the formalisms of first order logic (FOL); thus, they inherit a classical semantics that is based on models. Also, they can be interpreted within a proof-theoretic approach to logical consequence from the logic programming paradigm (Lloyd, 1987). The extended database semantics paradigm is developed to lay before the foundations of query-answering tasks and related questions (see Minker, 1999), but its aim is not to deal with inconsistencies. The data cleaning task may involve—in the framework of repairing logic databases—logical reasoning and automated theorem proving (Boskovitz, Goré, & Hegland, 2003).

On the other hand, new paradigms, such as SW, need new formalisms to reason about data. Description logics (DL) provide logic systems based on objects, concepts, and relationships, with which we can construct new concepts and relations for reasoning (Baader, Calvanese, McGuiness, Nardi, & Patel-Schneider, 2003). Formally, DL are a subset of FOL, and the classical problems on consistency remain, but several sublogics of DL provide nice algorithms for reasoning services. The ontology Web language (OWL; its DL-sublanguage) is a description logic designed for automated reasoning, not only designed for the classical ask–tell paradigm. With languages such as OWL, ontologies exceed their traditional aspects (e.g., taxonomies and dictionaries) to be essential in frameworks as data integration.

The classical notion of inconsistency in databases mainly deals with the violation of integrity constraints. This notion must be expanded because of the new notion of logic databases in SW, in which ontologies and data both play the same role in knowledge management. Therefore, there are several sources of inconsistency (see Table 2). This role is not only limited to the database but also includes the verification and validation task of knowledge-based systems (Bench-Capon, 2001). Inconsistency arises in the initial steps of ontology building due to several reasons and not only by the updating of data. In general, the repair of a logic database involves the study of...
of the soundness and perhaps completeness (i.e., the method output’s only correct solutions and all the relevant solutions). Semantics would support reasoning services such as self-consistency, checking the relations between concepts (as subsumption), and classification of objects according to the ontology.

Systems exist in which both paradigms, classical and SW logic databases, are conciliated under the extension of the former (see, e.g., Pan & Hehlin, 2003). However, the relation between DL and database models may not be fruitfully formalized because of the limited expressiveness of the DL system selected to make the reasoning feasible (see chapter 4 in Baader et al., 2003).

INCONSISTENCY HANDLING

Solutions that are suggested to work in the presence of inconsistencies can be classified according to different views (see Table 3, where several references appear). The first aspect—and maybe the most important—is the compatibility between the original semantics of the KDB source and the logical formalism selected to handle inconsistency. From this point of view there exist paraconsistent logics that limit the inference power of FOL to avoid undesired answers and also modal logics for representing different aspects of the information sources. These approaches manage semantics that are essentially different from the semantics of KDBs. On the other side we can find methods that classify, order, or both, interesting subsets according to the original semantics of the KDB, such as the argumentative approach or the integration of data by fusion rules, but they do not repair the KDB. Other methods also exist that propose how the KDB should be revised (e.g., integrity constraints of the extensional database). However, it is necessary to point out that the automated knowledge revision is an essentially different task in the case of ontologies, because the ontology source represents a key organisation of the knowledge of the owner and, as in every logical theory, minor changes may produce unexpected and dangerous anomalies.

Another point of view concerns the share of the KDB that is repaired when an anomaly is found. According to this, the methods based on arguments mentioned earlier can be used to repair only the anomalous argument. Due to the high complexity of consistency checking algorithms, to preserve consistency under updates is a better option than repairing. In the case of evolving ontologies, new systems such as KAON infrastructure are needed (for more information, see http://kaon.semanticweb.org/kaon).

There are methods dealing with the enforcement of consistent answers (i.e., answers that satisfies integrity constraints) from inconsistent databases; it is done by