Chapter XVIII
A Fixpoint Semantics for Rule–Base Anomalies

Du Zhang
California State University, USA

ABSTRACT

A crucial component of an intelligent system is its knowledge base that contains knowledge about a problem domain. Knowledge base development involves domain analysis, context space definition, ontological specification, and knowledge acquisition, codification and verification. Knowledge base anomalies can affect the correctness and performance of an intelligent system. In this chapter, we describe a fixpoint semantics for a knowledge base that is based on a multi-valued logic. We then use the fixpoint semantics to provide formal definitions for four types of knowledge base anomalies: inconsistency, redundancy, incompleteness, circularity. We believe such formal definitions of knowledge base anomalies will help pave the way for a more effective knowledge base verification process.

INTRODUCTION

Computing plays a pivotal role in our understanding of human cognition (Pylyshyn, 1989). The classical cognitive architecture for intelligent behavior assumes that both computers and minds have at least the following three distinct levels of organization (Pylyshyn, 1989). (a) The semantic level or the knowledge level where the behavior of human beings or appropriately programmed computers can be explained through the things they know and the goals they have. It attempts to establish, in some meaningful or even rational ways, connections between the actions (by human or computer) and what they know about their world. (b) The symbol level where the semantic content of knowledge and goals is assumed to be encoded through structured symbolic expressions. It deals with representation, structure and manipulation of symbolic expressions. (c) The physical or biological level where the physical embodiment of an entire system (human or computer) is considered. It encompasses the structure and the principles by which a physical object functions.

Pylyshyn’s cognitive penetrability criterion states that “the pattern of behavior can be altered in a rational way by changing subjects’ beliefs about the task” (Pylyshyn, 1989). It is the subjects’ tacit knowledge about the world, not the properties of the architecture that enables such behavior adjustment.
The hallmark of a knowledge-based system is that by design it possesses the ability to be told facts about its world and to alter its behavior accordingly (Brachman & Levesque, 2004). It exhibits the property of cognitive penetrability.

Today, knowledge-based systems not only play an important role in furthering the study in cognitive informatics (Wang et al., 2002; Patel et al., 2003; Chan et al., 2004; Kinsner et al., 2005; Wang, 2002, 2007; Wang and Kinsner, 2006), but also have found their way into so many problem domains (Cycorp, 2006) and have been utilized to generate numerous successful applications (IBM, 2006; Ross, 2003). A crucial component of an intelligent system or a knowledge-based system is its knowledge base (KB) that contains knowledge about a problem domain (Brachman & Levesque, 2004; Fagin et al, 1995; Levesque & Lakemeyer, 2000). Knowledge base development involves domain analysis, context space definition, ontological specification, and knowledge acquisition, codification and verification (Zhang, 2005).

When developing a KB for an application, it is important to recognize the context under which we formulate and reason about domain-specific knowledge. A context is a region in some n-dimensional space (Lenat, 1998). In a KB development process, domain analysis should result in identification of the region of interest in the context space. Specifying a context entails specifying or locating a point or region along each of those n dimensions. Once the context (or contexts) for a problem domain is identified, ontological development is in order. An ontology is a formal, explicit specification of a shared conceptualization (Chandrasekaran et al, 1999; Gomez-Perez et al, 2004; O’Leary, 1998). After the conceptualization is in place, knowledge acquisition, codification and verification can be carried out to build the KB for some application. Inevitably, there will be anomalies in a KB as a result of existing practices in its development process. Knowledge base anomalies can affect the correctness and performance of an intelligent system, though some systems are robust enough to perform rationally in the presence of the anomalies. It is necessary to define KB anomalies formally before identifying where they are in a KB and deciding what to do with them.

In this chapter, our focus is on formal definitions of KB anomalies and on the issue of how to identify them. Our attention is on rule-based KB. A rule-based KB has a set of facts that is stored in a working memory (WM) and a set of rules stored in a rule base (RB). Rules represent general knowledge about an application domain. They are entered into a RB during initial knowledge acquisition or subsequent KB updates. Facts in a WM provide specific information about the problems at hand and may be elicited either dynamically from the user during each problem-solving session, or statically from the domain expert during knowledge acquisition process, or derived through rule deduction.

We assume that rules in a KB have the following format: \( P_1 \land \ldots \land P_n \rightarrow R \), where \( P_i \) are the conditions (collectively, the left-hand side, LHS, of a rule), \( R \) is the conclusion (or right-hand side, RHS, of a rule), and the symbol “\( \rightarrow \)” is understood as the logical implication. The \( P_i \) and \( R \) are literals. If the conditions of a rule instance are satisfied by facts in WM, then the rule is enabled and its firing deposits its conclusion into WM.

A fact is represented as a ground atom. It specifies an instance of a relationship among particular objects in the problem domain. WM contains a collection of positive ground atoms which are deposited through either assertion (initial or dynamically), or rule deduction.

A negated condition \( \neg p(x) \) in the LHS of a rule is satisfied if \( p(x) \) is not in WM for any \( x \). A negated ground atom \( \neg p(a) \) in the LHS of a rule is satisfied if \( p(a) \) is not in WM. A negated conclusion \( \neg R \) in the RHS of a rule results in the removal of \( R \) from WM, when the LHS of the rule is satisfied. Rule instances and negated literals can be utilized by the inference system, but are never deposited into WM (Ginsberg & Williamson, 1993).

Let WM\(_0\) denote the initial state for WM. We use WM\(_i\), \( i = 1,2,3,\ldots \) to represent subsequent states of WM as a result of firing all enabled rules under the state of WM\(_{i-1}\). For the basic concepts and terminology in the first order predicate logic, readers are referred to (Ben-Ari, 1993; Chang & Lee, 1973).

The rest of the chapter is organized as follows. Section 2 offers a brief review of the related work. Section 3 discusses the four types of KB anomalies. Section 4 describes the fixpoint semantics we adopt for a KB. Formal definitions of the KB anomalies are given in Section 5 in terms of the fixpoint semantics. Finally Section 6 concludes the chapter with remark on future work.